Web Security and Symmetric Key Crypto Implementation

CSE 565 Fall 2015
Computer Security
Project 2

Computer Science and Engineering
University at Buffalo
Buffalo, NY-14214

Aniruddh Adkar
UB Person number: 50169486
aadkar@buffalo.edu

Aniket Thigale
UB Person number: 50168090
athigale@buffalo.edu

Saumitra Vaidya
UB Person number: 50168093
svaidya@buffalo.edu

Tushar Kulkarni
UB Person number: 50168831
tkulkarn@buffalo.edu

Pranav Jain
UB Person number: 50169659
pranavja@buffalo.edu

Part I

Task 1

Q. Do some research on these items and then answer the following questions.
1. Pick out 4 terms that belong to attack techniques and explain each of them.

Attack techniques
Browser hijacking is a form of online fraud that scammers use to modify web **browser**'s settings without a user's permission, to inject unwanted advertising into the user's **browser**. A **browser hijacker** may replace the existing home page, error page, or search page with its own.
Denial-of-service (**DoS**) is an attack that intends to make resource or service unavailable, which is generally done by sending more requests to the target than it can handle resulting in service interruption.
Privilege escalation: Attacker takes the advantage of vulnerability or a design flaw in Software or Operation System to move up the privilege ladder by granting himself more privileges that are not accessible to the user.
Sniffer is a program that catches the packets sent and received over the network legitimately or illegitimately, for particular types of networks, an **Ethernet sniffer** or **wireless sniffer**) is a computer program or piece of computer hardware that can intercept and log traffic that passes over a digital network or part of a network

2. Pick out 4 terms that belong to defense techniques and explain each of them

Defense techniques
Input Validation is the outer defensive perimeter for your web application which restricts the input format to avoid attacks such as SQL injection, buffer overflow etc.
Firewall is a network security system that is implemented to monitors and controls the incoming and outgoing network traffic based on predetermined security rules.
Intrusion detection system monitors network traffic and look out for suspicious activity that might compromise security of the system, it acts upon it by alerting the system or network administrator or blocking such activities.
Checksum is a method to ensure the integrity of the file once it has been transferred, it is calculated using hash function which user verifies by comparing with the desired Checksum value.

Task 2

Injection attacks

Q. List any four malicious objectives that an attacker can achieve using SQL injection attack?

1. Read (and misuse) sensitive data from the database.
2. Modify database data – insert/update/delete.
3. Execute administrative operations on the database like shut down the DBMS.
4. Issue commands to the operating system.

Source: https://blog.sucuri.net/2014/10/website-attacks-sql-injection-and-the-threat-they-present.html

Q. Briefly answer the following questions about SQL injection.
Q Can you see a full list of all the employees in the database? Explain your answer and write down the SQL queries involved, if any.

Yes, we can see a full list of all the employees in the database using SQL injection.
The original query is:
SELECT firstname FROM employees where lastname= '$LASTNAME';
To find a full list of employees we can insert $LASTNAME as:

lastname' OR 'X'='X
Final Query becomes:

SELECT firstname FROM employees where lastname= 'lastname' OR 'X'='X';

X=X is always true and hence it will match all records. The application will then return a list of all employees.

Q Can you add a new employee to the database? Explain your answer and write down the SQL queries involved, if any.

Yes, we can add a new employee to the database.
We can pass $LASTNAME as to insert a new employee record:

x';INSERT INTO employees VALUES ('firstName','lastName');--

Query becomes:

SELECT firstname FROM employees where lastname= 'x';INSERT INTO employees VALUES ('firstName','lastName');--';

However, this method might fail due to many reasons. For eg. if there are other fields in the employees table or if there are other associated tables, so that adding to one table alone is not sufficient.
Source: http://www.unixwiz.net/techtips/sql-injection.html

Q. Briefly describe and demonstrate the various prevention measures that can be taken against SQL injection attacks (no implementations necessary).

Defense Option 1: Prepared Statements (Parameterized Queries)
We can use Prepared Statements which will replace qualifiers in the pre-rendered query statement. Thus code and data separation is achieved, regardless of what user input is supplied. They ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.

Defense Option 2: Stored Procedures
The SQL code for a stored procedure is defined and stored in the database itself, and then called from the application. The user input data are then passed as parameters which acts similar to the Prepared Statement. They require the developer to just build SQL statements with parameters which are automatically parameterized.

Defense Option 3: Escaping All User Supplied Input
It is to be used only to retrofit the legacy code. We escape user input before putting it in a query. The DBMS will not confuse the escaped user input with SQL code written by the developer, thus avoiding any possible SQL injection vulnerabilities.
Defense Option 4: Least Privilege
We can minimize the privileges assigned to every database account in our environment. Assign only the access rights that the application account requires rather than assigning it DBA or admin type access rights. We can also create a view to that provides restricted access to table data so that the application can access that part.
Source: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

WebGoat

Q. Because WebGoat is a vulnerable web application, it will make your system insecure while you work on this project. How can an attacker attack your system if you are using WebGoat without taking any precautions?

