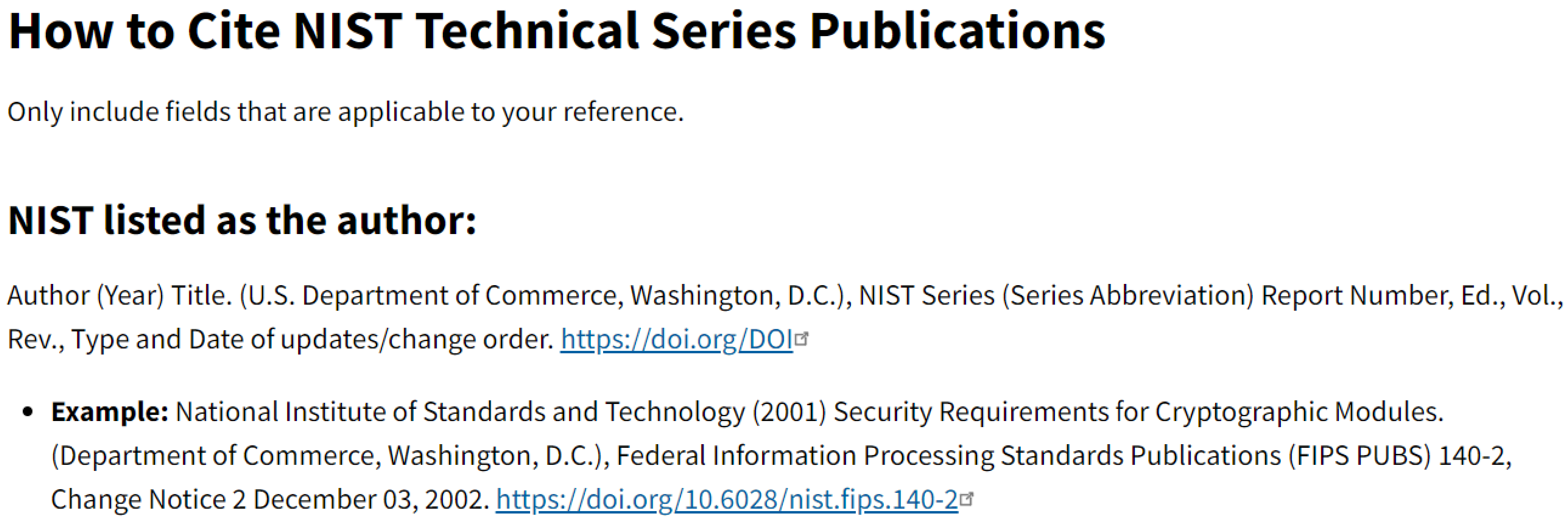
Managing Enterprise Cybersecurity MIS 4596

Class 10

Asymmetric Cryptography and Digital Signatures

References Format for NIST Publications



How to Cite NIST Technical Series Publications

Only include fields that are applicable to your reference.

NIST listed as the author:

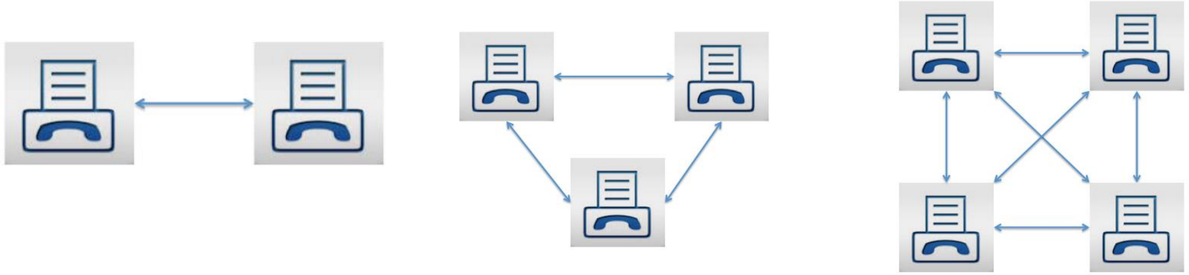
Author (Year) Title. (U.S. Department of Commerce, Washington, D.C.), NIST Series (Series Abbreviation) Report Number, Ed., Vol., Rev., Type and Date of updates/change order. https://doi.org/DOI

- **Example:** National Institute of Standards and Technology (2001) Security Requirements for Cryptographic Modules. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publications (FIPS PUBS) 140-2, Change Notice 2 December 03, 2002. https://doi.org/10.6028/nist.fips.140-2

Agenda

- Key sharing problem: Metcalf’s Law
- Symmetric key sharing with Diffie-Hellman public key algorithm
- Man-In-The-Middle Attack
- RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- Hybrid Encryption
- Session Key
- Digital Signature
- Finding algorithms in internet browser certificates

