



الجمهورية العربية السورية
جامعة دمشق

كلية الهندسة المعلوماتية
قسم هندسة البرمجيات ونظم المعلومات

اعداد الطالبة
أسماء محمد ياسر أبوكم

الفئة الثامنة
2025\11\13

المقدمة والاهداف

المقدمة

تزايدت الحاجة اليوم إلى تأمين المعلومات الرقمية بسبب انتشار الإنترنت وتبادل البيانات الحساسة بين الأفراد والمؤسسات. من أهم أساليب حماية البيانات :**التشفير الهجين** الذي يجمع بين قوة التشفير المتناظر (AES) وسهولة توزيع المفاتيح باستخدام التشفير غير المتناظر (RSA) ، والتوقيع الرقمي الذي يضمن صحة الرسالة وعدم التلاعب بها.

في هذه التجربة، يقوم الطالب بمحاكاة إرسال رسالة سرية من Alice إلى Bob بحيث يتم تشفير الرسالة بطريقة هجينية مع التوقيع الرقمي، ومن ثم يقوم Bob بفك التشفير والتحقق من صحة الرسالة.

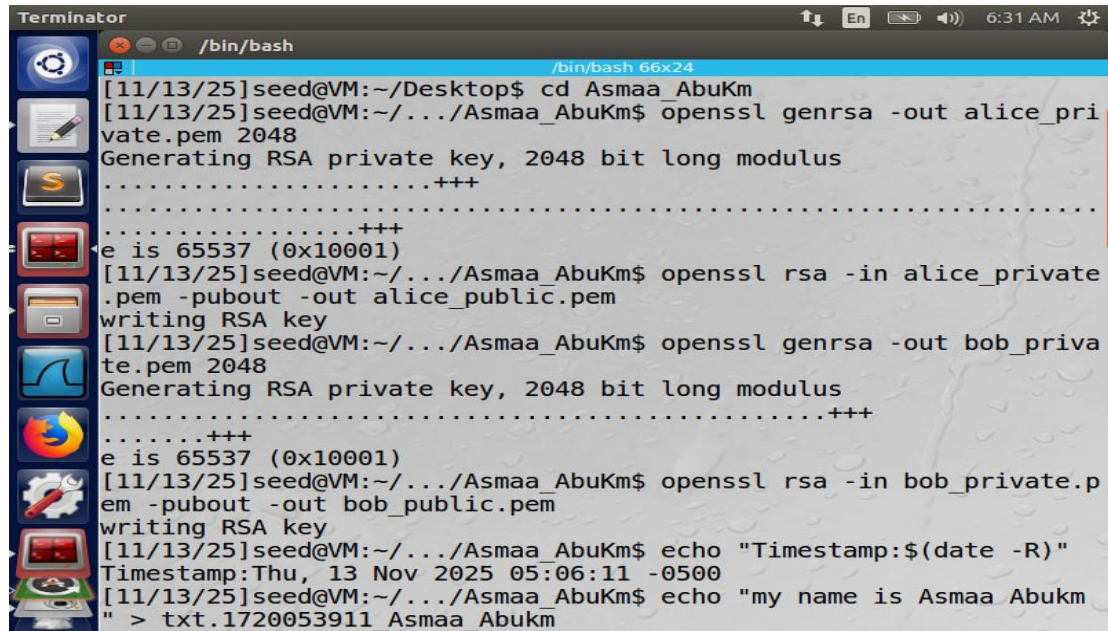
الاهداف

1. توليد مفاتيح RSA للأطراف المشاركة Alice و Bob لضمان التشفير غير المتناظر.
2. تشفير رسالة نصية باستخدام مفتاح AES-256 التشفير المتناظر.
3. تشفير مفتاح AES باستخدام مفتاح RSA العام للطرف المستقبل.
4. إنشاء توقيع رقمي للرسالة لضمان صحة البيانات وعدم التلاعب بها.
5. تجميع الملفات المرسل في أرشيف واحد (p7m) لتسهيل النقل.
6. استرجاع الرسالة وفك التشفير بواسطة الطرف المستقبل والتحقق من التوقيع الرقمي.
7. فهم الأخطاء الشائعة أثناء التشفير/فك التشفير وحلها مثل تنسيق المفتاح، IV

الإعداد والبنية التحتية

1. إنشاء المفاتيح (Alice و Bob)

- يتم إنشاء مفتاح RSA بطول 2048 بت لكل طرف.
- المفتاح العام يستخدم لتشفير الرسائل، والمفتاح الخاص لفك التشفير أو للتوقيع الرقمي.



```
Terminator /bin/bash
[11/13/25]seed@VM:~/Desktop$ cd Asmaa_AbuKm
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl genrsa -out alice_private.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl rsa -in alice_private.pem -pubout -out alice_public.pem
writing RSA key
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl genrsa -out bob_private.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl rsa -in bob_private.pem -pubout -out bob_public.pem
writing RSA key
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ echo "Timestamp:$(date -R)"
Timestamp:Thu, 13 Nov 2025 05:06:11 -0500
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ echo "my name is Asmaa Abukm" > txt.1720053911 Asmaa Abukm
```