Attacker can search for open ports on the network and can login to WebGoat by using default credentials.

Attacker can use code execution lessons in WebGoat to perform code injections and exploit user's system.

Q. How do you fix the above-mentioned problem? (Hint: There are multiple ways to fix this problem. You get points for mentioning the solution that does not hamper other system functionalities.)

Rule based filtering on system's firewall can block all inbound and outbound traffic for specific ports as well as for applications. It will allow us to block access to WebGoat application from outside of local machine and hence secure the system.

Q. What is the principle of least privilege? How can you manage access privileges for using WebGoat?
Least Privilege is best practice to assign user with least possible privileges to enforce access control and authorizations. Access Privileges can be managed using maintaining Access control matrix which involves user id and allowed actions.

Task 3 Part A

Run WebGoat and execute all the lessons under 'Injection flaws' (leave out the ones that require development version of WebGoat) and write a summary of your experience in one paragraph.

We performed variety of SQL injection attacks such as String SQL injection, Parameterized Query, numeric injections etc. It is highly insecure to pass user entered data directly to SQL query engine without checking and validating as can cause catastrophic issues and can breach the security. Injecting SQL queries in intended input box can cause user to get unauthorized access above security clearance and brings the whole integrity of system into question. User can gain read-write access and bypass established Access Control mechanisms and modify/update/delete data. Risks because of SQL injections can be mitigated by use of Prepared statements, stored procedures proper Access Control lists and least possible privileges along with pre validation of input data submitted by user with the help of limited whitelisting.

Task 3 Part B

From the lessons you learned (Task 3a), implement and demonstrate 1) string SQL injection and 2) Cross-Site Scripting attack (XSS attack). For each of these attacks, summarize your understanding including:

1. **String SQL Injection**

SQL queries

For Lab session in String SQL injection we intercepted login request with Webscarab and modified its password parameter to inject following SQL query.

smith' OR '1' = '1'

For modifying existing records, we can use queries like :

any'; UPDATE salaries SET salary=929 WHERE userid='jsmith

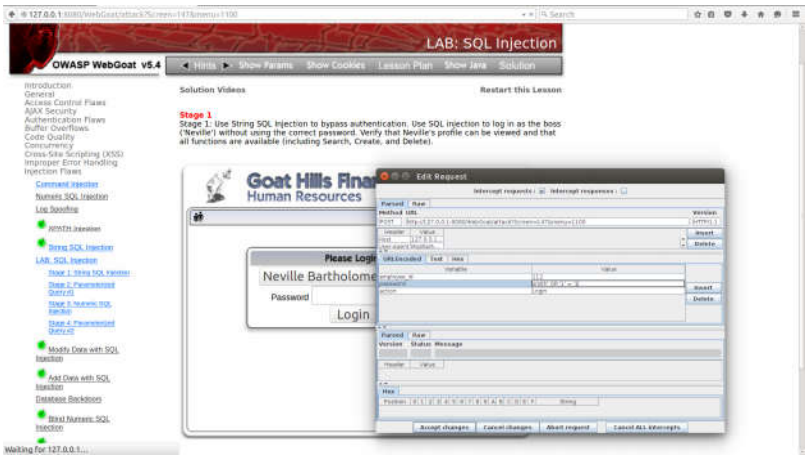
Also following queries can be used for inserting new records into table :

'insert into salaries values('ani1',1010)--'

Fixing the vulnerability

WebGoat fixed vulnerabilities by using Parameterized query and prepared statements.

