IDEA – International Data Encryption Algorithm

IDEA originalmente chamava-se IPES e o seus autores são X. Lai e J. Massey (artigo *X. Lai and J. Massey: A Proposal for a new Block Encryption Standard, Eurocrypt 90, Springer-Verlag 1991, pp 389-404*). É um algoritmo com chave secreta de 128 bits e tanto o texto legível (entrada) como o ilegível (saída) de 64 bits. Foi projetado para ser eficiente em implementações por software. É um algoritmo com patente nos EUA e Europa por Ascom-Tech AG.

IDEA possui uma estrutura semelhante ao DES: possui um número fixo de iterações (rounds) de um mesma função que utiliza subchaves distintas, e o mesmo algoritmo serve para criptografar e decriptografar alterando-se apenas a forma de geração das subchaves. É conhecido publicamente desde 1991 e até agora não há notícia de ter revelado qualquer vulnerabilidade apesar de ter sido criptanalisado por especialistas da comunidade por vários anos.

1. As três operações básicas do IDEA

O conceito de projeto dominante é a aplicação composta de três operações distintas de três grupos algébricos sobre 2^{16} elementos (i.e., dois bytes). Se A, B, C denotam três elementos de 16 bits, as três operações são:

1. Ou-exclusivo (XOR) sobre 16 bits, que será representada pelo símbolo $\oplus$, i.e., $A = B \oplus C$; note que $B \oplus C \oplus C = B$, ou seja, conhecendo-se A e C pode-se obter B.

2. soma $\bmod 2^{16}$, que é equivalente à soma usual em que o bit mais à esquerda correspondente ao valor 2^{16} deve ser sempre igual a zero após a soma; esta operação será denotada pelo símbolo $\boxplus$, i.e., $A = B \boxplus C$; note que se $\overline{C}$ é o inverso de $C \bmod 2^{16}$ (i.e. $\overline{C} + C = 2^{16} = 0 \bmod 2^{16}$), então $B \boxplus C \boxplus \overline{C} = B$, ou seja, conhecendo-se A e $\overline{C}$ pode-se obter B.

3. a terceira operação é representada pelo símbolo $\odot$, e é um pouco mais complicada que as anteriores: $A = B \odot C$ significa

   (a) multiplicar (da forma usual) B por C, obtendo-se um valor de 33 bits (no máximo), digamos Z, sendo que antes de multiplicar, se $B = 0$, altere-o para 2^{16}, e se $C = 0$, altere-o para 2^{16}.

(b) depois calcular o resto da divisão de Z por $2^{16}+1$ (i.e., calcular $Z \bmod (2^{16}+1)$),

(c) e se o resultado for 2^{16}, que excede 16 bits, o resultado final da operação deve ser zero, i.e., $A = 0$.

Em resumo: (onde $\times$ denota a multiplicação usual)

(i) se $(B \times C) \bmod (2^{16}+1) \neq 2^{16}$, então $B \odot C = (B \times C) \bmod (2^{16}+1)$, trocando zero por 2^{16} antes de aplicar $\times$,

(ii) senão $B \odot C = 0$.

Ademais, como $2^{16}+1$ é um número primo, qualquer elemento C entre 1 e 2^{16} possui um inverso $C^{-1} \bmod (2^{16}+1)$ (i.e., $C \times C^{-1} = 1 \bmod (2^{16}+1)$ e C^{-1} pode ser calculado eficientemente pelo Algoritmo de Euclides estendido), de modo que $B \odot C \odot C^{-1} = B$, ou seja, conhecendo-se A e C^{-1} pode-se obter B.

2. Geração das subchaves

A partir da chave secreta K de 128 bits são geradas 52 subchaves de 16 bits que chamaremos $K_1, K_2, K_3, ..., K_{52}$. As primeiras 8 subchaves são geradas simplesmente considerando os primeiros 16 bits da esquerda para a direita de K como sendo K_1, os 16 bits seguintes de K como sendo K_2, e assim por diante, os últimos 16 bits de K como sendo K_8.