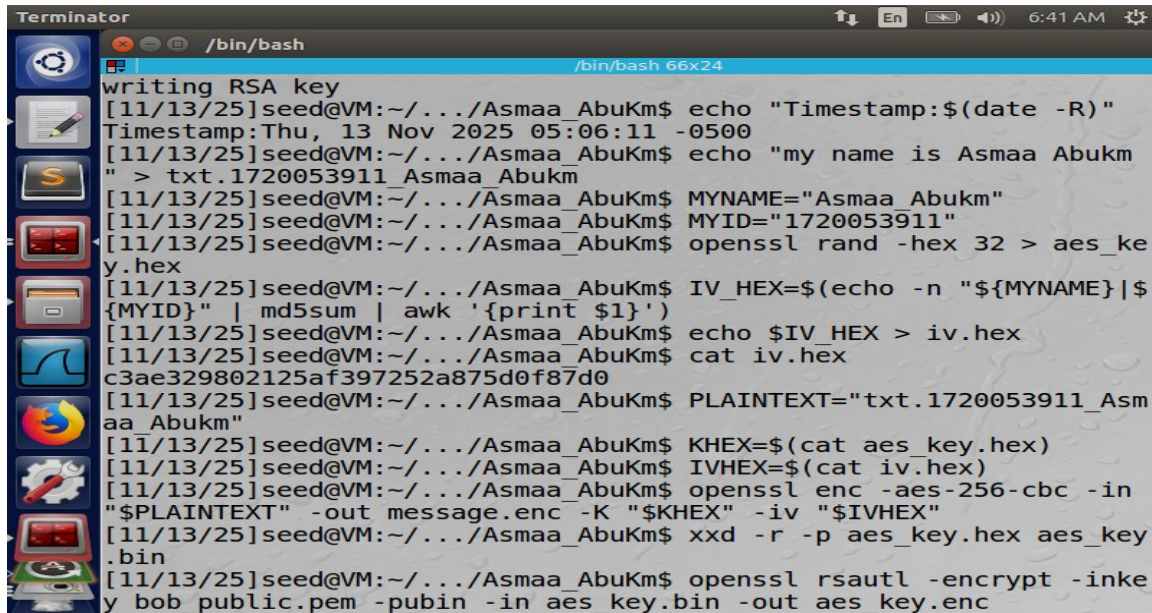
عملية الإرسال الآمن (Alice)

خطوات التشفير الهجين:

1. تشفير الرسالة بمفتاح AES-256 متناظر
 - تم توليد مفتاح AES-256 عشوائي. (aes_key.hex / aes_key.bin)
 - تم إنشاء IV من دمج الاسم والرقم الجامعي (Asmaa_Abukm|1720053911) وحسابه باستخدام MD5

2. تشفير مفتاح AES بمفتاح Bob العام (غير متناظر)

- الرسالة نفسها مشفرة AES للحصول على سرعة وكفاءة التشفير.
- مفتاح AES مشفر باستخدام RSA لضمان أن Bob فقط يستطيع فك التشفير



```
Terminator
/bin/bash
writing RSA key
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ echo "Timestamp:$(date -R)"
Timestamp:Thu, 13 Nov 2025 05:06:11 -0500
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ echo "my name is Asmaa Abukm"
" > txt.1720053911 Asmaa Abukm
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ MYNAME="Asmaa Abukm"
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ MYID="1720053911"
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl rand -hex 32 > aes_key
y.hex
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ IV_HEX=$(echo -n "${MYNAME}|${
{MYID}}" | md5sum | awk '{print $1}')
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ echo $IV_HEX > iv.hex
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ cat iv.hex
c3ae329802125af397252a875d0f87d0
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ PLAINTEXT="txt.1720053911_Asm
aa_Abukm"
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ KHEX=$(cat aes_key.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ IVHEX=$(cat iv.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl enc -aes-256-cbc -in
"$PLAINTEXT" -out message.enc -K "$KHEX" -iv "$IVHEX"
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ xxd -r -p aes_key.hex aes_key
.bin
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl rsautl -encrypt -inke
y bob public.pem -pubin -in aes_key.bin -out aes_key.enc
```

التوقيع الرقمي

- يتم حساب هاش SHA-256 للرسالة الأصلية.
- الهش يُوقع باستخدام المفتاح الخاص لـ Alice لإنتاج التوقيع الرقمي.
- هذا يضمن سلامة الرسالة ومصادقة المرسل.