Screenshots



2. Cross-Site Scripting

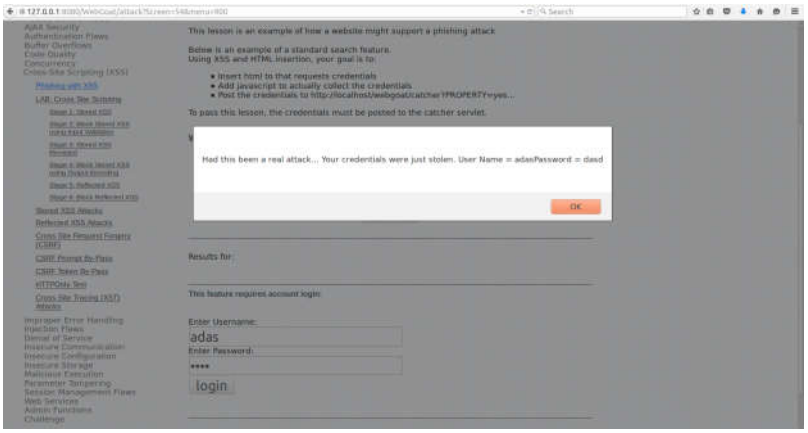
XSS Scripts

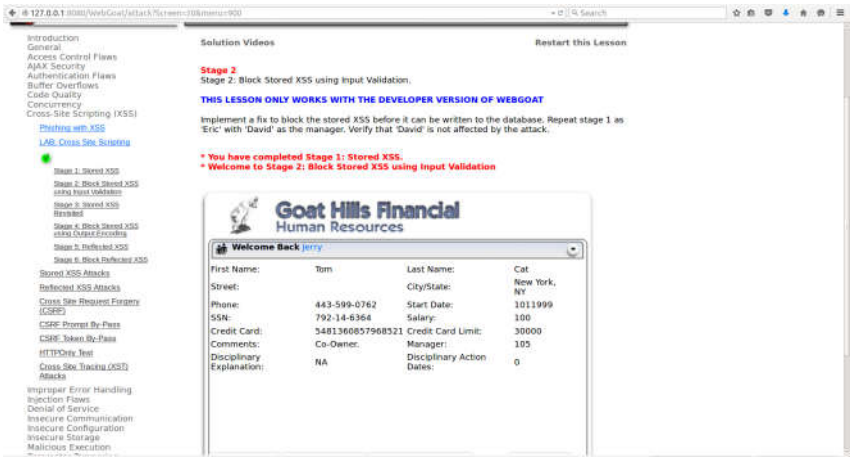
<script>alert("XSS works");</script>
<script>alert("system exploited ");</script>.

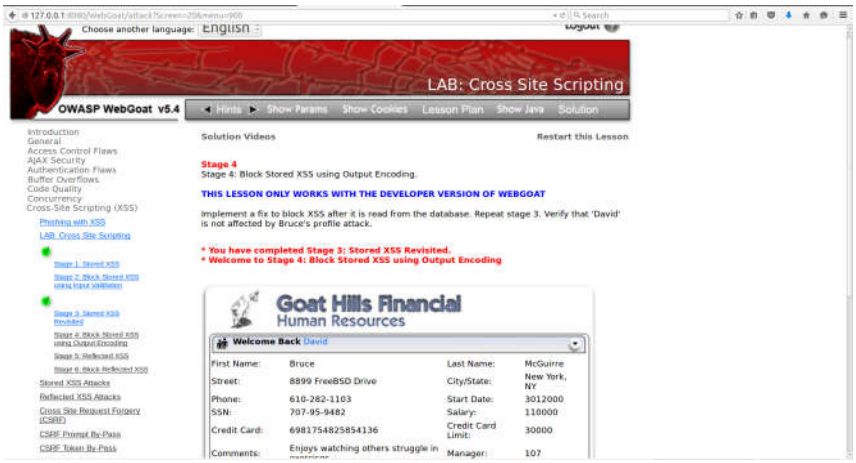
Fixing the vulnerability

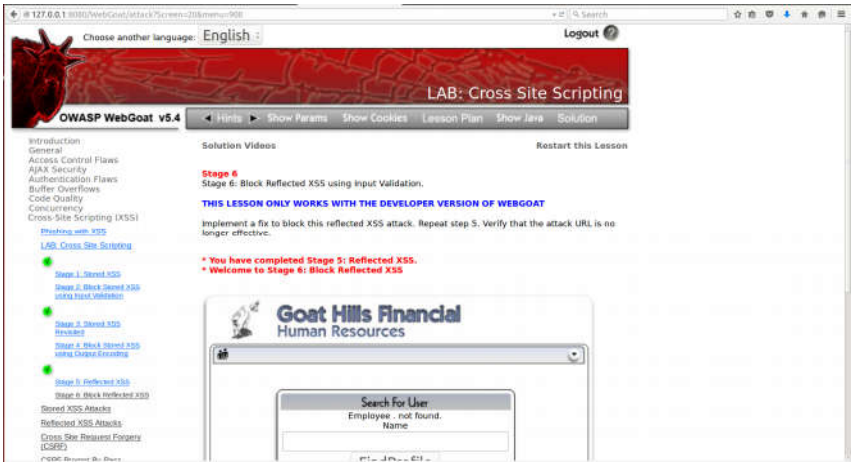
UpdateProfile's parseEmployeeProfile method is modified to add escape sequence and regex which will whitelist the user input. Also user input should be passed through filter which will identify any suspicious token.

