กลุ่ม : G01 ข้อที่ 6

6509611809 ธีรภัทร ศิริธรรม

6509611544 กิตติธรา สุทธาภิรมย์

6509611858 ประพล ขาวสอาด

การอธิบายจะเริ่มจาก

1. การเตรียมข้อมูลในส่วนนี้จะเล่ากระบวนการเตรียมข้อมูลต่างๆเช่น public key , signature , certificate
2. การรับประกัน Public Key ของเซิฟเวอร์ ในรูปแบบของใบรับรองดิจิทัล (Digital Certificate)

ส่วน A : ขั้นตอนการเตรียมข้อมูล

คำตอบ 3.2.2 – 3.2.3 อภิปรายการทำงาน สิ่งที่สังเกตได้ และผลลัพธ์ของการทำงาน

ขั้นตอนที่ 1 : เราจะทำการตรวจสอบ website certificate และ chain ของ certificate โดยในที่นี้เราจะเลือก website www.reg.tu.ac.th

```
➜ CS324_Lab01 git:(master) X openssl s_client -connect www.reg.tu.ac.th:443 -showcerts
Connecting to 103.20.120.131
CONNECTED(00000003)
depth=2 C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert Global Root G2
verify return:1
depth=1 C=US, O=DigiCert Inc, OU=www.digicert.com, CN=Thawte TLS RSA CA G1
verify return:1
depth=0 CN=*.reg.tu.ac.th
verify return:1
---
Certificate chain
 0 s:CN=*.reg.tu.ac.th
   i:C=US, O=DigiCert Inc, OU=www.digicert.com, CN=Thawte TLS RSA CA G1
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Aug  9 00:00:00 2024 GMT; NotAfter: Sep  9 23:59:59 2025 GMT
-----BEGIN CERTIFICATE-----
MIIGHjCCBQagAwIBAgIQAdf2H4d0kCyJySiScy/I1zANBgkqhkiG9w0BAQsFADBe
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMR0wGwYDVQQDExRUaGF3dGUgVExTIFJTQSBDQSBHMTAe
Fw0yNDA4MDkwMDAwMDBaFw0yNTA5MDkyMzU5NTlaMBkxFzAVBgNVBAMMDioucmVn
LnR1LmFjLnRoMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArt77pcRd
ijfZAP+qK1LdmDB5v2KGXFr3lPzGgJ3FF2wD5Ys1e1CCCi6ytseUBhK+8d++XYhp
A4fO5k9cTrNCcpIILpeN6NRdCt3blErGrwHX5ff+CBqEvklQ8BNMuO59IX0Bsh53
Dh30ITgUOqOSNU9ZbrUUUs2cjn4MztwesuxrXAezdM8BLZC4eiZK/wxRGoH39S6w
7HpSFY8V9UE4E2fcpDQPFdqqOPOLuP2zZvAWpqj8RXKfPsWqeLSwK38FwZSz8yM6
sJWafGsQ6bRNmx2HVIFEkbBoHkIDjC9RK9A4rsxSc/cSMHo0+9TSNDb99T77D49U
89dc9OPXphhHSQIDAQABo4IDGzCCAxcwHwYDVR0jBBgwFoAUpYz+MszrDyzUGcYI
uAAkiF3DxbcwHQYDVR0OBBYEFCecT1kbko8tAwrl3wd2X+VUnZkSMCcGA1UdEQQg
MB6CDioucmVnLnR1LmFjLnRoggxyZWcudHUuYWMudGgwPgYDVR0gBDcwNTAzBgZn
gQwBAgEwKTAnBggrBgEFBQcCARYbaHR0cDovL3d3dy5kaWdpY2VydC5jb20vQ1BT
MA4GA1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIw
OwYDVR0fBDQwMjAwoC6gLIYqaHR0cDovL2NkcC50aGF3dGUuY29tL1RoYXd0ZVRM
```