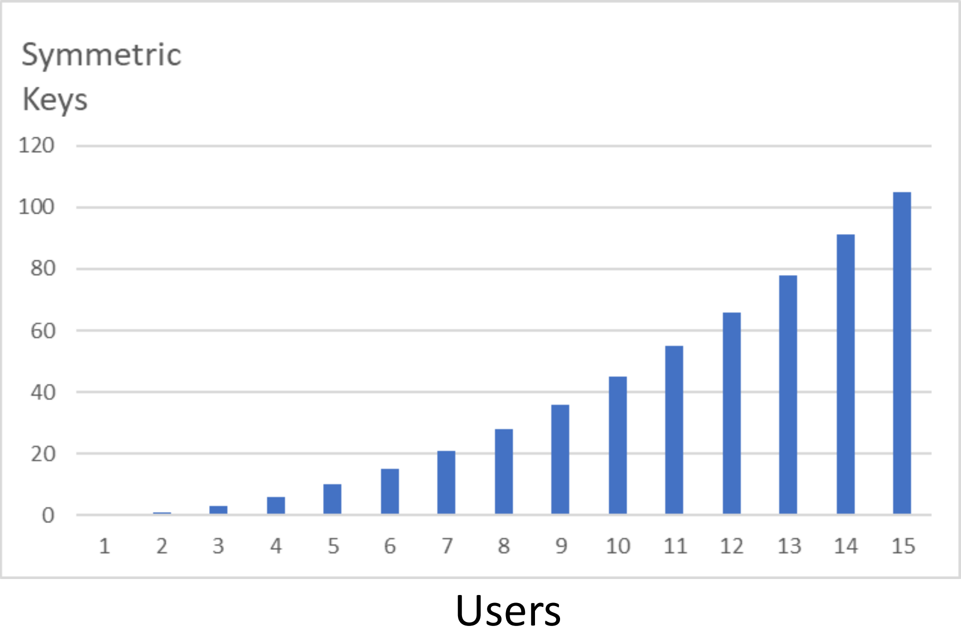
Key sharing problem



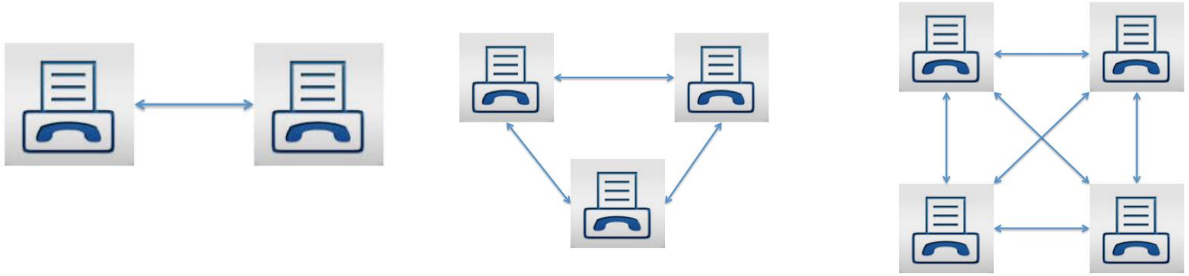
Sharing cryptographic keys has been a problem throughout history

- The number of pairs of keys (“secure network connections”) grows at a near exponential rate (i.e. geometric rate) as the number of users increases

Users	Symmetric Keys
1	0
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45
11	55
12	66
13	78
14	91
15	105
…	…



Key sharing problem



- The number of pairs of keys needed for “n” users is determined by an equation known as Metcalf’s Law
- Number of key pairs needed for n users = (n*(n-1))/2
  - *The reason for the n-1 is that you do not need to communicate with yourself*

- For MIS 4596 with 20 students how many keys would we need:

(20 *19)/2 = 190 keys

Agenda

✓ Key sharing problem: Metcalf’s Law

- Symmetric key sharing with Diffie-Hellman public key algorithm
- Man-In-The-Middle Attack
- RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- Hybrid Encryption
- Session Key
- Digital Signature
- Finding algorithms in internet browser certificates

Diffie-Hellman Algorithm: *Secret symmetric key derivation through public key sharing*

Assumptions:

A prime number is a positive whole number whose only factors (i.e. integer divisors) are 1 and the number itself (e.g. 2, 3, 5, 7, 11, 13, 17, 19, 23, …). Bob & Alice want to compute a shared secret key to protect confidentiality of their conversation. Eve eavesdrops…

Algorithm:

1. Bob & Alice publicly agree on "**p**" called *prime modulus* (e.g. **p = 23**) & "**g**" called *generator* (e.g. **g = 5**), Eve overhears
2. Bob & Alice each choose their own secret key:
   - Bob's secret key is referred to as "**x_bob**" which is a number between 1 and p-1 (e.g. **x_bob = 12**)
   - Alice's secret key is referred to as "**x_alice**" which also is a number between 1 and p-1 (e.g**. x_alice = 7**)
3. Bob & Alice each computes their own public key, which they share with each other and Eve intercepts…
   - Bob computes: **y_bob = g^{x_bob} mod p** which is: **y_bob = 5^{12} mod 23 = 18** which he shares with Alice (and Eve)
   - Alice computes: **y_alice = g^{x_alice} mod p** which is: **y_alice = 5^{7} mod 23 = 17** which she shares with Bob (and Eve)
4. Bob & Alice each compute their shared secret symmetric key
   - Bob computes: **$y\_alice^{x\_bob}$ mod p** which is: **17^{12} mod 23 = 6**
   - Alice computes: **$y\_bob^{x\_alice}$ mod p** which is: **18^{7} mod 23 = 6**
5. Bob & Alice now have a **shared secret ("symmetric") key = 6**
6. Eve has Bob & Alice's public keys: y_bob=18 & y_alice=17, prime modulus: p=23 and generator: g=5, but not their secret keys x_bob = 12 & x_alice = 7
   - ***Eve cannot calculate Bob& Alice's shared symmetric secret key from their public keys, p and g alone – even though she knows they are using the Diffie-Hellman algorithm!***

let's use WolframAlpha online calculator

Generic Diffie-Hellman Algorithm

Run the algorithm together

1. Bob & Alice publicly agree on “**p**” called *prime modulus* (e.g. **p = 31**) and “**g**” called *generator* (e.g. **g = 3**), Eve overhears
2. Bob & Alice each choose your own secret key:
   - Bob’s secret key is referred to as “**x_bob**” which is a number between 1 and p-1 (write it down but keep it secret)
   - Alice’s secret key is referred to as “**x_alice**” which also is a number between 1 and p-1 (write it down but keep it secret)
3. Bob & Alice each computes your own public key, which you share with each other, and Eve intercepts…
   - Bob computes: **y_bob = g^{x_bob} mod p** which he writes down and shares with Alice (and Eve)
   - Alice computes: **y_alice = g^{x_alice} mod p** which she writes down and shares with Bob (and Eve)
4. Bob & Alice each secretly compute your shared secret symmetric key which you write down and do not share
   - Bob computes: **$y\_alice^{x\_bob}$ mod p** which he writes down and does not share
   - Alice computes: **$y\_bob^{x\_alice}$ mod p** which she writes down and does not share
5. Bob & Alice now have a **shared secret (“symmetric”) key** compare your secret keys and confirm they are the same
6. Eve has Bob & Alice’s public keys: **y_bob** & **y_alice**, prime modulus: **p** and generator: **g**, but not their secret keys **x_bob** & **x_alice**
   - ***Eve cannot calculate Bob& Alice’s shared symmetric secret key from their public keys, p and g alone – even though she knows they are using the Diffie-Hellman algorithm!***

In practice, *p* must be much larger prime number… this is a 4096-bit *p*

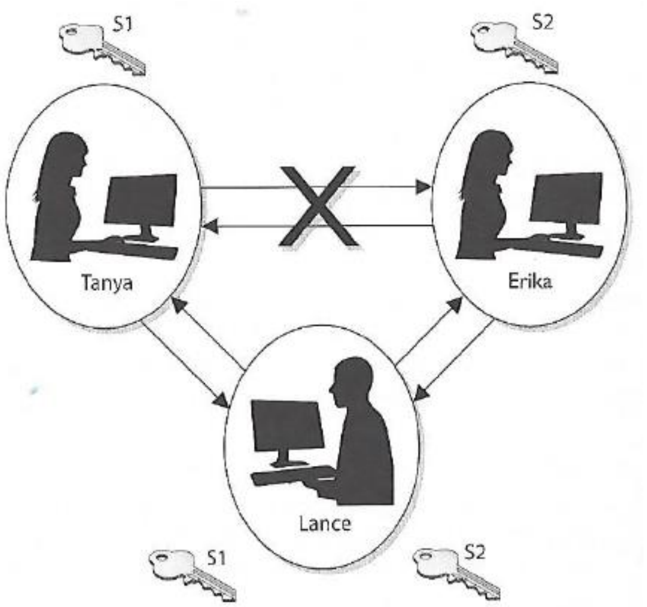
857,756,147,438,808,767,721,482,523,862,479,196,091,217,066,271,200,126,894,701,702,329,327,872,802,487,425,224,246,373,206,756,773,954,180,315,945,664,685,564,049,690,107,228,861,210,053,005,306,168,041,237,244,792,245,832,497,260,206,801,417,396,745,674,574,281,768,112,711,519,809,332,223,737,878,554,093,201,446,763,995,425,025,965,323,912,149,043,161,823,975,594,943,915,411,109,637,902,372,642,611,214,196,649,667,036,726,005,577,041,694,781,738,635,943,018,156,362,403,714,091,905,448,620,990,965,500,814,912,289,738,636,687,051,381,358,564,729,963,735,782,176,280,511,819,070,673,927,579,180,484,836,950,910,945,840,410,470,935,832,100,360,510,117,962,261,152,920,101,946,255,789,679,435,711,472,267,368,823,730,863,971,596,718,223,674,224,106,003,985,209,174,353,308,077,140,794,884,546,003,360,030,727,697,326,025,663,819,442,780,105,880,604,943,197,516,223,343,068,846,392,924,237,875,653,640,416,933,764,628,191,065,601,980,281,442,005,263,033,849,543,723,716,743,986,123,624,356,871,152,793,177,027,462,801,070,011,526,783,269,474,338,816,734,553,122,757,257,382,121,230,562,181,721,318,331,271,107,036,972,788,062,816,322,387,506,944,045,038,739,178,684,349,474,317,534,892,731,313,651,324,179,101,369,222,316,429,969,662,605,450,068,078,088,031,941,042,867,503,697,721,512,539,949,128,099,005,160,179,345,242,776,041,458,121,259,813,719,561,319,392,760,414,249,584,984,440,063,314,771,039,261,920,249,005,444,014,069,555,961,131,639,966,539,872,980,057,279,636,609,441,274,119,014,567,294,590,620,498,019,375,631,405,622,479,332,810,401,520,856,695,524,524,855,468,645,479,042,909,834,183,316,487,318,824,544,358,235,183,243,643

