INSTITUT TEKNOLOGI DEL

MATERI PRAKTIKUM

Keamanan Perangkat Lunak
SEMESTER GASAL TAHUN AJAR 2024/2025

Session Date	**:**	23 Oktober 2024
Semester	**:**	V
Courses	**:**	Software Security / Keamanan Perangkat Lunak
Week/Session	**:**	04/03
Key Topics	**:**	**Teori bilangan, f*inite fields* dan Advanced Encryption Standard Algorithm**
Activity	**:**	Praktikum
Duration	**:**	110 menit
Delivery	**:**	Laporan Tugas *softcopy*
Deadline of delivery	**:**	-
Place of delivery	**:**	e-Course
Goal	**:**	Mahasiswa memahami konsep dasar dari teori bilangan, *finite fields,* dan algoritma AES

PENUGASAN:

Sebelum bekerja, setiap mahasiswa harus membaca instruksi di bawah ini.

Sangat disarankan bagi anda untuk:

1. ***Membaca soal-soal yang diberikan secara.***
2. ***Mencari sumber-sumber lain seperti buku, artikel, bahkan video untuk memperkaya wawasan dan meningkatkan pemahaman anda.***
3. ***Jika anda merasa ada hal yang belum dipahami, silakan untuk berkonsultasi pada TA.***
4. ***Dengan demikian diharapkan anda mampu mengikuti materi kuliah dan praktikum sebaik mungkin.***
5. ***Anda diharapkan membaca buku yang diberikan, untuk topik kali ini diambil dari Part One : Symetric Chiper Chapter 4 dan 5.***

Selamat Belajar & Good Luck!

Review Questions

1. Briefly define a group and give the example.
2. Briefly define a ring and give the example.
3. Briefly define a field and give the example.
4. List three classes of polynomial arithmetic.

Problem

1. Determine gcd(1970, 1066) using Euclidean Algorithm.
2. Give the addition and multiplication table for GF(2^2) with $m(x) = x^2 + x + 1$.
3. Given $f(x) = x^3 + x + 1$ and $g(x) = x^6 + x^4 + x^3 + 1$

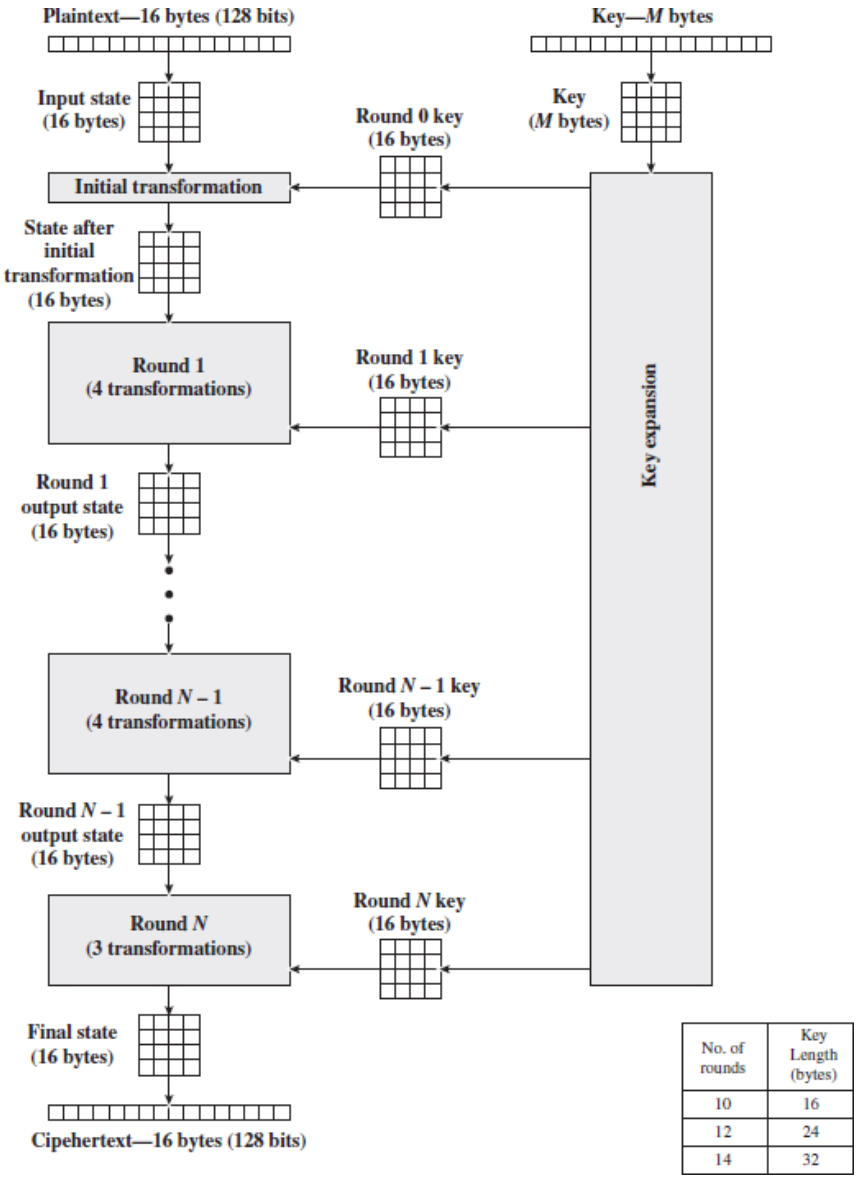
   Calculate the multiplication of f(x) and g(x) in GF(2^8)
4. Determine the gcd of the following pairs of polynomials.
   a. $x3 + x + 1$ and $x2 + x + 1$ over GF(2)
   b. $x3 - x + 1$ and $x2 + 1$ over GF(3)

Case

The plaintext is a hexadecimal palindrome. The plaintext and key

Plaintext: **`0123456789abcdeffedcba9876543210`**
Key: **`0f1571c947d9e8590cb7add6af7f6798`**

apply 1st round (**round 0**) of AES encryption.