จากนั้นเราก็ทำการบันทึก CA's Certificate และ Server's Certificate ในไฟล์ C1.pem และ C0.pem ตามลำดับซึ่งหลังจากการบันทึกได้ผลลัพธ์ตาม

```
➜ CS324_Lab01 git:(master) X cat c1.pem
-----BEGIN CERTIFICATE-----
MIIEizCCA3OgAwIBAgIQCQ7oxd5b+mLSri/3CXxIVzANBgkqhkiG9w0BAQsFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBH
MjAeFw0xNzExMDIxMjI0MjVaFw0yNzExMDIxMjI0MjVaMF4xCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxGTAXBgNVBAsTEHd3dy5kaWdpY2VydC5j
b20xHTAbBgNVBAMTFFRoYXd0ZSBUTFMgUlNBIENBIEcxMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAxjngmPhVetC0b/ozbYJdzOBUA1sMog47030cAP+P
23ANUN8grXECL8NhDEF4F1R9tL0wY0mczHaR0a71YanlxtwWo1s2uGnnyDs6mOCs
66ew2w3YETr6Tb14xgjpu1gGFtAeewaikO9Fud8hxGJTSwn8xeNkfKVWpD2L4vFN
36FNgxei1K6aE4ykgGAzNlokTp6hNOLAYpDySdLAPKzuJSQ7JCEZ6O+SDKywIdXL
oMTnpxuBKGSG88NWTo3CHCOGmQECia2yqdPDjgLqnEiYNjwQL8uMqj8rOvlMgviB
cHA7xty+7/uYLN6ZS7Vq1/F/lVhVOf5ej6jZdmB85szFbQIDAQABo4IBQDCCATww
HQYDVR0OBBYEFKWM/jLM6w8s1BnGCLgAJIhdw8W3MB8GA1UdIwQYMBaAFE4iVCAY
lebjbuYP+vq5Eu0GF485MA4GA1UdDwEB/wQEAwIBhjAdBgNVHSUEFjAUBggrBgEF
BQcDAQYIKwYBBQUHAwIwEgYDVR0TAQH/BAgwBgEB/wIBADA0BggrBgEFBQcBAQQo
MCYwJAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLmRpZ21jZXJ0LmNvbTBCBgNVHR8E
OzA5MDegNaAzhjFodHRwOi8vY3JsMy5kaWdpY2VydC5jb20vRGlnaUNlcnRHbG9i
YWxSb290RzIuY3JsMD0GA1UdIAQ2MDQwMgYEVR0gADAqMCgGCCsGAQUFBwIBFhxo
dHRwczovL3d3dy5kaWdpY2VydC5jb20vQ1BTMA0GCSqGSIb3DQEBCwUAA4IBAQC6
km0KA4sTb2VYpEBm/uL2HL/pZX9B7L/hbJ4NcoBe7V56oCnt7aeIo8sMjCRWTCWZ
D1dY0+2KZOC1dKj8d1VXXAtnjytDDuPPf6/iow0mYQTO/GAg/MLyL6CDm3FzDB8V
tsH/aeMgP6pgD1XQqz+haDnfnJTKBuxhcpnx3Adbleue/QnPf1hHYa8L+Rv8Pi5U
h4V9FwHOfphdMXOxi14OqmsiTbc5cOs9/uukH+YVsuFdWTna6IVw1qh+tEtyH16R
vmi7pkqyZYULOPMIE7avrljVVBZuikwARtY8tCVV6Pp9l3VeagBqb2ffgqNJt3C0
TYNYQI+BXG1R1cABlold
-----END CERTIFICATE-----%
```

```
➜ CS324_Lab01 git:(master) X cat c0.pem
-----BEGIN CERTIFICATE-----
MIIGHjCCBQagAwIBAgIQAdf2H4d0kCyJySiScy/I1zANBgkqhkiG9w0BAQsFADBe
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMR0wGwYDVQQDExRUaGF3dGUgVExTIFJTQSBDQSBHMTAe