عملية الاستقبال والتحقق (Bob)

فك أرشيف الرسالة

```
tar -xzf signed_and_encrypted.p7m
```

فك تشفير مفتاح AES بمفتاح Bob الخاص

```
openssl rsautl -decrypt -inkey bob_private.pem -in aes_key.enc -  
out aes_key.bin
```

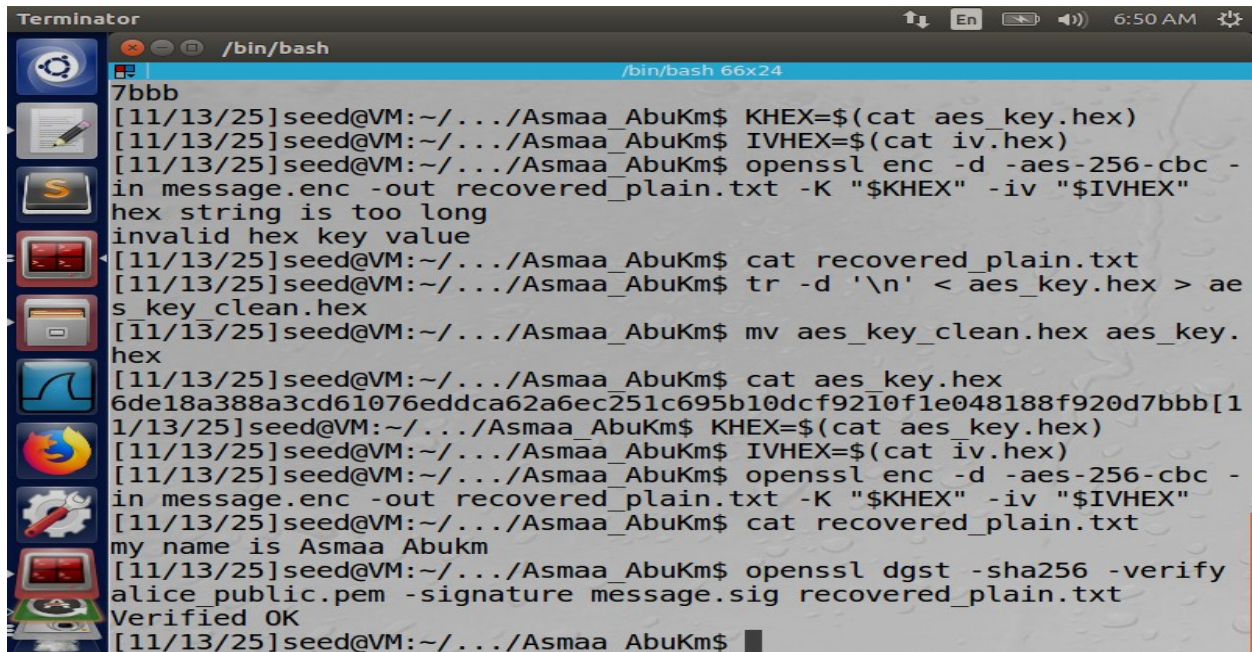
تحويل المفتاح إلى صيغة hex لتجنب أخطاء openssl

```
xxd -p aes_key.bin > aes_key.hex
```

دمج الأسطر في سطر واحد إذا احتوى المفتاح على أكثر من سطر

```
tr -d '\n' < aes_key.hex > aes_key_clean.hex
```

```
mv aes_key_clean.hex aes_key.hex
```



The screenshot shows a Terminator terminal window with a dark background and a light blue title bar. The terminal displays a series of commands and their outputs. The user 'seed@VM' is in the directory '~/.../Asmaa_AbuKm'. The commands and outputs are as follows:

```
/bin/bash
7bbb
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ KHEX=$(cat aes_key.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ IVHEX=$(cat iv.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl enc -d -aes-256-cbc -
in message.enc -out recovered_plain.txt -K "$KHEX" -iv "$IVHEX"
hex string is too long
invalid hex key value
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ cat recovered_plain.txt
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ tr -d '\n' < aes_key.hex > ae
s_key_clean.hex
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ mv aes_key_clean.hex aes_key.
hex
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ cat aes_key.hex
6de18a388a3cd61076eddca62a6ec251c695b10dcf9210f1e048188f920d7bbb[1
1/13/25]seed@VM:~/.../Asmaa_AbuKm$ KHEX=$(cat aes_key.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ IVHEX=$(cat iv.hex)
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl enc -d -aes-256-cbc -
in message.enc -out recovered_plain.txt -K "$KHEX" -iv "$IVHEX"
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ cat recovered_plain.txt
my name is Asmaa Abukm
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$ openssl dgst -sha256 -verify
alice_public.pem -signature message.sig recovered_plain.txt
Verified OK
[11/13/25]seed@VM:~/.../Asmaa_AbuKm$
```