CRIPTOGRAFÍA

GRADO EN INGENIERÍA INFORMÁTICA

GRUPO 16

SIGN THEN ENCRYPT

Variante 2

Curso 2019/2020

Jorge Rodríguez Fraile, 100405951, Grupo 81, 100405951@alumnos.uc3m.es
Carlos Rubio Olivares, 100405834, Grupo 81, 100405834@alumnos.uc3m.es
Francisco José Ruiz de la Cruz, 100405807, Grupo 81, 100405834@alumnos.uc3m.es
Iván Serrano García,100405836,Grupo 81, 100405836@alumnos.uc3m.es

Índice

Especificación del sistema **3**
Representación de datos. 3
Algoritmos y esquemas. 3
Funciones resumen a realizar: 3
Derivación de la Clave de sesión: 4
Cifrador de bloque CBC: 5
Claves asimétricas 5
Coherencia de representaciones y tamaños de bloques y espacios de trabajo 7

Comunicación de las partes **8**

2. Especificación del sistema

2.1. Representación de datos.

La representación de los mensajes se realiza haciendo una sustitución de cada letra por su correspondiente valor numérico, las letras van de “A” a “Z” (sin “Ñ”), por lo que empezará en el 0 con la “A” y terminará en el 25 con la “Z”, no tendremos en cuenta espacios.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

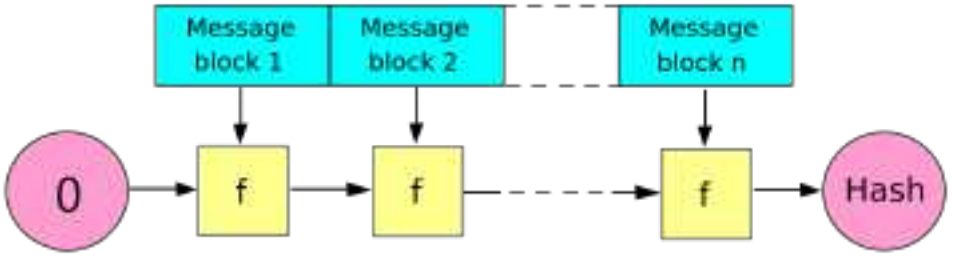
Table 2: Encoding English capital letters using integers from $\mathbb{Z}_{25}$.

Los identificadores (ID) se han definido cogiendo la codificación de las 3 primeras letras de su nombre y sumando sus valores

2.2. Algoritmos y esquemas.

Funciones resumen a realizar:

Merkle–Damgård modificado:



Esta función es la que utilizamos para el cifrado de los mensajes que se intercambian Alicia y Benito.

Consiste en dividir el mensaje en n bloques de 5 bits de longitud de izquierda a derecha y si es necesario se rellena el último bloque con 0’s a la derecha. Después se hace XOR de los distintos bloques que hemos formado y la salida de la función es la salida del último bloque.

En esta versión modificada el vector inicial es 0 y en el último bloque no se adjunta la longitud del mensaje.

Sor + Xor:

Utilizamos una función resumen llamada SOR junto con un XOR, que se basa en el siguiente proceso:

Hacemos un XOR entre bloques de tamaño n que ocupen la misma posición en los dos mensajes, es decir el bloque 1 con el 1, el 2 con el 2 y así sucesivamente. Entre los bloques resultado del paso anterior se aplica un XOR, dando lugar a la salida deseada. Si se da el caso de que un mensaje no tiene los bits suficientes para dividirlo en bloques de n bits, se rellenará con 0's a la derecha.Ejemplo:
(1111 0100 SOR 1001 1010) = (1111 XOR 1001) XOR (1001 XOR 1010)

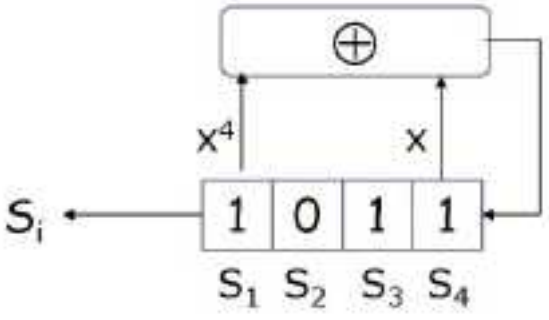
Una vez hecho este SOR,hacemos un XOR con el Id de la entidad para completar el hash.