Fw0yNDA4MDkwMDAwMDBaFw0yNTA5MDkyMzU5NTlaMBkxFzAVBgNVBAMMDioucmVn
LnR1LmFjLnRoMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArt77pcRd
ijfZAP+qK1LdmDB5v2KGXFr3lPzGgJ3FF2wD5Ys1e1CCCi6ytseUBhK+8d++XYhp
A4fO5k9cTrNCcpIILpeN6NRdCt3blErGrwHX5ff+CBqEvklQ8BNMuO59IX0Bsh53
Dh30ITgUOqOSNU9ZbrUUUs2cjn4MztwesuxrXAezdM8BLZC4eiZK/wxRGoH39S6w
7HpSFY8V9UE4E2fcpDQPFdqqOPOLuP2zZvAWpqj8RXKfPsWqeLSwK38FwZSz8yM6
sJWafGsQ6bRNmx2HVIFEkbBoHkIDjC9RK9A4rsxSc/cSMHo0+9TSNDb99T77D49U
89dc9OPXphhHSQIDAQABo4IDGzCCAxcwHwYDVR0jBBgwFoAUpYz+MszrDyzUGcYI
uAAkiF3DxbcwHQYDVR0OBBYEFCecT1kbko8tAwrl3wd2X+VUnZkSMCcGA1UdEQQg
MB6CDioucmVnLnR1LmFjLnRoggxyZWcudHUuYWMudGgwPgYDVR0gBDcwNTAzBgZn
gQwBAgEwKTAnBggrBgEFBQcCARYbaHR0cDovL3d3dy5kaWdpY2VydC5jb20vQ1BT
MA4GA1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIw
OwYDVR0fBDQwMjAwoC6gLIYqaHR0cDovL2NkcC50aGF3dGUuY29tL1RoYXd0ZVRM
U1JTQUNBRzEuY3JsMHAGCCsGAQUFBwEBBGQwYjAkBggrBgEFBQcwAYYYaHR0cDov
L3N0YXR1cy50aGF3dGUuY29tMDoGCCsGAQUFBzAChi5odHRwOi8vY2FjZXJ0cy50
aGF3dGUuY29tL1RoYXd0ZVRMU1JTQUNBRzEuY3J0MAwGA1UdEwEB/wQCMAAwggF+
BgorBgEEAdZ5AgQCBIIBbgSCAWoBaAB2ABLxTjS9U3JMhAYZw48/ehP457Vih4ic
bTAFhOvlhiY6AAABkTYLgP8AAAQDAEcwRQIgELiI97kXElZM/PPLdaY7f4H5EwQO
Ff6i/omtjZ7QHuICIQDqJ3f8g1uFeV8IfRtvRIltyDoJZv0hIvigDQmC9wLc6wB2
AH1ZHhLheCp7HGFnfF79+NCHXBSgTpWeuQMv2Q6MLnm4AAABkTYLgPcAAAQDAEcw
RQIhAJD1cI6RFDp3aVB5Dti3vO11wrMjUBPMxvJMyK+Keg4LAiAaaoqoaukPbo5M
5s3zuBNaRs7fIFv7HB9B0/S0tUPhuQB2AObSMWNAd4zBEEEG13G5zsHSQPaWhIb7
uocyHf0eN45QAAABkTYLgQkAAAQDAEcwRQIgG2luG//I2rLhbOItNnukz9uyhz1f
v7NkIJvXY6LmVEACIQDdfwBa76oglIDqIuAEoQSchKMACK9LZv+ORDNfJZ5i2DAN
BgkqhkiG9w0BAQsFAAOCAQEAKL8vGM0FmbCoZz+M7FP2ZFKu02xVukyz52HC0Yx1
xlMrNPJ/u63ipesYSVqvTs6l3bRzL6TtQFXEWklyOjkZpdBqL2Lpr5Xau/TozDw5
+b/vBWdhPrDgQyj2SNYqpOyaMkNUt01ddAGn/5oy3gHATJz37PTqXJHKQdiRF90G
GHPOoAFRpl+18p9F4HOKwmsRZYwo9ulXbc++249Kwfso1QP2sB8e1pKteD5lTGSv
CC4yk1WodmPG/oj4jDKBSSAUit3Ihfllol ZxQpZ6rV+7jTXqhZGNiW5J+Pw3xdGI
DGB0RLk9X70kT3nYB9dzvHY8mRwQuNqq3OTvaejfuTVBYg==
-----END CERTIFICATE-----%
```