Diffie-Hellman

- Uses asymmetric public and private keys to exchange a symmetric key
- Does not use asymmetric keys for confidentiality (i.e. to encrypt or decrypt any messages)
- Users/systems need to negotiate a new key for every new person
- No authentication, no non-repudiation

Diffie-Hellman was vulnerable to man-in-the-middle attack, because no authentication occurs before public keys are exchanged

1. Tanya sends her public key to Erika, but Lance grabs the key during transmission so it never makes it to Erika
2. Lance spoofs Tanya’s identity and sends over his public key to Erika. Erika now thinks she has Tanya’s public key
3. Erika sends her public key to Tanya, but Lance grabs the key during transmission so it never makes it to Tanya
4. Lance spoofs Erika’s identity and sends over his public key to Tanya. Tanya now thinks she has Erika’s public key
5. Tanya combines her private key and Lance’s public key and creates a symmetric key S1
6. Lance combines his private key and Tanya’s public key and creates symmetric key S1
7. Erika combines her private key and Lance’s public key and creates symmetric key S2
8. Lance combines his private key and Erika’s public key and creates symmetric key S2
9. Now Tanya and Lance share a symmetric key (S1) and Erika and Lance share a different symmetric key (S2). Tanya and Erika think they are sharing a key between themselves and do not realize Lance is involved
10. Tanya writes a message to Erika, and uses her symmetric key (S1) to encrypt the message, and sends it
11. Lance grabs the message and decrypts it with symmetric key S1, reads or modifies the message and re-encrypts it with symmetric key S2, and then sends it to Erika
12. Erika take symmetric key S2 and uses it to decrypt and read the message….