Derivación de la Clave de sesión:

LFSR:

El vector de inicialización de CBC se ha obtenido a partir del algoritmo LFSR.

LFSR consiste en que a partir de un polinomio usado como semilla realizamos desplazamientos a izquierdas de los bits de la K_SESSION a la vez que hacemos un XOR con los bits cuya posición coincide con el grado de las X's del polinomio semilla y cuyo resultado nos otorgará el bit que entrará en el siguiente paso por la derecha. Repetimos este proceso hasta que la secuencia de bits resultante sea igual a la inicial, o bien, hasta que lleguemos al paso $2^n - 1$ siendo n el número de bits que tiene la secuencia de bits.



CLAVE DE ENCRIPTACIÓN:

Por otro lado, hemos obtenido la clave de encriptación mediante un NOT a nuestra KEY_SESSION, y rellenando los bits faltante con 1's.

NOT(K_sesion)||11111 11111 11111

Cifrador de bloque CBC:

ShiftRow+AddRoundKey, los bloques serán de 20 bits de longitud, divididos en grupos de 5 para representar los caracteres y números pertinentes.

En cuanto el ShiftRow lo hacemos con 4 bloques de 5 bits, los primeros 5 no se desplazan, los siguientes se desplazan 1 vez hacia la izquierda, los siguientes 2 veces…Hasta que hayamos terminado con los 20 bits.

Una vez hayamos obtenido estos 20 bits desplazados, hacemos un XOR a este resultado con nuestra K_ENCRYPTION.
Cabe recalcar que solo se hace 1 ronda para cada bloque de 20 bits.

2.3. Claves asimétricas

Los procesos asimétricos utilizados han sido RSA, mayoritariamente para firmar las funciones resumen de los mensajes enviados, y Diffie-Hellman para el intercambio de la clave de sesión.

Las claves utilizadas en Diffie-Hellman han sido:

Claves de Alicia: (3, 35)
- Clave pública = 3
- Clave privada = 35

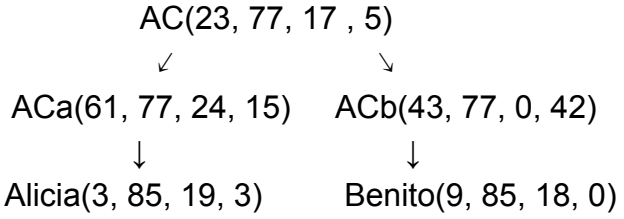
Claves de Benito: (9, 22)
- Clave pública = 9
- Clave privada = 22

Clave K_sesion
- Ka = 15
- Kb = 15

Las parejas de claves utilizadas para el intercambio de las claves k-sesion han sido las siguientes. Se adjunta también el certificado asociado a dicha clave:

Certificados:

Nombre(kPública, N, ID , F(KprivadaAutoridadSuperior; H(Kpublica, N, ID))



AC(KuAC =23, NAC = 77, ID_AC = 17 , F(KvAC; H(KuAC, NAC, ID_AC)) = 5)

KvAC = 23^(-1) mod 60= 47

H(KuAC, NAC, ID_AC) = (1011 1000 SOR 1001 1010) XOR 1000 1000 = (0010 XOR 0010) XOR 1000 1000 = 0000 0000 XOR 1000 1000 = 1000 1000 = 136

F(KvAC; H(KuAC,NAC, ID_AC)) = 136^47 mod 77 = 5

ACa(KuACa = 61, NACa = 77 , ID(SECA)= 24,F(KvAC; H(KuACa, NACa, ID_ACa)) = 15)