128 bits $\rightarrow$	16	16	16	16	16	16	16	16
de K	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8

A seguir as 8 subchaves seguintes são geradas de forma análoga mas a K_9 começando após 25 bits à direita do extremo esquerdo de K, a K_{10} começando após $25+16 = 41$ bits à direita do extremo esquerdo de K, e assim por diante, a K_{15} começa após $25 + 6 * 16 = 121$ bits à direita do extremo esquerdo de K. Por enquanto a K_{15} contém apenas 7 bits. Os 9 bits restantes da K_{15} começam no primeiro bit de K como se uma cópia de K estivesse concatenado à direita de K como na figura abaixo. Isto equivale a ter deslocado circularmente para a *esquerda* a chave K, de 25 posições, *antes* de iniciar a geração de cada uma das 8 subchaves, e K_9 começar no *primeiro* bit.

128 bits → de K	25	16 K_9	16 K_{10}	16 K_{11}	16 K_{12}	16 K_{13}	16 K_{14}	7 parte K_{15}

9 parte K_{15}	16 K_{16}	16	16	16	16	16	16	7 ← 128 bits de K

As 8 subchaves seguintes são geradas de forma análoga mas a K_{17} começando após 50 bits à direita do extremo esquerdo de K, a K_{18} começando após $50+16=66$ bits à direita do extremo esquerdo de K, e assim por diante, a K_{21} começa após $50+4*16=114$ bits à direita do extremo esquerdo de K, e termina no segundo bit de K como se uma cópia de K estivesse concatenado à direita de K. E a K_{22} começa no terceiro bit de K, como na figura abaixo. Isto equivale a ter deslocado circularmente para a *esquerda* a chave K, de 25 posições além das 25 anteriores, *antes* de iniciar a geração de cada uma das 8 subchaves, e K_{17} começar no *primeiro* bit

128 bits → K	50	16 K_{17}	16 K_{18}	16 K_{19}	16 K_{20}	14 parte K_{21}

2 parte K_{21}	16 K_{22}	16 K_{23}	16 K_{24}	16	16	16	16	14 ← 128 bits de K

As subchaves K_{25} a K_{32} são geradas como na figura a seguir:

128 bits → de K	75	16 K_{25}	16 K_{26}	16 K_{27}	5 parte K_{28}

11 parte K_{28}	16 K_{29}	16 K_{30}	16 K_{31}	16 K_{32}	16	16	16	5 ← 128 bits de K

As subchaves K_{33} a K_{40} são geradas como na figura a seguir:

128 bits → de K	100	16 K_{33}	12 parte K_{34}

4 parte K_{34}	16 K_{35}	16 K_{36}	16 K_{37}	16 K_{38}	16 K_{39}	16 K_{40}	16	12 ← 128 bits de K

As subchaves K_{41} a K_{48} são geradas como na figura a seguir:

128 bits → de K	125	3 parte K_{41}

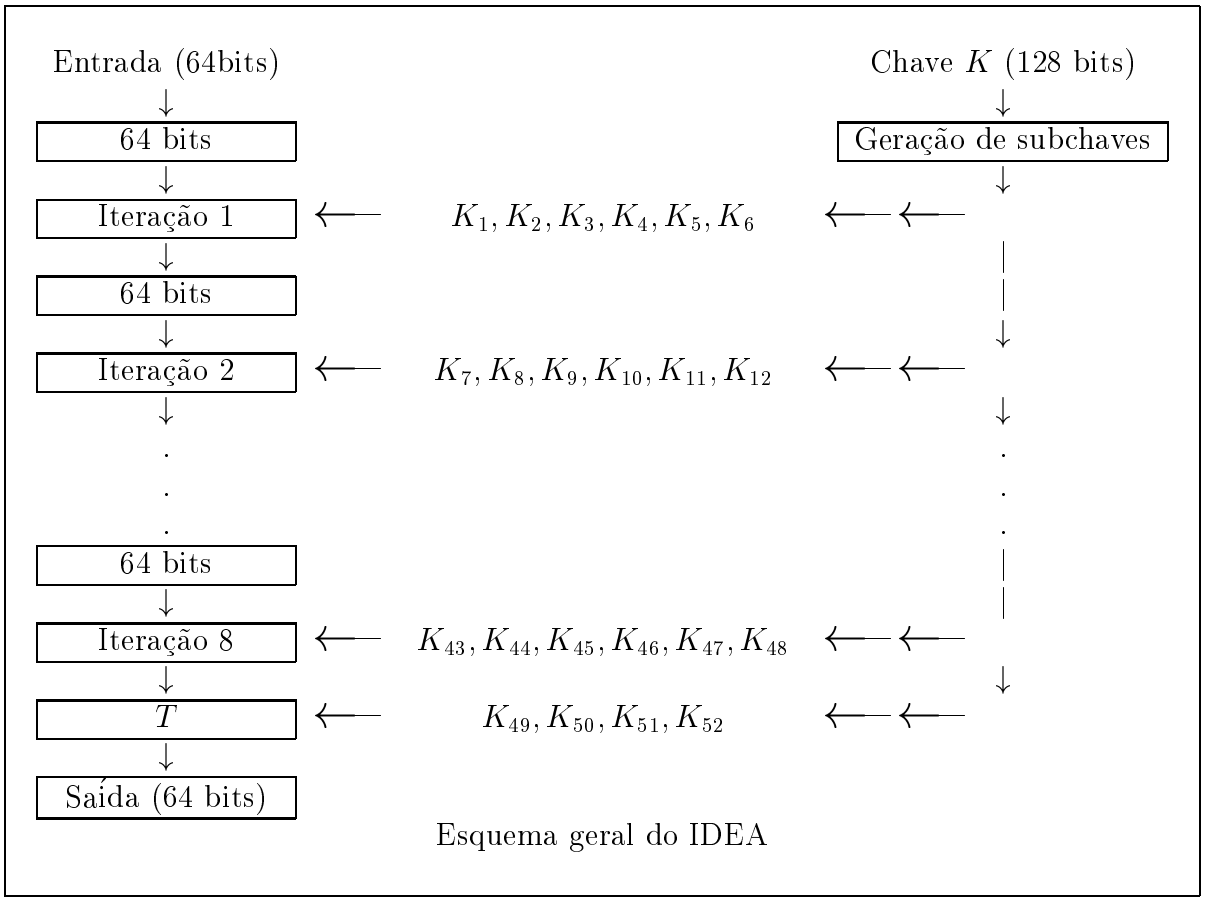
13 parte K_{41}	16 K_{42}	16 K_{43}	16 K_{44}	16 K_{45}	16 K_{46}	16 K_{47}	16 K_{48}	3 ← 128 bits de K

As últimas subchaves K_{49} a K_{52} são geradas como na figura a seguir:

128 bits	$\rightarrow$	22	16	16	16	16	16	16	10
de K			K_{49}	K_{50}	K_{51}	K_{52}			