Harris, S. and Maymi, F. (2019) All-In-One CISSP Exam Guide

Agenda

✓Key sharing problem: Metcalf’s Law

✓Symmetric key sharing with Diffie-Hellman public key algorithm

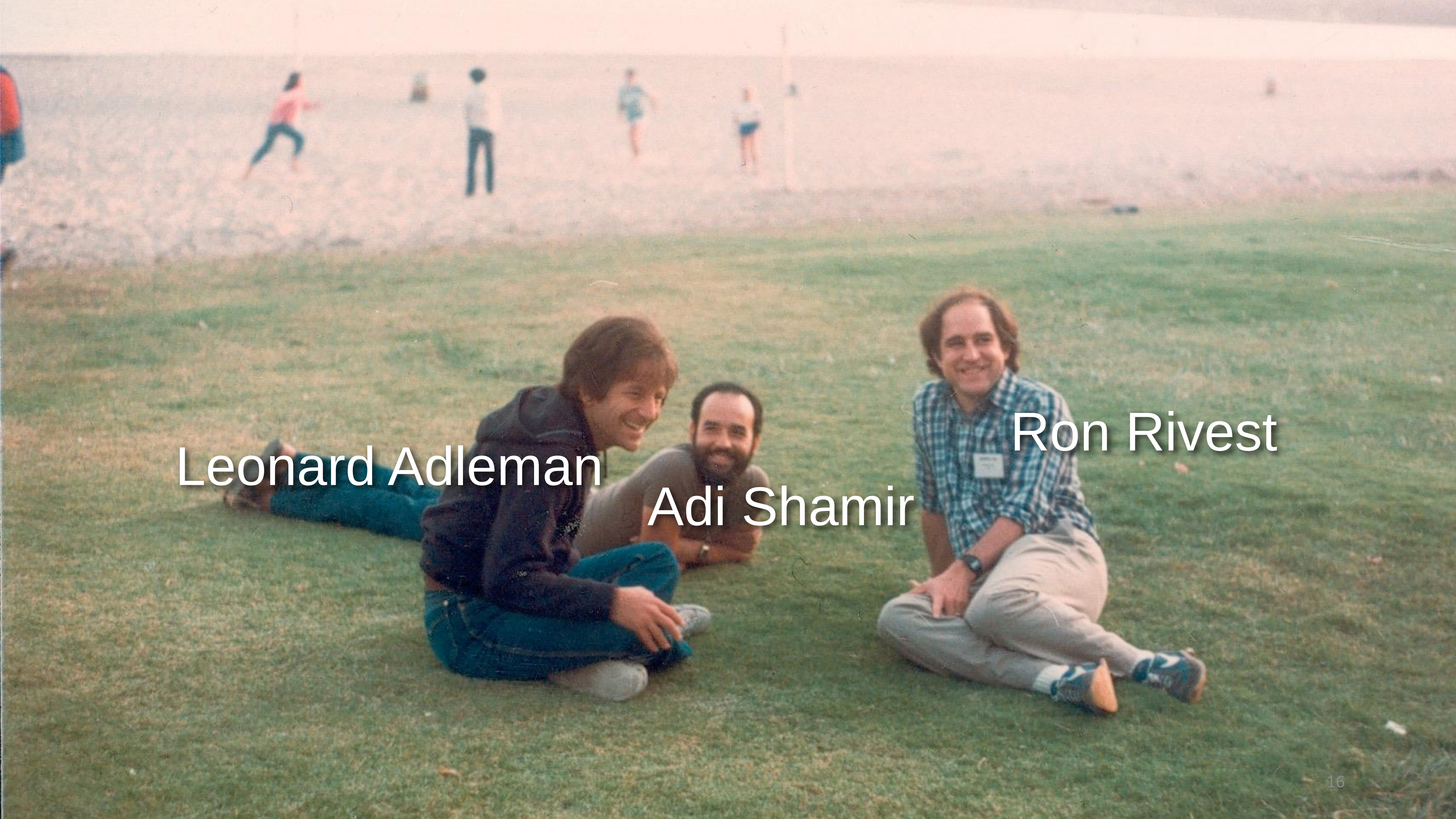
✓Man-In-The-Middle Attack

- RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- Hybrid Encryption
- Session Key
- Digital Signature
- Finding algorithms in internet browser certificates

Symmetric versus asymmetric algorithms

- Symmetric cryptography
  - Use a copied pair of symmetric (identical) secret keys
  - The sender and the receive use the same key for encryption and decryption functions
  - Confidentiality, but no integrity, authentication nor non-repudiation

- Asymmetric cryptography
  - Also know as “public key cryptography”
  - Use different (“asymmetric”) keys for encryption and decryption
  - One is called the “private key” and the other is the “public key”
  - Confidentiality, but also want authenticity and non-repudiation


Leonard Adleman
Adi Shamir
Ron Rivest

RSA Public Key Algorithm

- Most popular worldwide standard, that can be used for:
  - Asymmetric encryption/decryption
  - Key exchange (i.e. used to encrypt AES symmetric key)
  - Digital signatures
- In one direction, RSA provides:
  - Confidentiality through encryption
  - Authentication and non-repudiation through signature verification
- In the inverse direction, RSA provides:
  - Confidentiality through decryption
  - Authentication and non-repudiation through signature generation

RSA Public Key Algorithm

- Based on factoring large numbers into their prime numbers
  - A prime number is a positive whole number whose only factors (i.e. integer divisors) are 1 and the number itself
    - E.g. 2, 3, 5, 7, 11, 13, 17, 19, 23, …
  - Prime number factoring is
    - Easy when you know the result and one of the factors
      - 6,700,283 = 1889 * 3547
    - Difficult when you do not know the factors, and the result is large
      - 6,700,283 = prime1 * prime2

912,000,833,142,392,234,931,095,438,312,170,357,695,712,756,726,097,734,441,072,301,836,839,393,353,139,295,831,007,333,431,845,325,988,055,078,535,723,070,121,899,982,515,821,096,513,935,693,429,159,810,068,629,730,360,987,721,191,239,128,388,101,705,884,309,757,897,995,146,963,367,920,258,875,045,283,800,013,428,503,089,286,243,910,365,443,336,583,304,589,741,301,149,906,707,508,832,951,802,034,609,255,816,376,427,847,745,175,505,389,216,575,446,117,214,435,309,308,014,792,888,796,704,735,885,959,753,047,089,134,349,280,135,328,216,026,587,690,550,563,014,619,967,646,165,581,934,916,994,388,164,807,475,497,618,817,178,492,168,759,798,526,076,195,659,132,696,724,374,189,538,701,725,588,364,053,265,311,713,122,599,620,063,110,587,984,125,160,066,509,094,636,495,654,197,043,440,384,099,590,663,387,607,347,763,569,889,588,046,648,769,380,051,353,352,323,215,616,700,132,767,221,738,255,618,066,992,935,073,985,886,089,858,691,117,257,124,338,259,178,666,315,503,726,679,904,506,880,795,225,928,179,249,708,512,521,519,802,379,088,471,059,576,692,488,554,724,378,606,462,675,913,887,571,281,558,908,666,408,509,112,360,978,089,673,490,666,194,566,892,424,767,464,525,985,354,883,620,245,066,389,972,670,528,760,628,056,151,340,458,770,638,783,170,937,336,003,358,144,954,416,252,316,459,167,693,365,704,770,051,596,394,325,584,518,899,185,083,613,743,340,976,318,518,122,032,762,826,960,167,883,646,888,151,502,959,194,155,684,395,680,807,784,172,903,618,731,005,977,092,813,955,195,470,328,083,428,604,222,138,565,171,106,482,154,997,950,843,259,717,191,116,046,110,961,976,117,683,744,708,282,531,877,426,978,230,302,213,288,137,147