KvACa = 61^(-1) mod 60 = 1

H(KuACa, NACa, ID_ACa)= (1111 0100 SOR 1001 1010) XOR 1100 0000= (0110 XOR 1110) XOR 1100 0000 = 1000 0000 XOR 1100 0000 = 0100 0000 = 64

F(KvAC; H(KuACa, NACa, ID_ACa)) = 64^47 mod 77 = 15

Alicia(A = 3, NA = 85, ID_A= 19,F(KvACa; H(A, NA, ID_A)) = 3)

KvA = 3 ^(-1) mod 4*16 = 43

H(A, NA, ID_A)= (1100 0000 SOR 1010 1010) XOR 1001 1000 = (0110 XOR 1010) XOR 1001 1000 = 1100 0000 XOR 1001 1000 = 0101 1000 = 88

F(KvACa; H(A, NA**,** ID_A)) = 88^1 mod 77= 11

ACb(KuACb = 43, NACb = 77 , ID_ACb= 0 , F(KvAC; H(KuACb, NACb)) = 42)

KvACb = 43^(-1) mod 60 = 7

H(KuACb, NACb, ID_ACb)= (1101 0000 SOR 1001 1010) XOR 0000 = (0100 XOR 1010) XOR 0000 = 1110 XOR 0000 = 1110 = 14

F(KvAC; H(KuACb, NACb, ID_ACb)) =14^47 mod 77 = 42

Benito(B = 9, NB = 85, ID_B= 18 , F(KvACb; H(B, NB))=0)

KpB = 9^(-1) mod 4*16 = 57

H(B, NB, ID_B) = (1001 0000 SOR 1010 1010) XOR 1001 0000 = (0011 XOR 1010)XOR 1001 0000 = 1001 0000 XOR 1001 0000 = 0000 0000 = 0

F(KpACb; H(B, NB, ID_B)) = 0^7 mod 77= 0

2.4. Coherencia de representaciones y tamaños de bloques y espacios de trabajo

Utilizamos el número de bits necesarios para que todos los números posibles en nuestro módulo puedan ser representados sin ningún problema. En nuestro caso, se utilizan 8 bits para la estructura del certificado y 5 para representación de caracteres.

Por otro lado, en CBC utilizamos bloques de 20 bits, que pensamos que es un tamaño razonable para los cálculos, aunque es menos seguro.

En cuanto a la jerarquía de reparto de claves públicas, hemos tenido en cuenta que las claves privadas sean inversas de la clave pública de Φ(mod), y por supuesto, que estas claves públicas sean coprimas de Φ(mod).

3. Comunicación de las partes

Alicia quiere establecer una comunicación segura con Benito. Para ello primero deben acordar una clave de sesión para poder determinar el resto de los valores que necesitaran para la comunicación asimétrica. Esto lo conseguimos mediante el algoritmo de intercambio de claves Diffie-Hellman.

Alicia envía a Benito: K_u, p, g y la cadena de certificados.

Número primo “p”: 17
Generador “g”: 7

K_v:

Clave a: 35

K_u:

A = 7^35 mod 17 = (7^5)^7 mod 17 = 4^4 mod 17 = 3 mod 17

Certificados:

AC(23, 77, 17 , 5)
ACa(61, 77, 24, 15)
Alicia(3, 85, 19, 3)

Después de esto, Benito verifica la clave pública de Alicia. Esto se realiza mediante la verificación de los certificados expedidos por las autoridades pertinentes.