3. Uma iteração (round) do IDEA

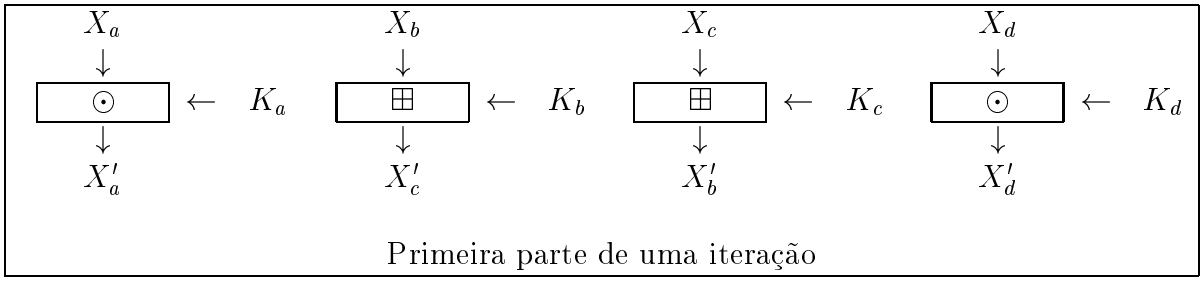
IDEA possui 8 iterações (ou rounds), de forma semelhante ao DES, e uma transformação final que chamaremos T. A transformação T utiliza as últimas 4 subchaves: $K_{49}, K_{50}, K_{51}, K_{52}$, da maneira que descreveremos mais adiante. Cada iteração utiliza 6 subchaves e possui duas partes que descreveremos a seguir.



Esquema geral do IDEA

3.1. Primeira parte de uma iteração

Esta parte utiliza 4 subchaves que chamaremos K_a, K_b, K_c, K_d. A sua entrada é de 64 bits, tratada como 4 subentradas de 16 bits que chamaremos X_a, X_b, X_c, X_d. Após certas operações aplicadas sobre esta entrada, a sua saída será constituída de novas versões destes X_a, X_b, X_c, X_d que chamaremos X'_a, X'_b, X'_c, X'_d. Na primeira iteração, $K_a = K_1, K_b = K_2, K_c = K_3, K_d = K_4$, e na segunda iteração $K_a = K_7, K_b = K_8, K_c = K_9, K_d = K_{10}$, e assim por diante.



Primeira parte de uma iteração

As operações são as seguintes:

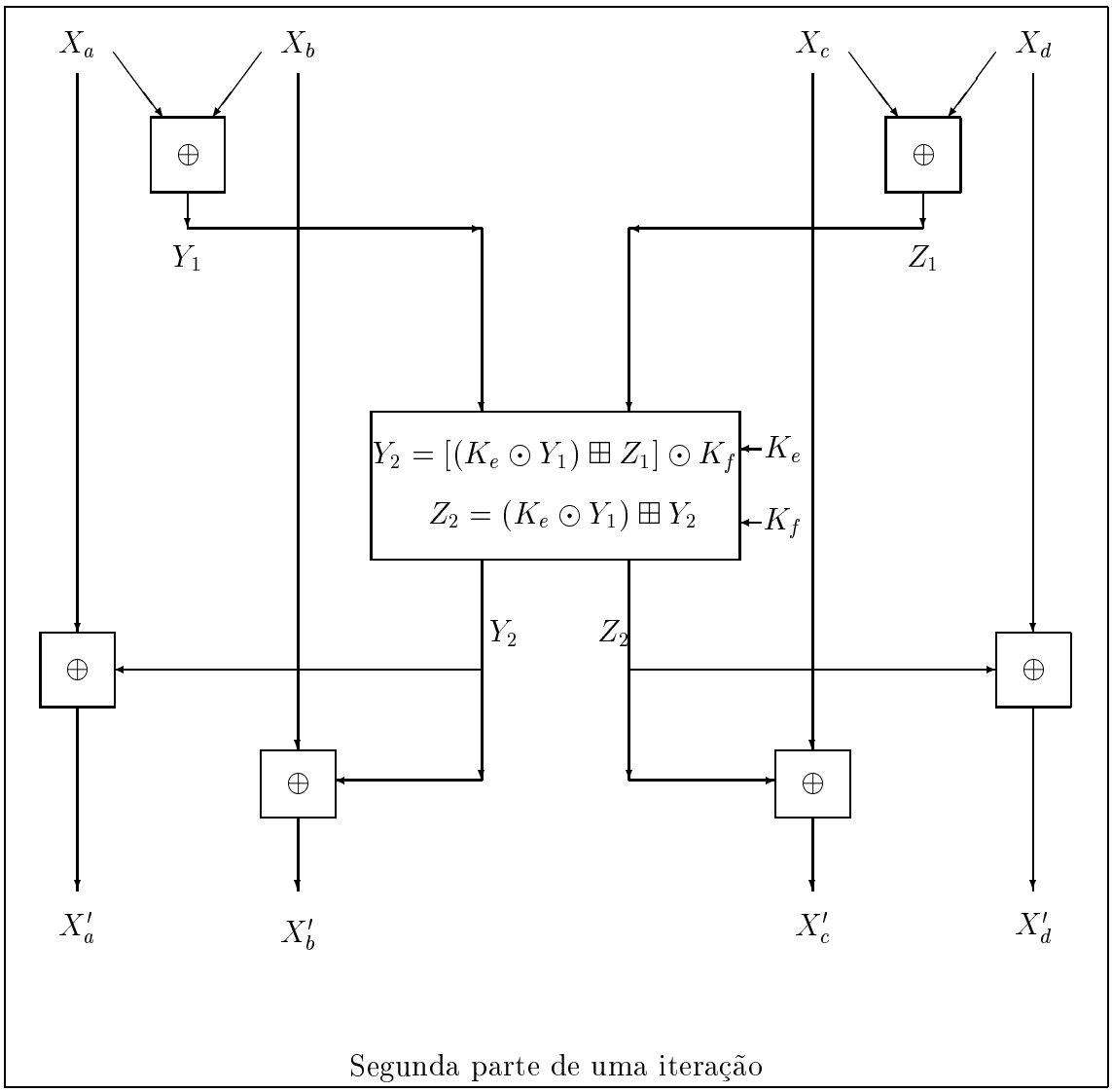
1. X'_a é $X_a \odot K_a$
2. X'_d é $X_d \odot K_d$
3. X'_b é $X_c \boxplus K_c$
4. X'_c é $X_b \boxplus K_b$