ขั้นตอนที่ 2 เราต้องการ public key (ค่า e, n) ของ c1.pem , CA’s certificate

จากรูปทางด้านล่างเราใช้คำสั่งจาก openssl เพื่อ extract ค่า n ออกมาได้ผลลัพธ์ทางด้านล่าง

```
➜ CS324_Lab01 git:(master) X openssl x509 -in c1.pem -noout -modulus
Modulus=C639E098F8557AD0B46FFA336D825DCCE054035B0CA20E3BD37D1C00FF8FDB700D50DF20AD71022FC3610C417817547DB4BD3063499CCC7691D1AEE561A9E5C6DC
16A35B36B869E7C83B3A98E0ACEBA7B0DB0DD8113AFA4DBD78C608E9BB580616D01E7B06A290EF45B9DF21C462534B09FCC5E3647CA556A43D8BE2F14DDFA14D8317A294AE
9A138CA4806033365A244E9EA134E2C06290F249D2C03CACEE25243B242119E8EF920CACB021D5CBA0C4E7A71B81286486F3C3564E8DC21C238699010289ADB2A9D3C38E02
EA9C4898363C102FCB8CAA3F2B3AF94C82F88170703BC6DCBEEFFB982CDE994BB56AD7F17F95585539FE5E8FA8D976607CE6CCC56D
➜ CS324_Lab01 git:(master) X
```

เรา extract e ออกมาได้ผลลัพธ์ตามด้านล่าง

```
➜ CS324_Lab01 git:(master) X openssl x509 -in c1.pem -text -noout | grep Exponent
            Exponent: 65537 (0x10001)
➜ CS324_Lab01 git:(master) X
```

ขั้นตอนที่ 3 extract signature จาก server certificate จากนั้นนำไปจัดรูปแบบให้เรียบร้อยได้ผลลัพธ์ตามรูปด้านล่าง

```
➜ CS324_Lab01 git:(master) X cat signature| tr -d '[:space:]:'
28bf2f18cd0599b0a8673f8cec53f66452aed36c55ba4cb3e761c2d18c75c6532b34f27fbbade2a5eb18495aaf4ecea5ddb4732fa4e
d4055c45a49723a3919a5d06a2f62e9af95dabbf4e8cc3c39f9bfef0567613eb0e04328f648d62aa4ec9a324354b74d5d7401a7ff9a
32de01c04c9cf7ecf4ea5c91ca41d89117dd061873cea00151a65fb5f29f45e0738ac26b11658c28f6e9576dcfbedb8f4ac1fb28d50
3f6b01f1ed692ad783e654c64af082e329355a87663c6fe88f88c32814920148addc885f965a2567142967aad5fbb8d35ea85918d89
6e49f8fc37c5d1880c607444b93d5fbd244f79d807d773bc763c991c10b8daaadce4ef69e8dfb93541625
```

