FUNDAMENTALS OF CYPTOGRAPHY ASSIGNMENT 01 (SEM 1, 2020)

NICHOLAS KLVANA-HOOPER, 19/04/2020

All possible keys for Affine

With format (a, b). Eligible keys:

(1,0), (1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (1,9), (1,10), (1,11), (1,12), (1,13), (1,14), (1,15), (1,16), (1,17), (1,18), (1,19), (1,20), (1,21), (1,22), (1,23), (1,24), (1,25), (1,26)
(2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (2,9), (2,10), (2,11), (2,12), (2,13), (2,14), (2,15), (2,16), (2,17), (2,18), (2,19), (2,20), (2,21), (2,22), (2,23), (2,24), (2,25), (2,26)
(4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (4,7), (4,8), (4,9), (4,10), (4,11), (4,12), (4,13), (4,14), (4,15), (4,16), (4,17), (4,18), (4,19), (4,20), (4,21), (4,22), (4,23), (4,24), (4,25), (4,26)
(5,0), (5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (5,7), (5,8), (5,9), (5,10), (5,11), (5,12), (5,13), (5,14), (5,15), (5,16), (5,17), (5,18), (5,19), (5,20), (5,21), (5,22), (5,23), (5,24), (5,25), (5,26)
(7,0), (7,1), (7,2), (7,3), (7,4), (7,5), (7,6), (7,7), (7,8), (7,9), (7,10), (7,11), (7,12), (7,13), (7,14), (7,15), (7,16), (7,17), (7,18), (7,19), (7,20), (7,21), (7,22), (7,23), (7,24), (7,25), (7,26)
(8,0), (8,1), (8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (8,8), (8,9), (8,10), (8,11), (8,12), (8,13), (8,14), (8,15), (8,16), (8,17), (8,18), (8,19), (8,20), (8,21), (8,22), (8,23), (8,24), (8,25), (8,26)
(10,0), (10,1), (10,2), (10,3), (10,4), (10,5), (10,6), (10,7), (10,8), (10,9), (10,10), (10,11), (10,12), (10,13), (10,14), (10,15), (10,16), (10,17), (10,18), (10,19), (10,20), (10,21), (10,22), (10,23), (10,24), (10,25), (10,26)
(11,0), (11,1), (11,2), (11,3), (11,4), (11,5), (11,6), (11,7), (11,8), (11,9), (11,10), (11,11), (11,12), (11,13), (11,14), (11,15), (11,16), (11,17), (11,18), (11,19), (11,20), (11,21), (11,22), (11,23), (11,24), (11,25), (11,26)
(13,0), (13,1), (13,2), (13,3), (13,4), (13,5), (13,6), (13,7), (13,8), (13,9), (13,10), (13,11), (13,12), (13,13), (13,14), (13,15), (13,16), (13,17), (13,18), (13,19), (13,20), (13,21), (13,22), (13,23), (13,24), (13,25), (13,26)
(14,0), (14,1), (14,2), (14,3), (14,4), (14,5), (14,6), (14,7), (14,8), (14,9), (14,10), (14,11), (14,12), (14,13), (14,14), (14,15), (14,16), (14,17), (14,18), (14,19), (14,20), (14,21), (14,22), (14,23), (14,24), (14,25), (14,26)
(16,0), (16,1), (16,2), (16,3), (16,4), (16,5), (16,6), (16,7), (16,8), (16,9), (16,10), (16,11), (16,12), (16,13), (16,14), (16,15), (16,16), (16,17), (16,18), (16,19), (16,20), (16,21), (16,22), (16,23), (16,24), (16,25), (16,26)
(17,0), (17,1), (17,2), (17,3), (17,4), (17,5), (17,6), (17,7), (17,8), (17,9), (17,10), (17,11), (17,12), (17,13), (17,14), (17,15), (17,16), (17,17), (17,18), (17,19), (17,20), (17,21), (17,22), (17,23), (17,24), (17,25), (17,26)
(19,0), (19,1), (19,2), (19,3), (19,4), (19,5), (19,6), (19,7), (19,8), (19,9), (19,10), (19,11), (19,12), (19,13), (19,14), (19,15), (19,16), (19,17), (19,18), (19,19), (19,20), (19,21), (19,22), (19,23), (19,24), (19,25), (19,26)
(20,0), (20,1), (20,2), (20,3), (20,4), (20,5), (20,6), (20,7), (20,8), (20,9), (20,10), (20,11), (20,12), (20,13), (20,14), (20,15), (20,16), (20,17), (20,18), (20,19), (20,20), (20,21), (20,22), (20,23), (20,24), (20,25), (20,26)
(22,0), (22,1), (22,2), (22,3), (22,4), (22,5), (22,6), (22,7), (22,8), (22,9), (22,10), (22,11), (22,12), (22,13), (22,14), (22,15), (22,16), (22,17), (22,18), (22,19), (22,20), (22,21), (22,22), (22,23), (22,24), (22,25), (22,26)
(23,0), (23,1), (23,2), (23,3), (23,4), (23,5), (23,6), (23,7), (23,8), (23,9), (23,10), (23,11), (23,12), (23,13), (23,14), (23,15), (23,16), (23,17), (23,18), (23,19), (23,20), (23,21), (23,22), (23,23), (23,24), (23,25), (23,26)
(25,0), (25,1), (25,2), (25,3), (25,4), (25,5), (25,6), (25,7), (25,8), (25,9), (25,10), (25,11), (25,12), (25,13), (25,14), (25,15), (25,16), (25,17), (25,18), (25,19), (25,20), (25,21), (25,22), (25,23), (25,24), (25,25), (25,26)
(26,0), (26,1), (26,2), (26,3), (26,4), (26,5), (26,6), (26,7), (26,8), (26,9), (26,10), (26,11), (26,12), (26,13), (26,14), (26,15), (26,16), (26,17), (26,18), (26,19), (26,20), (26,21), (26,22), (26,23), (26,24), (26,25), (26,26)