Primero verificamos el certificado de Alicia:

H(Alicia)= 11^61 mod 77 = 11
H(3, 85, 19)= (1100 0000 SOR 1010 1010) XOR 1001 1000 = 0101 1000 = 88 mod 77 = 11

Después verificamos el certificado ACa:

H(ACa)= 15^23 mod 77 = 64
H(61, 77 , 24)= (1111 0100 SOR 1001 1010) XOR 1100 0000 = 0100 0000 = 64

Por último verificamos la Autoridad de certificado en la que confía Benito, que se autoafirma:

H(AC)= 5^23 mod 77 = 59
H(23, 77, 17)=(1011 1000 SOR 1001 1010) XOR 1000 1000=1000 1000 = 136 mod 77 = 59

Viendo que la verificación ha sido correcta, y que la autoría del mensaje es efectivamente de Alicia, Benito procede a enviar a Alicia K_u y la cadena de certificados.

K_v:

Clave b: 22

K_u:

B = 7^22 mod 17 = (7^2)^11 mod 17 = 15^11 mod 17 = (3^11 * 5 ^11) mod 17 = 9 mod 17

Certificados:

AC(23, 77, 17 , 5)
ACb(43, 77, 0, 42)
Benito(9, 85, 18, 0)

Una vez enviado todo esto, Alicia procede a verificar la identidad de Benito, para ello realiza los siguientes pasos:

Primero verificamos el certificado de Benito:

H(Benito)= 0^43 mod 77= 0
H(9, 85, 18)= (1001 0000 SOR 1010 1010) XOR 1001 0000 = 0000 0000 = 0

Después verificamos el certificado ACb:

H(ACb)= 42^23 mod 77 = 14
H(43, 77, 0)= (1101 0000 SOR 1001 1010) XOR 0000 = 1110 = 14

Por último verificamos la Autoridad de certificado en la que confía Alicia, que se autofirma:

H(AC)= 5^23 mod 77 = 59
H(23, 77, 17)=(1011 1000 SOR 1001 1010) XOR 1000 1000=1000 1000 = 136 mod 77 = 59

Una vez Alicia ha autentificado la clave pública de Benito, cada uno conoce la clave pública del otro, pudiendo hallar la clave de sesión paralelamente.

K_sesion

Ka = 9^35 mod 17 = 15
Kb = 3^22 mod 17 = 15

Antes de que Alicia mande su mensaje, los dos extremos de la comunicación deben derivar la K_session para obtener los elementos necesarios para encriptar o desencriptar los mensajes. Estos elementos son el vector de inicialización IV y la K_Encrypt. Para obtenerlos utilizamos dos algoritmos de derivación:

LFSR para la obtención de IV:

Para implementar este algoritmo usamos la K_session y una semilla dada:

x^3+x^2+1

x-01111-0

0-11110-0
1-11100-1
1-11001-0
1-10010-1
1-00101-1

0-01011-1
0-10111-0
1-01110-0
0-11100-1
1-11001-0

1-10010-1
1-00101-1
0-01011-1
0-10111-0
1-01110-0

0-11100-1
1-11001-0
1-10010-1
1-00101-1
0-01011-1

El IV serán los primeros 20 bits generados por el LFSR.

IV: 01111 00101 11001 01110

Para obtener el K_Encrypt, utilizamos un NOT a partir de la K_Session y rellenamos con 1’s a la derecha tantas veces como bits necesitemos:

K_Encrypt = NOT(01111)||11111 11111 11111 = 10000 11111 11111 11111

Una vez generada K_ENCRYPT e IV, Alicia aplica el hash correspondiente y firma su ID concatenado con el mensaje mediante una firma RSA tipo 2, obteniendo así SIGMA. Este resultado se une al mensaje quedando así ID||mensaje||SIGMA, esta tira de bits se dividen en bloques de 20 bits, y pasan a cifrarse simétricamente mediante CBC, cada bloque se encripta como se ha indicado anteriormente y se obtiene el mensaje encriptado de la siguiente manera:

M1 = 19||NEC ESIT OUNA LATA =

10011 01101 00100 0001000100 10010 01000 1001101110
10100 01101 0000001011 00000 10011 00000