Note que o resultado desta parte, em ordem, é X'_a, X'_b, X'_c, X'_d.

Observe que estas 4 operações são inversíveis. Para se obter X_a a partir de X'_a basta termos calculado previamente a inversa multiplicativa K_a^{-1}pois $X'_a \odot K_a^{-1} = X_a \odot K_a \odot K_a^{-1} = X_a$. De forma análoga, pode se obter X_d a partir de X'_d. E para se obter X_c a partir de X'_b basta termos calculado previamente a inversa aditiva $\overline{K_c}$ pois $X'_b \boxplus \overline{K_c} = X_c \boxplus K_c \boxplus \overline{K_c} = X_c$. E de forma análoga, pode se obter X_b a partir de X'_c.

3.2. Segunda parte de uma iteração

Esta parte utiliza 2 subchaves que chamaremos K_e, K_f. A sua entrada é a saída da primeira parte, de 64 bits, tratada de novo como 4 subentradas de 16 bits que chamaremos X_a, X_b, X_c, X_d. Após outras operações aplicadas sobre esta entrada, a sua saída será constituída de novas versões destes X_a, X_b, X_c, X_d que, de novo, chamaremos X'_a, X'_b, X'_c, X'_d. Na primeira iteração, $K_e = K_5, K_f = K_6$, e na segunda iteração $K_e = K_{11}, K_f = K_{12}$, e assim por diante.