Screenshots




Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass
HTTPOnly Test
Cross Site Tracing (XST) Attacks
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Solution Videos
Restart this Lesson
Stage 2
Stage 2: Block Stored XSS using Input Validation.
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT
Implement a fix to block the stored XSS before it can be written to the database. Repeat stage 1 as 'Eric' with 'David' as the manager. Verify that 'David' is not affected by the attack.
* You have completed Stage 1: Stored XSS.
* Welcome to Stage 2: Block Stored XSS using Input Validation
Goat Hills Financial
Human Resources
Welcome Back Jerry
First Name: Tom
Last Name: Cat
Street:
City/State: New York, NY
Phone: 443-599-0762
Start Date: 1011999
SSN: 792-14-6364
Salary: 100
Credit Card: 5481360857968521
Credit Card Limit: 30000
Comments: Co-Owner.
Manager: 105
Disciplinary Explanation: NA
Disciplinary Action Dates: 0


Choose another language: English
LAB: Cross Site Scripting
OWASP WebGoat v5.4
Hints
Show Params
Show Cookies
Lesson Plan
Show Java
Solution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass
Solution Videos
Restart this Lesson
Stage 4
Stage 4: Block Stored XSS using Output Encoding.
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT
Implement a fix to block XSS after it is read from the database. Repeat stage 3. Verify that 'David' is not affected by Bruce's profile attack.
* You have completed Stage 3: Stored XSS Revisited.
* Welcome to Stage 4: Block Stored XSS using Output Encoding
Goat Hills Financial
Human Resources
Welcome Back David
First Name: Bruce
Last Name: McGuirre
Street: 8899 FreeBSD Drive
City/State: New York, NY
Phone: 610-282-1103
Start Date: 3012000
SSN: 707-95-9482
Salary: 110000
Credit Card: 6981754825854136
Credit Card Limit: 30000
Comments: Enjoys watching others struggle in
Manager: 107


Choose another language: English
Logout
LAB: Cross Site Scripting
OWASP WebGoat v5.4
Hints
Show Params
Show Cookies
Lesson Plan
Show Java
Solution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
Solution Videos
Restart this Lesson
Stage 6
Stage 6: Block Reflected XSS using Input Validation.
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT
Implement a fix to block this reflected XSS attack. Repeat step 5. Verify that the attack URL is no longer effective.
* You have completed Stage 5: Reflected XSS.
* Welcome to Stage 6: Block Reflected XSS
Goat Hills Financial
Human Resources
Search For User
Employee . not found.
Name

Part II
Part A: Block Cipher Internals

The received AES triple is in the form **Key Plain Cipher**. Our triple is:

Key: 44ef34ac419c2a09084b46b64f38e457
Plain: b210287fcab7f612fcd2d8601a25a1b0
Cipher: a6a1bda7f3ecc102ae5df1fdc4797bb4

Proceeding as per instructions given on the link
http://williamstallings.com/Crypto/AESCalc/AESlab.html, the following has been asked in this part:

Q. Encrypt the plaintext using the key given in your triple, with tracing of the round values. Note how the value of the **state** (result of each round) changes from round to round. What is the value of your **state** after round 4?

The value of state after round 4 is 5f61d081fbb9d5eb3d84c83d4a6738d5. The screenshot is given below