ขั้นตอนที่ 4 แกะ server certificate ของ www.reg.tu.ac.th โดย X.509 cert จะถูก encode ใน Format ASN.1 standard และตรงตาม instruction จะได้

```
➜ CS324_Lab01 git:(master) X openssl asn1parse -i -in c0.pem
    0:d=0  hl=4 l=1566 cons: SEQUENCE
    4:d=1  hl=4 l=1286 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  16 prim:   INTEGER           :01D7F61F8774902C89C92892732FC8D7
   31:d=2  hl=2 l=  13 cons:   SEQUENCE
   33:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   44:d=3  hl=2 l=   0 prim:    NULL
   46:d=2  hl=2 l=  94 cons:   SEQUENCE
   48:d=3  hl=2 l=  11 cons:    SET
   50:d=4  hl=2 l=   9 cons:     SEQUENCE
   52:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
   57:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   61:d=3  hl=2 l=  21 cons:    SET
   63:d=4  hl=2 l=  19 cons:     SEQUENCE
   65:d=5  hl=2 l=   3 prim:      OBJECT            :organizationName
   70:d=5  hl=2 l=  12 prim:      PRINTABLESTRING   :DigiCert Inc
   84:d=3  hl=2 l=  25 cons:    SET
   86:d=4  hl=2 l=  23 cons:     SEQUENCE
   88:d=5  hl=2 l=   3 prim:      OBJECT            :organizationalUnitName
   93:d=5  hl=2 l=  16 prim:      PRINTABLESTRING   :www.digicert.com
  111:d=3  hl=2 l=  29 cons:    SET
  113:d=4  hl=2 l=  27 cons:     SEQUENCE
  115:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  120:d=5  hl=2 l=  20 prim:      PRINTABLESTRING   :Thawte TLS RSA CA G1
  142:d=2  hl=2 l=  30 cons:   SEQUENCE
  144:d=3  hl=2 l=  13 prim:    UTCTIME           :240809000000Z
  159:d=3  hl=2 l=  13 prim:    UTCTIME           :2509092359592
7D591E12E1782A7B1C61677C5EFDF8D0875C14A04E959EB9032FD90E8C2E79B800000191360B80F700000403004730450221009 0F57
08E91143A776950790ED8B7BCE975C2B3235013CCC6F24CC8AF8A7A0E0B02201A6A8AA86AE90F6E8E4CE6CDF3B8135A46CEDF205BFB
1C1F41D3F4B4B543E1B9007600E6D2316340778CC1104106D771B9CEC1D240F6968486FBBA87321DFD1E378E5000000191360B81090
000040300473045022 01B696E1BFFC8DAB2E16CE22D367BA4CFDBB2873D5FBFB364209BD763A2E65440022100DD7F005AEFAA209480
EA22E004A1049C84A30008AF4B66FF8E44335F259E62D8
 1294:d=1  hl=2 l=  13 cons:  SEQUENCE
 1296:d=2  hl=2 l=   9 prim:   OBJECT            :sha256WithRSAEncryption
 1307:d=2  hl=2 l=   0 prim:   NULL
 1309:d=1  hl=4 l= 257 prim:  BIT STRING
```

คำนวณ hash โดยใช้ algorithm sha-256 ได้ผลตามรูปภาพด้านล่าง

```
→ CS324_Lab01 git:(master) X openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
→ CS324_Lab01 git:(master) X sha256sum c0_body.bin
1752c78ddac36afcf357ba23160e193228519ba1e0fb0bdba2da3d858df151df  c0_body.bin
```

ขั้นตอนที่ 5 verify the signature โดยใช้ C Code

ส่วน B : การรับประกัน Public Key ของเซิฟเวอร์ ในรูปแบบของใบรับรองดิจิทัล (Digital Certificate)

คำตอบในข้อ 3.2.1

ชุดคำสั่งทั้งภาษาซีที่นักศึกษาใช้ในการทำงานตามข้อกำหนด Task 6 ของ Lab Sheet พร้อมคำอธิบายการทำงานของแต่ละคำสั่ง (สามารถเขียนในรูปแบบคอมเมนต์ประกอบแต่ละคำสั่งได้)