B is able to be any number between 0 and 26 inclusive as it has to be modded by 27. A can be any number between 1 and 26 inclusive that is not divisible by 3. This is because A has to be coprime with 27 and 3 is a common factor that occurs so therefore cannot be used. This gives us a total of 486 possible keys to use for the affine cipher.

Showing that affine code works

Original test file:

testfile-Affine.txt

```
Antoniwroteapaper,Inthispaperweconsidertheproblemofrobustfacerecognitionusingcolor
informationinthiscontextsparserepresentationbasedalgorithmsarethe
stateoftheartsolutionsforgrayfacialimageSproposedmodelthecontrolpar
ameterizationTechniquetOgetherwiththeconstrainttranscriptionmethodi
susedbytransformingtheproposedproblemintoasequenceofoptimalparameter
selectionproblemsFinallyapracticalexampleonbeersalesisusedtoshowtheeffectiveness
ofproposedmodelandwepresenttheoptimAladvertisingstrategiescorrespondingtodifferent
competitionsituationS.Wanquanpolishthispaper.
```

Encrypted then decrypted file: (using key of a=7, b=15)

output

```
Antoniwroteapaper,Inthispaperweconsidertheproblemofrobustfacerecognitionusingcolor
informationinthiscontextsparserepresentationbasedalgorithmsarethe
stateoftheartsolutionsforgrayfacialimageSproposedmodelthecontrolpar
ameterizationTechniquetOgetherwiththeconstrainttranscriptionmethodi
susedbytransformingtheproposedproblemintoasequenceofoptimalparameter
selectionproblemsFinallyapracticalexampleonbeersalesisusedtoshowtheeffectiveness
ofproposedmodelandwepresenttheoptimAladvertisingstrategiescorrespondingtodifferent
competitionsituationS.Wanquanpolishthispaper.
```

Both files are exactly the same, therefore the plaintext is recovered.