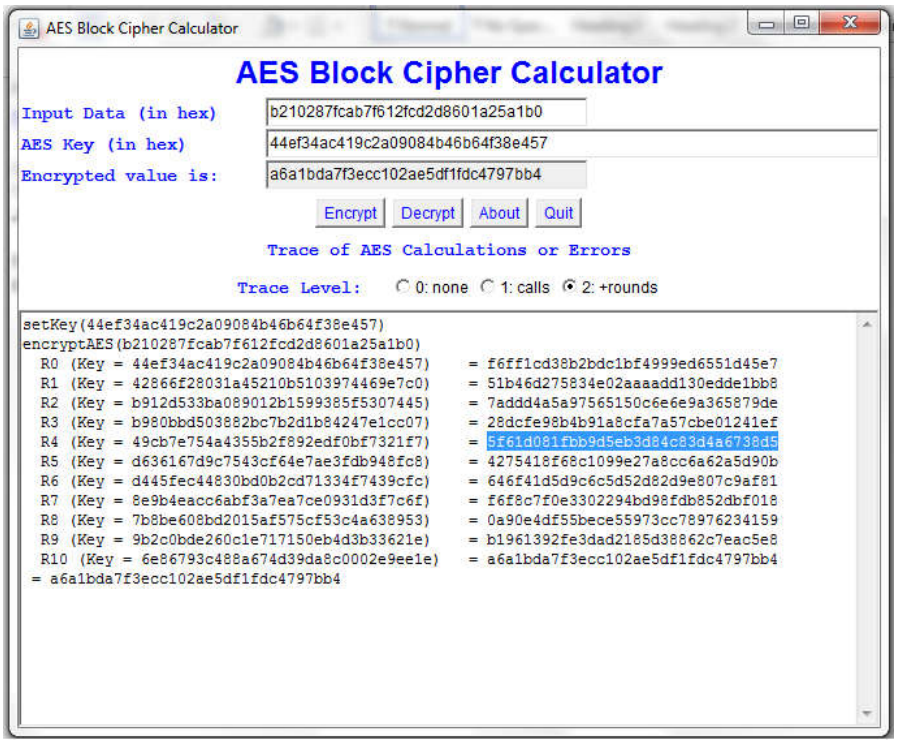


Figure 1 AES Implementation on AES Calculator

Q. Change AES bit 12 of the PLAINTEXT in your triple (i.e. change the 0 to 1, or 1 to 0 as appropriate), assuming AES bit numbering from left (MSB) bit 0 to right (LSB) bit 127. Encrypt this new plaintext value using the AES Calculator. Using the trace output, after each of the first four rounds list in a table how many bits of state differ from the corresponding values in part i (nb. you will have to convert between hexadecimal & binary and compare the relevant bits to do this).

Given plaintext is b210287fcab7f612fcd2d8601a25a1b0.

The bit 12 in the plaintext will be the first bit in the hex digit 0, which is fourth from the left. Therefore, substituting 1 at the 12^{th} bit instead of zero, the new plaintext is:

b218287fcab7f612fcd2d8601a25a1b0

Using the same key, the cipher text received is:
a1be60c59d201f7a28a8a94799452995

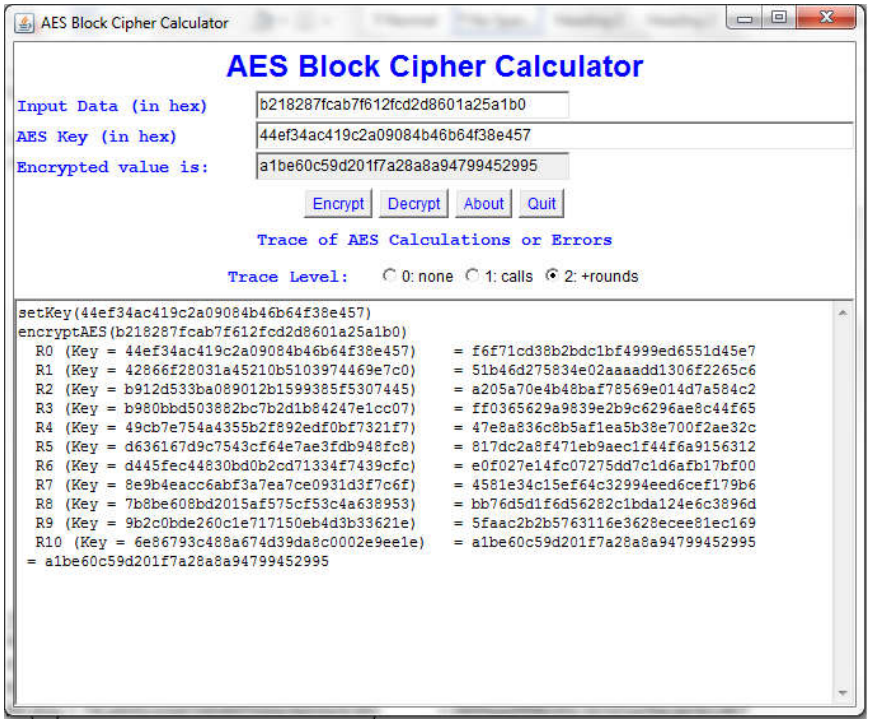


Figure 2 Implementation with new plaintext

The original output:

setKey(44ef34ac419c2a09084b46b64f38e457)
encryptAES(b210287fcab7f612fcd2d8601a25a1b0)
R0 (Key = 44ef34ac419c2a09084b46b64f38e457) = f6ff1cd38b2bdc1bf4999ed6551d45e7
R1 (Key = 42866f28031a45210b5103974469e7c0) = 51b46d275834e02aaaadd130edde1bb8
R2 (Key = b912d533ba089012b1599385f5307445) = 7addd4a5a97565150c6e6e9a365879de
R3 (Key = b980bbd503882bc7b2d1b84247e1cc07) = 28dcfe98b4b91a8cfa7a57cbe01241ef
R4 (Key = 49cb7e754a4355b2f892edf0bf7321f7) = 5f61d081fbb9d5eb3d84c83d4a6738d5
R5 (Key = d636167d9c7543cf64e7ae3fdb948fc8) = 4275418f68c1099e27a8cc6a62a5d90b
R6 (Key = d445fec44830bd0b2cd71334f7439cfc) = 646f41d5d9c6c5d52d82d9e807c9af81
R7 (Key = 8e9b4eacc6abf3a7ea7ce0931d3f7c6f) = f6f8c7f0e3302294bd98fdb852dbf018
R8 (Key = 7b8be608bd2015af575cf53c4a638953) = 0a90e4df55bece55973cc78976234159
R9 (Key = 9b2c0bde260c1e717150eb4d3b33621e) = b1961392fe3dad2185d38862c7eac5e8
R10 (Key = 6e86793c488a674d39da8c0002e9ee1e) = a6a1bda7f3ecc102ae5df1fdc4797bb4
= a6a1bda7f3ecc102ae5df1fdc4797bb4