Code ทั้งหมดของข้อที่ 6

```c
#include <stdio.h>
#include <string.h>
#include <openssl/bn.h>

#define NBIT 128


// This is Signature From the website https://reg.tu.ac.th
const char* signature =
"28bf2f18cd0599b0a8673f8cec53f66452aed36c55ba4cb3e761c2d18c75c6532b34f27fbbade
2a5eb18495aaf4ecea5ddb4732fa4ed4055c45a49723a3919a5d06a2f62e9af95dabbf4e8cc3c3
9f9bfef0567613eb0e04328f648d62aa4ec9a324354b74d5d7401a7ff9a32de01c04c9cf7ecf4e
a5c91ca41d89117dd061873cea00151a65fb5f29f45e0738ac26b11658c28f6e9576dcfbedb8f4
ac1fb28d503f6b01f1ed692ad783e654c64af082e329355a87663c6fe88f88c32814920148addc
885f965a2567142967aad5fbb8d35ea85918d896e49f8fc37c5d1880c607444b93d5fbd244f79d
807d773bc763c991c10b8daaadce4ef69e8dfb9354162";
// Step 2 : n
const char* modulus =
"C639E098F8557AD0B46FFA336D825DCCE054035B0CA20E3BD37D1C00FF8FDB700D50DF20AD710
22FC3610C417817547DB4BD3063499CCC7691D1AEE561A9E5C6DC16A35B36B869E7C83B3A98E0A
CEBA7B0DB0DD8113AFA4DBD78C608E9BB580616D01E7B06A290EF45B9DF21C462534B09FCC5E36
47CA556A43D8BE2F14DDFA14D8317A294AE9A138CA4806033365A244E9EA134E2C06290F249D2C
03CACEE25243B242119E8EF920CACB021D5CBA0C4E7A71B81286486F3C3564E8DC21C238699010
289ADB2A9D3C38E02EA9C4898363C102FCB8CAA3F2B3AF94C82F88170703BC6DCBEEFFB982CDE9
94BB56AD7F17F95585539FE5E8FA8D976607CE6CCC56D";
// Step 2 : e
const char* expo = "10001";

// hash bodycert that want to check is equal to decrypted signature?
const char* bodycert_hash =
"1752C78DDAC36AFCF357BA23160E193228519BA1E0FB0BDBA2DA3D858DF151DF"; //for
checking
```

```c
void printBN(char *msg, BIGNUM * a)
{
/* Use BN_bn2hex(a) for hex string
* Use BN_bn2dec(a) for decimal string */
char * number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}

int main(){
    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* S = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* result = BN_new();

    BN_hex2bn(&S , signature);
    BN_hex2bn(&e , expo);
    BN_hex2bn(&n , modulus);

    BN_mod_exp(result , S , e , n , ctx);


    //get only last 64 digit of result (get rid of padding and other)
    char decrypt_signature[65];
    char* res_string = BN_bn2hex(result);
    strncpy(decrypt_signature,res_string+strlen(res_string)-64, 64);

    printBN("Decrypted Signature : ", result);
    printf("Decrypted Signature : %s\n",decrypt_signature);
    printf("\nHashed cert : %s\n", bodycert_hash);

    //free memory
    OPENSSL_free(res_string);
    BN_CTX_free(ctx);
    BN_free(S);
    BN_free(e);
    BN_free(n);
    BN_free(result);
    return 0;
}
```

อธิบายการทำงานของ code ทีละส่วน

ส่วนที่ 1 ค่า public key (e, n) ได้จากขั้นตอนที่ 2

```
const char* modulus =
"C639E098F8557AD0B46FFA336D825DCCE054035B0CA20E3BD37D1C00FF8FDB700D50DF20AD710
22FC3610C417817547DB4BD3063499CCC7691D1AEE561A9E5C6DC16A35B36B869E7C83B3A98E0A
CEBA7B0DB0DD8113AFA4DBD78C608E9BB580616D01E7B06A290EF45B9DF21C462534B09FCC5E36
47CA556A43D8BE2F14DDFA14D8317A294AE9A138CA4806033365A244E9EA134E2C06290F249D2C
03CACEE25243B242119E8EF920CACB021D5CBA0C4E7A71B81286486F3C3564E8DC21C238699010
289ADB2A9D3C38E02EA9C4898363C102FCB8CAA3F2B3AF94C82F88170703BC6DCBEEFFB982CDE9
94BB56AD7F17F95585539FE5E8FA8D976607CE6CCC56D";
const char* expo = "10001";
```