Segunda parte de uma iteração

1. Inicialmente são calculados dois valores intermediários chamados Y_1 e Z_1 da seguinte forma:

    1. $Y_1 = X_a \oplus X_b$
    2. $Z_1 = X_c \oplus X_d$

2. A seguir outros dois valores intermediários chamados Y_2 e Z_2 são calculados:

    1. $Y_2 = [(K_e \odot Y_1) \boxplus Z_1] \odot K_f$
    2. $Z_2 = (K_e \odot Y_1) \boxplus Y_2$

3. E os valores X'_a, X'_b, X'_c, X'_d são calculados da seguinte maneira:

    1. $X'_a = X_a \oplus Y_2$
    2. $X'_b = X_b \oplus Y_2$
    3. $X'_c = X_c \oplus Z_2$
    4. $X'_d = X_d \oplus Z_2$

Uma observação *muito importante* aqui é que a inversa desta segunda parte é a própria! Isto é, se a entrada para a segunda parte for X'_a, X'_b, X'_c, X'_d com subchaves K_e, K_f, obtém-se X_a, X_b, X_c, X_d de volta!! Para justificarmos isso, observe os seguintes passos:

1. Tendo X'_a, X'_b na entrada à esquerda, o início (Passo (1) anterior) desta segunda parte efetua $X'_a \oplus X'_b = (X_a \oplus Y_2) \oplus (X_b \oplus Y_2) = X_a \oplus X_b = Y_1$. E tendo X'_c, X'_d na entrada à direita, o início desta segunda parte efetua $X'_c \oplus X'_d = (X_c \oplus Z_2) \oplus (X_d \oplus Z_2) = X_c \oplus X_d = Z_1$. Portanto, os valores Y_1 e Z_1 são reconstruídos exatamente como no Passo (1) anterior.

2. Este passo para recalcular Y_2 e Z_2 é também exatamente igual ao Passo (2) anterior.

    1. $Y_2 = [(K_e \odot Y_1) \boxplus Z_1] \odot K_f$
    2. $Z_2 = (K_e \odot Y_1) \boxplus Y_2$

3. Este passo também é igual ao Passo (3) anterior pois:

1. $X'_a \oplus Y_2 = (X_a \oplus Y_2) \oplus Y_2 = X_a$
2. $X'_b \oplus Y_2 = (X_b \oplus Y_2) \oplus Y_2 = X_b$
3. $X'_c \oplus Z_2 = (X_c \oplus Z_2) \oplus Z_2 = X_c$
4. $X'_d \oplus Z_2 = (X_d \oplus Z_2) \oplus Z_2 = X_d$

Em resumo, a criptografia nesta segunda parte é exatamente igual à decriptografia, não necessitando qualquer cálculo de chave inversa (que a primeira parte exige).

3.3. A última transformação T

Após 8 iterações da primeira e segunda partes como descrito acima, o resultado X'_a, X'_b, X'_c, X'_d é fornecido como entrada para a última transformação T.

Como mencionado anteriormente, a transformação T utiliza as últimas 4 subchaves: $K_{49}, K_{50}, K_{51}, K_{52}$. E esta tranformação é quase igual à primeira parte de uma iteração, exceto que K_{50} é aplicado sobre X'_c e K_{51} é aplicado sobre X'_b:

1. X_a^{FINAL} é $X'_a \odot K_{49}$
2. X_d^{FINAL} é $X'_d \odot K_{52}$
3. X_b^{FINAL} é $X'_c \boxplus K_{50}$
4. X_c^{FINAL} é $X'_b \boxplus K_{51}$

4. Decriptografia pelo algoritmo IDEA

O projeto do IDEA foi bem feito de maneira que o mesmo circuito ou software serve tanto para criptografar como para decriptografar 64 bits. Para decriptografar, basta calcular previamente as subchaves inversas (apenas da primeira parte de cada iteração) e inverter a ordem em que as chaves são utilizadas, isto é,

1. Utilizar primeiro as subchaves inversas das chaves $K_{49}, K_{50}, K_{51}, K_{52}$ na primeira parte da primeira iteração. K_1 e K_4 na decriptografia são as inversas multiplicativas em relação a $\oplus$ respectivamente de K_{49} e K_{52}, e K_2 e K_3 na decriptografia são as inversas aditivas em relação a $\boxplus$ respectivamente de K_{50} e K_{51}.