Letter distribution graph of test file

```
a: ***************************************
b: *******
c: *****************
d: **************
e: ********************************************************************
f: *************
g: **********
h: ******************
i: *******************************************
j:
k:
l: *******************
m: ****************
n: ***************************************
o: ****************************************************
p: ****************************
q: ***
r: ******************************************
s: ***************************************
t: *******************************************************
u: *********
v: **
w: *****
x: **
y: ***
z: *
```

Skipping non-letter symbols

To skip non-letter symbols the code has an if statement that only affects characters that have a value that represents a capital or lower-case alphabet character. In this way any other character will not be affected.

Mathematical Proof of DES

DES is heavily based on the idea that A XOR B = C which means it can also be decrypted by A XOR C = B.

DES Pseudocode

Encrypt/Decrypt Stage

- Plain text is imported as binary
- Permutate it with the IP array
- Break it into left and right strings
- Make left equal to the previous right, and right function xor'ed with previous left becomes the new right
- Get left and right switched
- Permutate the switched block with IP_I
- Change binary back to hex

R function Stage

- Permutate right side with E
- XOR the right side with the current key (1-16)
- Go through 8 S_Box permutations
    - Use first and last bit to determine what row
    - Use the rest of the bits to determine what column
    - Use these row and col to find S_BOX value to replace the part of the string
- Permutate the S_box-replaced-phrase with P

Switch Stage

- Take in right and left substrings
- Switch it so right substring appears first

Main issue of programming with DES

Main difficulty with programming the code was determining whether or not the encryption was correct as its not until you try decrypting that you can see if it works.

Encryption/Decryption with all 0's

My program encrypted and decrypted like normal.

BEFORE ENCRYPTION

For example as pointed out by researcher. For each set of fuzzy terms, $A \subseteq M$, $\prod_{m\in

A}m$ represents a conjunction of the fuzzy terms in A. For

instance,

let $A=\{m_{1,2},m_{2,1},m_{4,2}\}\subseteq M$, a new

fuzzy concept ``$m_{1,2}$ and $m_{2,1}$ and $m_{4,2}$" with the linguist

interpretation ``\emph{short sepal and wide sepal and narrow petal}"

can be represented as $\prod_{m\in

A}m=m_{1,2}m_{2,1}m_{4,2}$. Then the fuzzy rules can be represented as follows:

\bigskip

\textbf{Rule} R_1 : If x is

$m_{1,2}m_{2,1}m_{4,2}$, then x belongs to Class 1;

\textbf{Rule} R_2 : If x is $m_{2,1}m_{3,2}$, then x belongs

to Class 1;

\textbf{Rule} R_3 : If x is $m_{1,2}m_{4,2}$, then x belongs

to Class 1.

\bigskip

AFTER DES ENCRYPTION (12345678)

C47E17D29DF88A422BC3F874947521E5073B7D0BC0C5FB4E94F4E6379454C59D5035C07BAC7C4A8
8746C8D21936F9EC0441E12C2243F517BF0AD0553B7EA6C9E0E4E5FBAB60CFD1002BD5287EEE3E7
035305C8678EAACF3B70B0FDF488491A651E3530B4DEFBE412

4D8C84860E3B5FB3ACCBF52F0FC88D8AE6BDAD5F4635117FF2E23AA0D9D34C236B22BB05983732
C8593E95B2AA2F84D80559111CF7FA718D7951FA4815A18CFC

B4051FC4082774A142B0BAA7E220BA22

D2F6BBEB2A8A3EA779C094299044D670C17A32EDBA08AA659BA8CE76A7079FDF137239827C5BBA
328B9F97A3BC93762CCEDFAD124955A042

9B8B361BA951F4507C1883B20AECE7495B1459902A7A57FD993A2C7039EB7F99CBEF67425388E62
F277C8ACD1B5FA862B8E7E2567E7F058752D69D96937666F6A612BBD13D908ED0594764E1A2B2D
98E