The output after changing the 12th bit:

setKey(44ef34ac419c2a09084b46b64f38e457)
encryptAES(b218287fcab7f612fcd2d8601a25a1b0)
R0 (Key = 44ef34ac419c2a09084b46b64f38e457) = f6f71cd38b2bdc1bf4999ed6551d45e7
R1 (Key = 42866f28031a45210b5103974469e7c0) = 51b46d275834e02aaaadd1306f2265c6
R2 (Key = b912d533ba089012b1599385f5307445) = a205a70e4b48baf78569e014d7a584c2
R3 (Key = b980bbd503882bc7b2d1b84247e1cc07) = ff0365629a9839e2b9c6296ae8c44f65
R4 (Key = 49cb7e754a4355b2f892edf0bf7321f7) = 47e8a836c8b5af1ea5b38e700f2ae32c
R5 (Key = d636167d9c7543cf64e7ae3fdb948fc8) = 817dc2a8f471eb9aec1f44f6a9156312
R6 (Key = d445fec44830bd0b2cd71334f7439cfc) = e0f027e14fc07275dd7c1d6afb17bf00
R7 (Key = 8e9b4eacc6abf3a7ea7ce0931d3f7c6f) = 4581e34c15ef64c32994eed6cef179b6
R8 (Key = 7b8be608bd2015af575cf53c4a638953) = bb76d5d1f6d56282c1bda124e6c3896d

R9 (Key = 9b2c0bde260c1e717150eb4d3b33621e) = 5faac2b2b5763116e3628ecee81ec169
R10 (Key = 6e86793c488a674d39da8c0002e9ee1e) = a1be60c59d201f7a28a8a94799452995

Table 1 Table showing change in bits in binary

Round number	In original computation	After changing bit 12	Number of different bits
0	f6ff1cd38b2bdc1bf4999ed6551d45e7	f6f71cd38b2bdc1bf4999ed6551d45e7	1
1	51b46d275834e02aaaadd130edde1bb8	51b46d275834e02aaaadd1306f2265c6	20
2	7addd4a5a97565150c6e6e9a365879de	a205a70e4b48baf78569e014d7a584c2	73
3	28dcfe98b4b91a8cfa7a57cbe01241ef	ff0365629a9839e2b9c6296ae8c44f65	67
4	5f61d081fbb9d5eb3d84c83d4a6738d5	47e8a836c8b5af1ea5b38e700f2ae32c	66

Q. Describe which characteristic(s) of a good block cipher design have been illustrated by this exercise, and how they are demonstrated.

In the above question, by changing just one bit, in the subsequent rounds, the number of bits changed with the progress of different rounds increases. This is called Avalanche effect – small change in the plaintext causes a huge change in the cipher text. It also displays the property of diffusion. Diffusion is when relationship between plaintext and cipher is complex.

Part B: Block Cipher Round

Source: Tables from Stallings 6e Ch-5

The given key is arranged in 4x4 matrix with each 4-byte vertical block forming a word of the four-word key (16 bytes).

44	EF	34	AC
41	9C	2A	09
08	4B	46	B6
4F	38	E4	57

Now, w0= 44 EF 34 AC

w1= 41 9C 2A 09

w2= 08 4B 46 B6

w3= 4F 38 E4 57

For w4-w7: w3 is passed through the g function:

1. **RotWord** performs a 1 byte left circular shift to get **38 E4 57 4F**

2. **SubWord** performs byte substitution using Table 5.2a from the book to get **07 69 5B 84**

3. Result is XORed with Round constant **Rcon[0]**. Leftmost byte of Rcon[0] is 01. Rightmost

bytes are always zero. The result is **06 69 5B 84** This is the output of g function.

To get w4, result of g function is XORed with w0.

06 69 5B 84

XOR 44 EF 34 AC

w4=> **42 86 6F 28**

w5= w1 XOR w4, w6= w2 XOR w5, w7= w3 XOR w6. Hence, final values are

w4= 42 86 6F 28

w5= 03 1A 45 21

w6= 0B 51 03 97

w7= 44 69 E7 C0

This is key for Round 1.

Now the state matrix for the input plaintext is:

B2	10	28	7F
CA	B7	F6	12
FC	D2	D8	60
1A	25	A1	B0

Key for Round 0 is:

44	EF	34	AC
41	9C	2A	09
08	4B	46	B6
4F	38	E4	57

AddRoundKey Output of Round 0 is

F6	FF	1C	D3
8B	2B	DC	1B
F4	99	9E	D6
55	1D	45	E7

SubBytes: Referring Table 5.2a from the textbook, the following is obtained

42	16	9C	66
3D	F1	86	AF
BF	EE	0B	F6
FC	A4	6E	94

ShiftRows: 1st Row is not altered; 2nd , 3rd and 4th Rows are shifted by 1,2 and 3 bytes respectively.