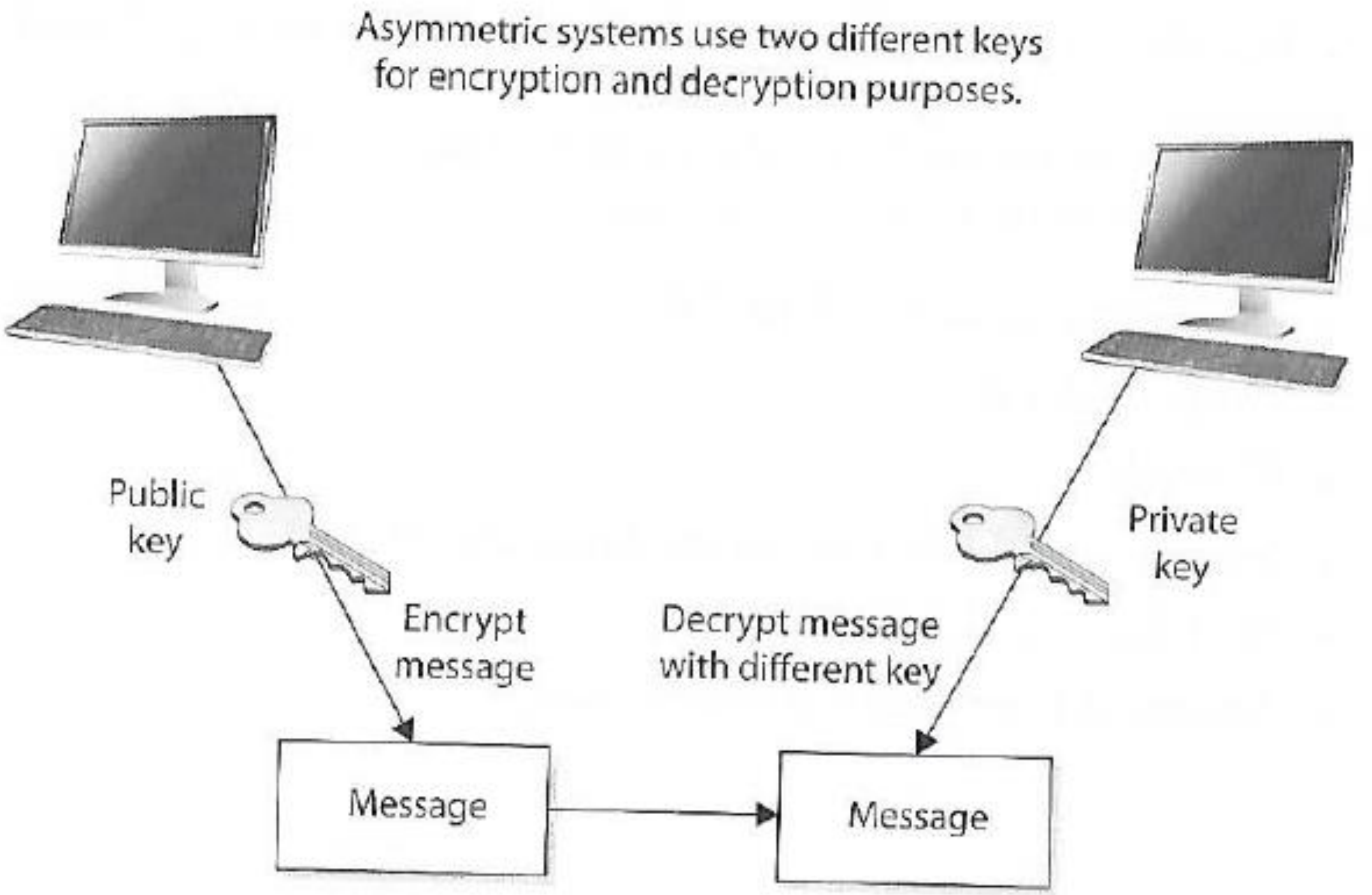
$\text{prime}_1 * \text{prime}_2 =$

912,000,833,142,392,234,931,095,438,312,
39,393,353,139,295,831,007,333,431,845,3
6,513,935,693,429,159,810,068,629,730,36
995,146,963,367,920,258,875,045,283,800,
89,741,301,149,906,707,508,832,951,802,0
5,446,117,214,435,309,308,014,792,888,79
216,026,587,690,550,563,014,619,967,646,
78,492,168,759,798,526,076,195,659,132,6
3,122,599,620,063,110,587,984,125,160,06
387,607,347,763,569,889,588,046,648,769,
55,618,066,992,935,073,985,886,089,858,6
4,506,880,795,225,928,179,249,708,512,52
606,462,675,913,887,571,281,558,908,666,
24,767,464,525,985,354,883,620,245,066,3
3,170,937,336,003,358,144,954,416,252,31
899,185,083,613,743,340,976,318,518,122,
55,684,395,680,807,784,172,903,618,731,0
8,565,171,106,482,154,997,950,843,259,71
877,426,978,230,302,213,288,137,147

Asymmetric cryptography

- **Public and Private** keys are mathematically related
  - Public keys are generated from private key
  - Private keys cannot be derived from the associated public key (if it falls into the wrong hands)

- **Public key** can be known by everyone
- **Private key** must be known and used only by the owner


Asymmetric systems use two different keys for encryption and decryption purposes.
Public key
Private key
Encrypt message
Decrypt message with different key
Message
Message

Asymmetric cryptography is computationally intensive and much slower (1,000 times slower) than symmetric cryptography

Harris, S. and Maymi, F. (2016) All-In-One CISSP Exam Guide

Asymmetric cryptography

- Do not get confused and think the public key is only for encryption and private key is only for decryption!

- Each key type can be use used to encrypt and decrypt