51A8A8365F5E92184340E9714044E03FA0A0AD18F5C9652F78272E5AE82448D9FF6A9827BA8A1A9
CB0C68BD6FC5588D2B0E356F492CCEE10688378163AAA582FF4A0FACBCDFE3972

E61E557AA4F7EC079449C62B51E82798795FFF6E586B6436086DAD3364CD656750D802DAE99ADF9
3

D11294BC69E8BF45B3EEA15FFE333360556A226D0C051022C9D2F149ABDC3DE106C3B2410A18A88
A746FADF37B86B6022057543861B012BC0C64275679CA81BB76DA9F874DE6DBD308B1EDC990998
016594764E1A2B2D98E

594764E1A2B2D98E

395F75837AEB9D21594764E1A2B2D98E

594764E1A2B2D98E

C3C096411E02E8E05E81A9FEC3339ED41D563FF4DA07B03AD4D9062BFF0D4439594764E1A2B2D98
E

5B1459902A7A57FDA61F0E5980CD41A801212ACA41B7C7EA17340401CC3EEF4678CA649BE6EFB4B
9344E0E3816040A7E950ACA671B165A50

594764E1A2B2D98E

23FA42DF8CB57A16868C175B36D900590BBA1D530F4775C6D27842843A8B182381F55D093A0745
C0E660EBB6989FAA2F588F65E984B21612070F2014E345E5A99F318AFFEA9F3F11

CBBF397F97A15ECFE564237EEBB95A2C

594764E1A2B2D98E

23FA42DF8CB57A16868C175B36D9005914E73462F2754EC2D27842843A8B18235B1459902A7A57F
D34638436750DECB9588F65E984B21612070F2014E345E5A99F318AFFEA9F3F11

CBBF397F97A15ECF51C5A670020FA8A9

395F75837AEB9D21594764E1A2B2D98E

AFTER DES DECRYPTION

For example as pointed out by researcher. For each set of fuzzy terms, $A \subseteq M$, $\prod_{m\in

A}m$ represents a conjunction of the fuzzy terms in A. For

instance,

let $A=\{m_{1,2},m_{2,1},m_{4,2}\}\subseteq M$, a new

fuzzy concept ``$m_{1,2}$ and $m_{2,1}$ and $m_{4,2}$" with the linguist

interpretation ``\emph{short sepal and wide sepal and narrow petal}"

can be represented as $\prod_{m\in

A}m=m_{1,2}m_{2,1}m_{4,2}$. Then the fuzzy rules can be represented as follows:

\bigskip

\textbf{Rule} R_1 : If x is

$m_{1,2}m_{2,1}m_{4,2}$, then x belongs to Class 1;

\textbf{Rule} R_2 : If x is $m_{2,1}m_{3,2}$, then x belongs

to Class 1;

\textbf{Rule} R_3 : If x is $m_{1,2}m_{4,2}$, then x belongs

to Class 1.

\bigskip

Question 3

Threats DES can overcome

DES protects primarily against confidentiality threats. This is due to DES scrambling the data so that unauthorised users that have access to the files cannot understand it. Availability and integrity aren't particularly protected by DES. Availability is decreased with DES as its harder to get access to the data, and integrity is changed as the file has been modified when encrypted.

Source coding for DES Program

My DES program splits a line of characters up into 8 character blocks and then converts each of the characters in that block into hexadecimal making it a block of 16 hexadecimal bits. This hexadecimal block is given to the encryption which converts it into binary. This is also how key gen works.

Decryption works slightly different as it starts in hexadecimal and is converted binary to decrypt. This will then be passed to convert straight back to characters.

If you want to achieve the highest probability of error-correction in information transmission, tell us what you should do when you design a channel encoder.

Designing a channel encoder you can repeat a bit multiple times to ensure if one bit has an error the other two repeats may be fine and averaged. Therefore if you want to have the highest probability of error-correction you increase the number of redundancy bits you repeat.