H(M1)= 11010 = 26

SIGMA = F(H(M1))= 26^43 mod 85 = 66

El mensaje que encripta Alicia es ID_A||M1||F:

10011 01101 00100 00010 00100 10010 01000 10011
01110 10100 01101 00000 01011 00000 10011 00000
00000 00000 00010 00010

Ahora procedemos a encriptar simétricamente en bloque CBC:

01111 00101 11001 01110
10011 01101 00100 00010

P1= IV XOR B1 = 11100 01000 11101 01100

ShiftRow hacia izquierda P1:

11100 11100
01000 10000
11101 10111
01100 00011

Obtenemos P1 = 11100 10000 10111 00011

AddRoundKey de P1:

11100 10000 10111 00011
10000 11111 11111 11111 XOR

C1= 01100 01111 01000 11100

P2= C1 XOR B2 = 01100 01111 01000 11100 XOR 00100 10010 01000 10011 = 01000 11101 00000 01111

ShiftRow de P2:

01000	01000
11101	11011
00000	00000
01111	11011

Obtenemos P2 = 01000 11011 00000 11011

AddRoundKey de P2

01000 11011 00000 11011
10000 11111 11111 11111 XOR

C2= 11000 00100 11111 00100

P3 = C2 XOR B3 = 11000 00100 11111 00100 XOR 01110 10100 01101 00000 = 10110 10000 10010 00100

ShiftRow de P3:

10110	10110
10000	00001
10010	01010
00100	00001

Obtenemos P3 = 10110 00001 01010 00001

AddRoundKey de P3

10110 00001 01010 0001
10000 11111 11111 11111 XOR

C3= 00110 11110 10101 11110

P4= C3 XOR B4 =
00110 11110 10101 11110 XOR 01011 00000 10011 00000 = 01101 11110 00110 11110

ShiftRow de P4:

01101	01101
11110	11101
00110	11000
11110	10111

Obtenemos P4 = 01101 11101 11000 10111

AddRoundKey de P4

01101 11101 11000 10111
10000 11111 11111 11111 XOR

C4 = 11101 00010 00111 01000

P5 = C4 XOR B5 =
11101 00010 00111 01000 XOR 00000 00000 00010 00010 = 11101 00010 00101 01010

ShiftRow de P5:

11101	11101
00010	00100
00101	10100
01010	10010

Obtenemos P5 = 11101 00100 10100 10010

AddRoundKey de P5

11101 00100 10100 10010
10000 11111 11111 11111 XOR

C5 = 01101 11011 01011 01101

C = 01100 01111 01000 11100 11000 00100 11111 00100 00110 11110 10101 11110 11101 00010 00111 01000 01101 11011 01011 01101

Este mensaje se envía a Benito (que ha obtenido K_SESSION y ha derivado K_ENCRYPT e IV) y hace un decrypt en CBC (indicado en los cálculos) y obtiene ID||mensaje||SIGMA, ahora bien, para poder comprobar la firma, se ha establecido que estas se encuentren en los 10 últimos bits del

mensaje, y que se rellene con 0's a la izquierda. Benito comprueba la firma RSA y finalmente comprueba los resultados:

Inverso de AddRoundKey para C1:

C1= 01100 01111 01000 11100
XOR 10000 11111 11111 11111

11100 10000 10111 00011

Inverso de ShiftRow para C1

11100	11100
10000	01000
10111	11101
00011	01100

Deshacemos el XOR con el IV:

11100 01000 11101 01100 XOR 01111 00101 11001 01110 =
10011 01101 00100 00010

Inverso de AddRoundKey para C2:

C2= 11000 00100 11111 00100
XOR 10000 11111 11111 11111

01000 11011 00000 11011

Inverso de ShiftRow para C2

01000	01000
11011	11101
00000	00000
11011	01111

Deshacemos el XOR con el bloque cifrado anterior:

01000 11101 00000 01111 XOR 01100 01111 01000 11100 =
00100 10010 01000 10011

Inverso de AddRoundKey para C3:

C3= 00110 11110 10101 11110
XOR 10000 11111 11111 11111

10110 00001 01010 00001

Inverso de ShiftRow para C3

10110	10110
00001	10000
01010	10010
00001	00100

Deshacemos el XOR con el bloque cifrado anterior:

10110 10000 10010 00100 XOR 11000 00100 11111 00100 =
01110 10100 01101 00000

Inverso de AddRoundKey para C4:

C4= 11101 00010 00111 01000
XOR 10000 11111 11111 11111

_____

01101 11101 11000 10111

Inverso de ShiftRow para C4

01101	01101
11101	11110
11000	00110
10111	11110

Deshacemos el XOR con el bloque cifrado anterior:

01101 11110 00110 11110 XOR 00110 11110 10101 11110 =
01011 00000 10011 0000

Inverso de AddRoundKey para C5:

C5= 01101 11011 01011 01101
XOR 10000 11111 11111 11111

_____

11101 00100 10100 10010

Inverso de ShiftRow para C5

11101	11101
00100	00010
10100	00101
10010	01010

Deshacemos el XOR con el bloque cifrado anterior:

11101 00010 00101 01010 XOR 11101 00010 00111 01000 =
00000 00000 00010 000010

Mensaje Descifrado completo:

10011||01101 00100 00010 00100 10010 01000 10011 01110
10100 01101 00000 01011 00000 10011 00000 00000
00000||00010 000010

los primeros 5 bits→ id = 19
últimos 10 bits → firma = 66
Texto en claro: NECESITOUNALATAAA

Obtenemos que la firma de Alicia es 10 00010, que es 66, ahora procedemos a verificarla, para ello calculamos el Hash del mensaje que acabamos de descifrar, que debe coincidir con la verificación de la firma con la clave pública de Alicia:

H(M1) = 26
V(F) = 66^3 mod 85 = 26 → coincide con H(M1), por lo que la firma queda verificada

Una vez verificado, Benito realiza los mismos pasos para enviarle un mensaje de respuesta a Alicia, encripta el mensaje:

M2 = 18||VAN ENCA MINO AHOR A =
10010 10101 00000 01101 00100 01101 00010 00000 01100 01000
01101 01110 00000 00111 01110 10001 00000

H(M2)=11110 = 30
SIGMA = F(H(M2)) = 30^57 mod 85 = 30

El mensaje que se encripta de Benito es ID_B||M2||F:

10010 10101 00000 01101 00100 01101 00010 00000 01100
01000 01101 01110 00000 00111 01110 10001 00000 00000
00000 11110

Ahora procedemos a encriptar simétricamente en bloque CBC:

IV=01111 00101 11001 01110
B1=10010 10101 00000 01101

P1=IV XOR B1 = 11101 10000 11001 00011

ShiftRow hacia izquierda P1:

11101 11101
10000 00001
11001 00111
00011 11000

Obtenemos P1= 11101 00001 00111 11000

AddRoundKey de P1:

11101 00001 00111 11000 XOR
10000 11111 11111 11111
_____

C1 = 01101 11110 11000 00111

P2 = C1 XOR B2 = 01001 10011 11010 00111

ShiftRow hacia izquierda P2:

01001 01001
10011 00111
11010 01011
00111 11001

Obtenemos P2= 01001 00111 01011 11001

AddRoundKey de P2:

01001 00111 01011 11001 XOR
10000 11111 11111 11111
_____

C2 = 11001 11000 10100 00110

P3 = C2 XOR B3 = 10101 10000 11001 01000

ShiftRow hacia izquierda P3:

10101 10101
10000 00001
11001 00111

01000 00010

Obtenemos P3= 10101 00001 00111 00010

AddRoundKey de P3:

10101 00001 00111 00010 XOR
10000 11111 11111 11111

_____

C3=00101 11110 11000 11101

P4 = C3 XOR B4 = 00101 11001 10110 01100

ShiftRow hacia izquierda P4:

00101 00101
11001 10011
10110 11010
01100 00011

Obtenemos P4= 00101 10011 11010 00011

AddRoundKey de P4:

00101 10011 11010 00011 XOR
10000 11111 11111 11111

_____

C4=10101 01100 00101 11100

P5= C4 XOR B5 = 10101 01100 00101 00010

ShiftRow hacia izquierda P5:

10101 10101
01100 11000
00101 10100
00010 10000

Obtenemos P5= 10101 11000 10100 10000

AddRoundKey de P5:

10101 11000 10100 10000 XOR
10000 11111 11111 11111

_____

00101 00111 01011 01111

C5=00101 00111 01011 01111

Mensaje cifrado por Benito:

CBenito= 01101 11110 11000 00111 11001 11000 10100 00110 00101 11110 11000 11101 00101 00111 01011 01111

Y Alicia comprueba el mensaje y la firma:

Inverso de AddRoundKey para C1:

C1=01101 11110 11000 00111
XOR 10000 11111 11111 11111

11101 00001 00111 11000

Inverso de ShiftRow para C1:

11101	11101
00001	10000
00111	11001
11000	00011

Deshacemos el XOR con el IV:

11101 10000 11001 00011 XOR
01111 00101 11001 01110

10010 10101 00000 01101

P1=10010 10101 00000 01101

Inverso de AddRoundKey para C2:

C2 = 11001 11000 10100 00110
XOR 10000 11111 11111 11111

01001 00111 01011 11001

Inverso de ShiftRow para C2:

01001	01001
00111	10011
01011	11010
11001	00111

Deshacemos el XOR con el C1:

01001 10011 11010 00111 XOR
01101 11110 11000 00111

_____

00100 01101 00010 00000

P2 = 00100 01101 00010 00000

Inverso de AddRoundKey para C3:

C3= 00101 11110 11000 11101
XOR 10000 11111 11111 11111

_____

10101 00001 00111 00010

Inverso de ShiftRow para C3:

10101 10101
00001 10000
00111 11001
00010 01000

Deshacemos el XOR con el C2:

10101 10000 11001 01000 XOR
11001 11000 10100 00110

_____

01100 01000 01101 01110

P3=01100 01000 01101 01110

Inverso de AddRoundKey para C4:

C4=10101 01100 00101 11100 XOR
10000 11111 11111 11111

_____

00101 10011 11010 00011

Inverso de ShiftRow para C4:

00101 00101

10011 11001
11010 10110
00011 01100

Deshacemos el XOR con el C3:

00101 11001 10110 01100 XOR
00101 11110 11000 11101

_____

00000 00111 01110 10001
P4=00000 00111 01110 10001

Inverso de AddRoundKey para C5:

C5=00101 00111 01011 01111 XOR
10000 11111 11111 11111

_____

10101 11000 10100 10000

Inverso de ShiftRow para C5:

10101 10101
11000 01100
10100 00101
10000 00010

Deshacemos el XOR con el C4:

10101 01100 00101 00010 XOR
10101 01100 00101 11100

_____

00000 00000 00000 11110
P5=00000 00000 00000 11110

id=5 primeros bits
firma=10 últimos bits

Mensaje descifrado completo, M2:

10010||10101 00000 01101 00100 01101 00010 00000 01100 01000 01101 01110 00000 00111 01110 10001 00000 00000||00000 11110
los primeros 5 bits→ id = 18
últimos 10 bits → firma = 30
Texto en claro: VANENCAMINOAHORAA

Obtenemos que la firma de benito es 00000 11110 que es 30 en decimal, procederemos a verificarla:

H(M2) = 30
V(F) = 30^9 mod 85 = 30 → coincide con H(M2) = 30, por lo que la firma queda verificada

Se ha verificado la firma del mensaje recibido de Benito, por lo que la comunicación ha sido satisfactoria.