42	16	9C	66
F1	86	AF	3D
0B	F6	BF	EE
94	FC	A4	6E

MixColumns: Static Matrix for Mix Column is

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

02·42 XOR 03·F1 XOR 01·0B XOR 01·94
In GF(2^8) addition is a bitwise XOR operation. And Multiplication can be performed as per the rule
Multiplication of 02 and 42 can be done as :
0000 0010 XOR 0100 0010 = 1000 0100

03 . F1 can be done as:
F1 XOR (F1 XOR 02)
1111 0001 XOR (1110 0010 XOR 0001 1011) = 0000 1000

Similarly ,
01.0B = 0B and 01.94 = 94

Hence 02·42 XOR 03·F1 XOR 01·0B XOR 01·94 = $0001\ 0011_{10}$ = $(13)_{16}$

The answer of multiplication is :

13	32	02	0F
5B	2E	A5	0B
A1	FC	D2	A7
A9	B7	FC	78

AddRoundKey: XORing the key for Round1 with the above, the following is yielded which is also the

result for Round1:

51	B4	6D	27
58	34	E0	2A
AA	AD	D1	30
ED	DE	1B	B8

51B46D275834E02AAAADD130EDDE1BB8 is the result.

Part C: Block Cipher Modes of Use

a) Implementation of CBC Mode

Key: *pranavjainabcdef*

HEX: K = 7072616E61766A61696E616263646566

Message:

This is a sample test message for Pranav!

HEX:
5468697320697320612073616D706C652074657374206D6573736167652066F72205072616E617621

The last block size is less than 16 bytes. We choose null padding to make the block size to 16 bytes. After decrypting, we trim all the null characters starting from the LSB until we find a non-null character. Since we are encrypting ASCII text, this method won't be a problem.

(*Source: http://www.di-mgt.com.au/cryptopad.html#exampleaes*)

Padding the last block with nulls (zeros) to make the last block size 32 bytes, we get:

HEX: 546869732069732061207361 6D706C65 2074657374206D65737361676520666F 72205072616E6176210000000000000000

P1 = 546869732069732061207361 6D706C65

P2 = 2074657374206D65737361676520666F

P3 = 72205072616E6176210000000000000000

C_0 = IV = 00000000000000000000000000000000

Encryption:

$P_1 \oplus C_0$ = 546869732069732061207361 6D706C65 $\oplus$ 00000000000000000000000000000000

= 546869732069732061207361 6D706C65

$C_1 = E(P_1 \oplus C_0) = E(546869732069732061207361 6D706C65)$

C_1 = AFBE12B77F97FA367197E3648DF4FC3A

$C_1 \oplus P_2$ = AFBE12B77F97FA367197E3648DF4FC3A $\oplus$ 2074657374206D65737361676520666F

= 8FCA77C40BB7975302E48203E8D49A55

$C_2 = E(C_1 \oplus P_2) = E(8FCA77C40BB7975302E48203E8D49A55)$

C_2 = A48F6D3351827F25E67EE6062BED3787

$C_2 \oplus P_3$ = A48F6D3351827F25E67EE6062BED3787 $\oplus$ 72205072616E6176210000000000000000

= D6AF3D4130EC1E53C77EE6062BED3787

$C_3 = E(C_2 \oplus P_3) = E(D6AF3D4130EC1E53C77EE6062BED3787)$

C_3 = 8A42EDFCB021F58B7AF3BBEBB90F8EDF

Decryption:

$D(C_1) = D(AFBE12B77F97FA367197E3648DF4FC3A) = 546869732069732061207361 6D706C65$

$P_1 = D(C1) \oplus C_0$

$P_1 = 546869732069732061207361 6D706C65 \oplus 00000000000000000000000000000000$

P_1 = 546869732069732061207361 6D706C65

$D(C_2) = D(A48F6D3351827F25E67EE6062BED3787) = 8FCA77C40BB7975302E48203E8D49A55$

$P_2 = D(C_2) \oplus C_1$

$P_2 = 8FCA77C40BB7975302E48203E8D49A55 \oplus AFBE12B77F97FA367197E3648DF4FC3A$

P_2 = 2074657374206D657373616767 6520666F

$D(C_3) = D(8A42EDFCB021F58B7AF3BBEBB90F8EDF) = D6AF3D4130EC1E53C77EE6062BED3787$

$P_3 = D(C3) \oplus C_2$

$P_3 = D6AF3D4130EC1E53C77EE6062BED3787 \oplus A48F6D3351827F25E67EE6062BED3787$

P_3 = 72205072616E617621 00000000000000000

b) Implementation of CFB-128 Mode

Key and Message selected for this mode is same as the one selected for CFB mode.