No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 1 AES Encryption Process

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

Figure 2 AES S-box

Computer programming (Java)

Try the codes below in your IDE (e.g. Netbeans), run the program, and analyse the program

1. Program #1

```
package aes;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.Cipher;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.security.InvalidAlgorithmParameterException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import sun.misc.BASE64Encoder;

/**
 *
 * @author itdel
 */

public class AES {
public static void main(String[] args) {

String strDataToEncrypt = new String();
String strCipherText = new String();
String strDecryptedText = new String();
try{
/**
 * Step 1. Generate an AES key using KeyGenerator
 * Initialize the keysize to 128
 *
 */
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey secretKey = keyGen.generateKey();

/**
 * Step2. Create a Cipher by specifying the following
parameters
 * a. Algorithm name - here it is AES
 */
Cipher aesCipher = Cipher.getInstance("AES");

/**
 * Step 3. Initialize the Cipher for Encryption
 */
aesCipher.init(Cipher.ENCRYPT_MODE,secretKey);

/**
 * Step 4. Encrypt the Data
 * 1. Declare / Initialize the Data. Here the data is of type String
 * 2. Convert the Input Text to Bytes
```

```
 * 3. Encrypt the bytes using doFinal method
 */
strDataToEncrypt = "Hello World of Encryption using AES ";
byte[] byteDataToEncrypt = strDataToEncrypt.getBytes();
byte[] byteCipherText = aesCipher.doFinal(byteDataToEncrypt);
strCipherText = new BASE64Encoder().encode(byteCipherText);

System.out.println("Cipher Text generated using AES is " +strCipherText);
/**
 * Step 5. Decrypt the Data
 * 1. Initialize the Cipher for Decryption
 * 2. Decrypt the cipher bytes using doFinal method
 */
aesCipher.init(Cipher.DECRYPT_MODE,secretKey,aesCipher.getParameters());
byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
strDecryptedText = new String(byteDecryptedText);
System.out.println(" Decrypted Text message is " +strDecryptedText);
}
catch (NoSuchAlgorithmException noSuchAlgo)
{
System.out.println(" No Such Algorithm exists " + noSuchAlgo);
}
catch (NoSuchPaddingException noSuchPad)
{
System.out.println(" No Such Padding exists " + noSuchPad);
}
catch (InvalidKeyException invalidKey)
{
System.out.println(" Invalid Key " + invalidKey);
}
catch (BadPaddingException badPadding)
{
System.out.println(" Bad Padding " + badPadding);
}
catch (IllegalBlockSizeException illegalBlockSize)
{
System.out.println(" Illegal Block Size " + illegalBlockSize);
}
catch (
InvalidAlgorithmParameterException invalidParam)
{
System.out.println(" Invalid Parameter " + invalidParam);
}
}
}
```

2. Program #2

```
package aes;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
/**
 *
 * @author itdel
 */
public class AES_Message_inFILE {
 private static Cipher encrypt;
 private static Cipher decrypt;
 private static final byte[] initialization_vector = { 03, 43, 27, 17,
20, 4, 001, 23,
 67, 23, 11, 34,
27, 19, 73, 47 };
 String strChiperText=new String();
public static void main(String[] args) {
 //choose the right path
 String messageFile = "D……../message.txt";
 String encryptedFile = "D…… /encryptedMessage.txt";
 String decryptedFile = "D:/@. ... /decryptedMessage.txt";

 try {
 SecretKey secret_key =
KeyGenerator.getInstance("AES").generateKey();
 System.out.println(secret_key);
AlgorithmParameterSpec alogrithm_specs = new
IvParameterSpec(initialization_vector);
// set encryption mode ...
encrypt = Cipher.getInstance("AES/CBC/PKCS5Padding");
 encrypt.init(Cipher.ENCRYPT_MODE, secret_key, alogrithm_specs);

// set decryption mode
decrypt = Cipher.getInstance("AES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, secret_key, alogrithm_specs);
 // encrypt file
 encrypt(new FileInputStream(messageFile), new
FileOutputStream(encryptedFile));
 // decrypt file
 decrypt(new FileInputStream(encryptedFile), new FileOutputStream(
decryptedFile));
System.out.println("End of Encryption/Decryption procedure!");
```

```
 } catch (NoSuchAlgorithmException nsae){
 nsae.printStackTrace();
 }catch (NoSuchPaddingException nspe){
 nspe.printStackTrace();
}catch(InvalidKeyException ike){
 ike.printStackTrace();
 }catch (InvalidAlgorithmParameterException iape){
 iape.printStackTrace();
 }catch (IOException ioe){
 ioe.printStackTrace();
 }

 }
private static void encrypt(InputStream input, OutputStream output)throws
IOException {
 output = new CipherOutputStream(output, encrypt);
 writeBytes(input, output);


}
private static void decrypt(InputStream input, OutputStream output)throws
IOException {
 input = new CipherInputStream(input, decrypt);
 writeBytes(input, output);
}
private static void writeBytes(InputStream input, OutputStream output)throws
IOException {
 byte[] writeBuffer = new byte[512];
 int readBytes = 0;
 while ((readBytes = input.read(writeBuffer)) >= 0) {
 output.write(writeBuffer, 0, readBytes);
 }
 output.close();
 input.close();
}

}
```

Deliverables

Answer to **review question**, **problem** and **case** on handwritten paper.