  - If data is encrypted with a private key it cannot be decrypted with the same private key (but it can be decrypted with the related public key)
  - If data is encrypted with a public key it cannot be decrypted with the same public key (but it can be decrypted with the related private key)

Asymmetric cryptography

If the sender (“Jill”) encrypts data with her private key, the receiver (“Bill”) must have a copy of Jill’s public key to decrypt it

- By decrypting the message with Jill’s public key Bill can be sure the message really came from Jill
- A message can be decrypted with a public key only if the message was encrypted with the corresponding private key
  - *This provides **authentication** because Jill is only the only one who is supposed to have her private key*

If Bill (the receiver) wants to make sure Jill is the only one who can read his reply, he will encrypt the response with her public key

- *Only Jill will be able to decrypt the message, because she is the only one who has the necessary private key*
- *This provides **confidentiality** because only Jill is able to decrypt the message with her private key*

Asymmetric cryptography

Why would Bill (now the sender) choose to encrypt his reply to Jill with his private key instead of using Jill’s public key?

- **Authentication** – Bill wants Jill to know that the message came from him and no one else
- If he encrypted the data with Jill’s public key, it does not provide authenticity because anyone can get Jill’s public key
- If he uses his private key to encrypt the data, then Jill can be sure the message came from him and no one else

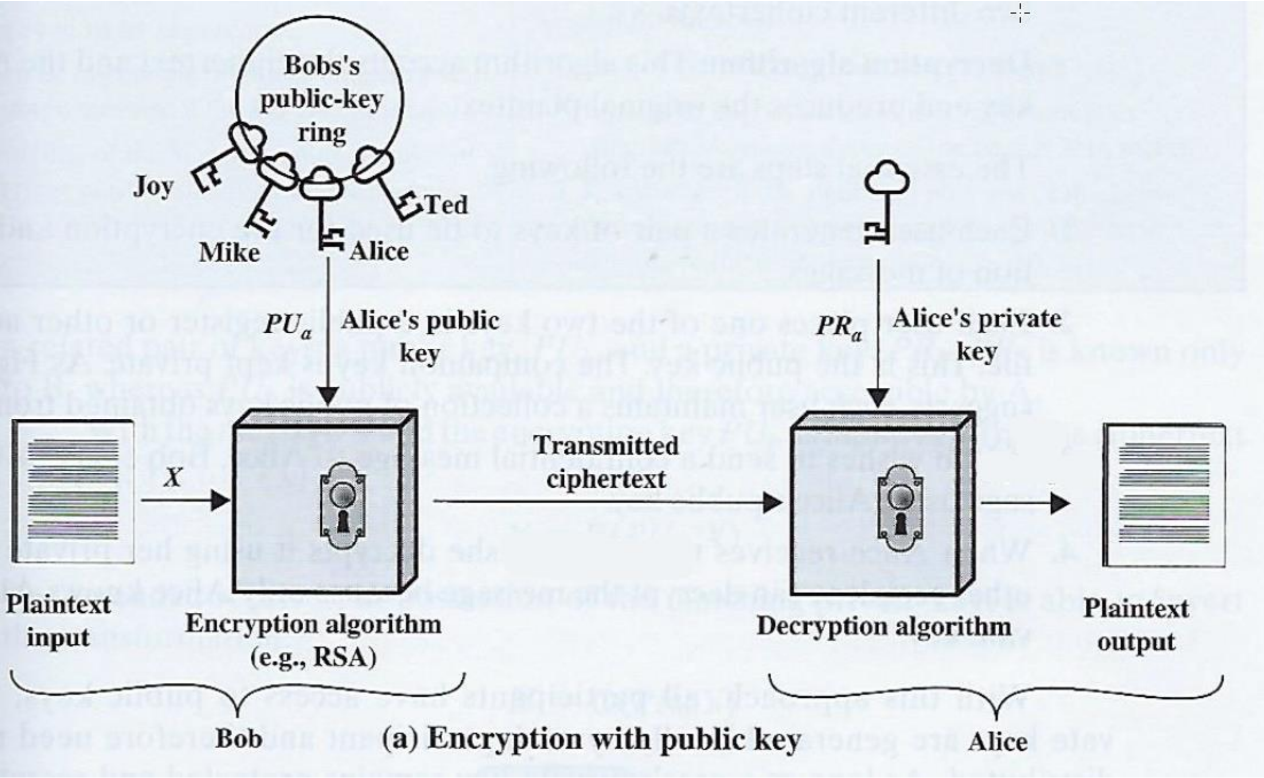
Note: *Symmetric keys do not provide authenticity – because the same key is used on both ends (using one of the secret keys does not ensure the message originated from a specific individual*