2. A seguir utilizar as subchaves K_{47}, K_{48} na segunda parte da primeira iteração.

3. A seguir utilizar as subchaves inversas das chaves $K_{43}, K_{44}, K_{45}, K_{46}$ na primeira parte da segunda iteração.

e assim por diante, até utilizar as subchaves K_1, K_2, K_3, K_4 na transformação T final da decriptografia.

5. Dados para teste

Listamos abaixo os dados em notação hexadecimal (base16) referentes à criptografia e decriptografia em cada iteração de um teste.

	Chave $K = (1,2,3,4,5,6,7,8)$						Entrada $(0,1,2,3)$			
	(128 bits)						(64 bits)			
iter.	K_a	K_b	K_c	K_d	K_e	K_f	X_a	X_b	X_c	X_d
1	0001	0002	0003	0004	0005	0006	00f0	00f5	010a	0105
2	0007	0008	0400	0600	0800	0a00	222f	21b5	f45e	e959
3	0c00	0e00	1000	0200	0010	0014	0f86	39be	8ee8	1173
4	0018	001c	0020	0004	0008	000c	57df	ac58	c65b	ba4d
5	2800	3000	3800	4000	0800	1000	8e81	ba9c	f77f	3a4a
6	1800	2000	0070	0080	0010	0020	6942	9409	e21b	1c64
7	0030	0040	0050	0060	0000	2000	99d0	c7f6	5331	620e
8	4000	6000	8000	a000	c000	e001	0a24	0098	ec6b	4925
T	0080	00c0	0100	0140	–	–	11fb	ed2b	0198	6de5

	Chave $K = (1,2,3,4,5,6,7,8)$						Entrada de 64 bits			
	(128 bits)						$(11fb, ed2b, 0198, 6de5)$			
iter.	K_a	K_b	K_c	K_d	K_e	K_f	X_a	X_b	X_c	X_d
1	fe01	ff40	ff00	659a	c0000	e001	d98d	d331	27f6	82b8
2	fffd	8000	a000	cccc	0000	2000	bc4d	e26b	9449	a576
3	a556	ffb0	ffc0	52ab	0010	0020	0aa4	f7ef	da9c	24e3
4	554b	ff90	e000	fe01	0800	1000	ca46	fe5b	dc58	116d
5	332d	c800	d000	fffd	0008	000c	748f	8f08	39ds	45cc
6	4aab	ffe0	ffe4	c001	0010	0014	3266	045e	2fb5	b02e
7	aa96	f000	f200	ff81	0800	0a00	0690	050a	00fd	1dfa
8	4925	fc00	fff8	552b	0005	0006	0000	0005	0003	000c
T	0001	fffe	fffd	c001	–	–	0000	0001	0002	0003

6. Exercício

Se B e C são tais que $0 \leq B, C \leq 2^{16}$, a operação $B \odot C = (B \times C) \bmod (2^{16}+1)$ pode ser implementada eficientemente da seguinte forma:

1. Seja $A = B \times C = a_0 2^{32} + a_1 2^{16} + a_2$, e então $a_0 = 0$ ou $a_0 = 1$ e $0 \leq a_1, a_2 < 2^{16}$.

2. Se $B = C = 2^{16}$, note que $a_0 = 1, a_1 = a_2 = 0$, e então $(B \times C) \bmod (2^{16}+1) = (-1)(-1) = 1$, pois $2^{16} = -1 \bmod (2^{16}+1)$.

3. Senão, $a_0 = 0$. Obter inicialmente a_1 e a_2 por multiplicação usual $B \times C$.

Neste último caso pede-se para provar que:

(a) se $a_1 \leq a_2$, então $(B \times C) \bmod (2^{16}+1) = a_2 - a_1$.

(b) se $a_1 > a_2$, então $(B \times C) \bmod (2^{16}+1) = a_2 - a_1 + (2^{16}+1)$ pois $-2^{16} < a_2 - a_1 < 0$.

Routo Terada (DCC-USP) 1999.