Therefore we have,

Key:

pranavjainabcdef

hex: K = 7072616e61766a61696e616263646566

Message:

This is a sample test message for Pranav!

HEX:
546869732069732061207361 6D706C652074657374206D657373616767 6520666F72205072616E617621

P1 = 546869732069732061207361 6D706C65

P2 = 2074657374206D657373616767 6520666F

P3 = 72205072616E617621

C_0 = IV = 00000000000000000000000000000000

Encryption:

For CFB mode, Encryption is given by:

$C_I = P_I \oplus E_K (C_{I-1})$

$C_1 = P_1 \oplus E_K(C_0)$

P_1 = 546869732069732061207361 6D706C65

$E_K(C_0) = E_K(00000000000000000000000000000000) = 9FFC21FA37510B3F46CB923BA703AF62$

$C_1 = P_1 \oplus E_K(C_0) = 546869732069732061207361 6D706C65 \oplus 9FFC21FA37510B3F46CB923BA703AF62$

C_1 = CB9448891738781F27EBE15ACA73C307

$C_2 = P_2 \oplus E_K(C_1)$

P2 = 2074657374206D65737361676520666F

$E_K(C1) = E_K(CB9448891738781F27EBE15ACA73C307) = 727E7197A41CCD2934B00B2F36D8E1DE$

$C_2 = P_2 \oplus E_K(C_1) = 2074657374206D65737361676520666F \oplus 727E7197A41CCD2934B00B2F36D8E1DE$

C_2 = 520A14E4D03CA04C47C36A4853F887B1

$C_3 = P_3 \oplus E_K(C_2)$

P3 = 72205072616E617621

$E_K(C_2) = E_K(520A14E4D03CA04C47C36A4853F887B1) = 77E0DC0238B19BAD7E1BD74DABE4EA85$

$C_3 = P_2 \oplus E_K(C_1) = 72205072616E617621 \oplus 77E0DC0238B19BAD7E1BD74DABE4EA85$

C_3 = 77E0DC0238B19BDF5E4BA52CC5859CA4

Decryption:

For CFB mode, decryption is given by:

$P_I = C_I \oplus E_K (C_{I-1})$

$P_1 = C_1 \oplus E_K(C_0)$

C_1 = CB9448891738781F27EBE15ACA73C307

$E_K(C_0) = E_K(00000000000000000000000000000000) = 9FFC21FA37510B3F46CB923BA703AF62$

$P_1 = C_1 \oplus E_K(C_0) = CB9448891738781F27EBE15ACA73C307 \oplus 9FFC21FA37510B3F46CB923BA703AF62$

P_1 = 546869732069732061207361 6D706C65

$P_2 = C_2 \oplus E_K(C_1)$

C_2 = 520A14E4D03CA04C47C36A4853F887B1

$E_K(C1) = E_K(CB9448891738781F27EBE15ACA73C307)$ = 727E7197A41CCD2934B00B2F36D8E1DE

$P_2 = C_2 \oplus E_K(C_1)$ = 520A14E4D03CA04C47C36A4853F887B1⊕727E7197A41CCD2934B00B2F36D8E1DE

P_2 = 2074657374206D657373616761652066 6F

$P_3 = C_3 \oplus E_K(C_2)$

C_3 = 77E0DC0238B19BDF5E4BA52CC5859CA4

$E_K(C_2) = E_K(520A14E4D03CA04C47C36A4853F887B1)$ = 77E0DC0238B19BAD7E1BD74DABE4EA85

$P_3 = C_3 E_K(C_2)$ = 77E0DC0238B19BDF5E4BA52CC5859CA4 ⊕77E0DC0238B19BAD7E1BD74DABE4EA85

P_3 = 72205072616E617621

CFB is easy to implement than CBC because following reasons:

1. In CFB, we need to perform only Encryption operations, whereas in CBC, we perform Encryption as well as decryption.
2. CFB does not require padding of the last block of plain text. This makes it easier and to decrypt plain text as it does not have the overhead of trimming the padded bits after decryption.

Applications of CBC and CFB:

CBC should be used if we want to encrypt a file/block of data.

CFB mode should be used when we have continuous stream of messages to be encrypted

Task Distribution

WebGoat (SQL Injections, Cross-Site Scripting)
Aniruddh Adkar

AES Block Cipher Round
Aniket Thigale, Saumitra Vaidya

Injection Attacks
Aniket Thigale, Saumitra Vaidya, Pranav Jain, Aniruddh Adkar

Block Cipher Internals, AES Block Cipher, Modes of Use (CBC, CFB-128)
Tushar Kulkarni
Pranav Jain

Attack & Defence techniques
Saumitra Vaidya