ส่วนที่ 2 Signature ของ www.reg.tu.ac.th ที่ได้รับมา

```
const char* signature =
"28bf2f18cd0599b0a8673f8cec53f66452aed36c55ba4cb3e761c2d18c75c6532b34f27fbbade
2a5eb18495aaf4ecea5ddb4732fa4ed4055c45a49723a3919a5d06a2f62e9af95dabbf4e8cc3c3
9f9bfef0567613eb0e04328f648d62aa4ec9a324354b74d5d7401a7ff9a32de01c04c9cf7ecf4e
a5c91ca41d89117dd061873cea00151a65fb5f29f45e0738ac26b11658c28f6e9576dcfbedb8f4
ac1fb28d503f6b01f1ed692ad783e654c64af082e329355a87663c6fe88f88c32814920148addc
885f965a2567142967aad5fbb8d35ea85918d896e49f8fc37c5d1880c607444b93d5fbd244f79d
807d773bc763c991c10b8daaadce4ef69e8dfb9354162";
```

สิ่งที่เราคาดหวังคือการต้องการทวนสอบว่า signature นั้นมีความสอดคล้องกับ public key หรือไม่

ส่วนที่ 3 ใช้ OpenSSL Library ในการคำนวณ RSA Signature Verification จากสมการซึ่งคือ

$$\text{result} = S^e \mod n$$

```
    BN_CTX* ctx = BN_CTX_new(); //สร้าง context สำหรับการคำนวณ
    BIGNUM* S = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* result = BN_new();

    BN_hex2bn(&S , signature);
    BN_hex2bn(&e , expo);
    BN_hex2bn(&n , modulus);

    BN_mod_exp(result , S , e , n , ctx); // คำนวณค่าที่ได้
```

จากนั้นทำการ print console ได้ผลลัพธ์จากการคำนวณดังนี้

```
● ➜ CS324_Lab01 git:(master) ✗ ./run_6
  Decrypted Signature :  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D060960864
  8016503040201050004201752C78DDAC36AFCF357BA23160E1932285
  19BA1E0FB0BDBA2DA3D858DF151DF
  Decrypted Signature : 1752C78DDAC36AFCF357BA23160E193228
  519BA1E0FB0BDBA2DA3D858DF151DF

  Hashed cert : 1752C78DDAC36AFCF357BA23160E193228519BA1E0
  FB0BDBA2DA3D858DF151DF
```

หมายเหตุ ./run_6 คือ script ในการ run ซึ่งภายในคือคำสั่ง

```
● ➜ CS324_Lab01 git:(master) ✗ cat run_6
  gcc CS324_Security_Lab01-Cryptography-RSA_G01_task-6.c -lcrypto -o CS324_Security_Lab01-Cryptograph
  y-RSA_G01_task-6.o
  ./CS324_Security_Lab01-Cryptography-RSA_G01_task-6.o %
```

จากข้อมูลที่ได้นั้นสามารถแบ่งออกได้เป็น 3 ส่วน

1. Prefix “01FFF .. FF” ส่วนนี้คือ PKCS#1 v1.5 padding schema
2. Hash Algoithm Identifier (003031300D060960864801650304020 1)
3. Actual Hash Value

   (1752C78DDAC36AFCF357BA23160E193228519BA1E0FB0BDBA2DA3D858DF151DF)

ซึ่งในที่นี้เราจะสนใจ 64 bit สุดท้าย (ในที่นี้คือ decrypt signature) และผลลัพท์ออกมาก็จะได้เท่ากับ certificate (ที่ได้รับจากขั้นตอนที่ 4) ที่ได้รับมาแสดงว่า certificate valid

ส่วนที่ 5 Free Memory ก่อนจบโปรแกรม

```
    //free memory
    OPENSSL_free(res_string);
    BN_CTX_free(ctx);
    BN_free(S);
    BN_free(e);
    BN_free(n);
    BN_free(result);
    return 0;
```