Asymmetric cryptography

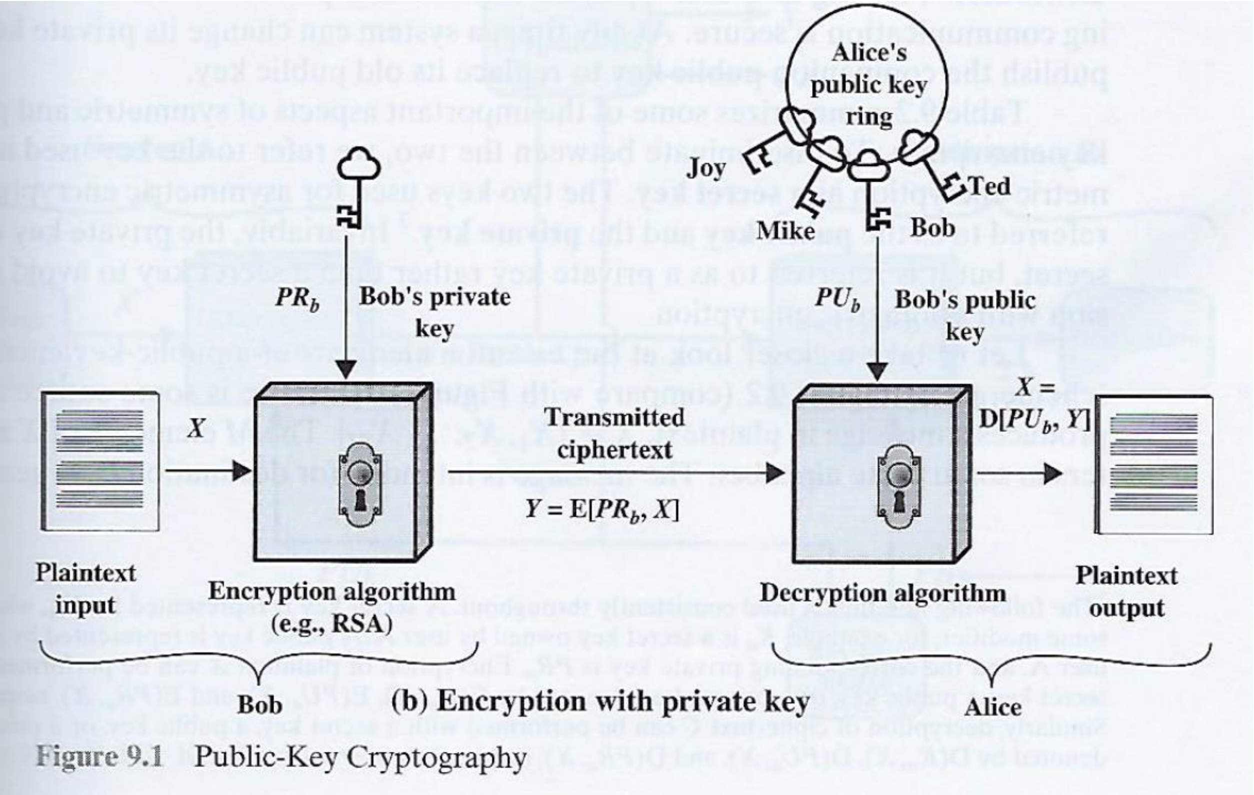
- If **confidentiality** is the most important security service, the sender would encrypt the file with the receiver's public key
  - This is called a "**secure message format**" because it can only be decrypted by the person with the corresponding private key

- If **authentication** is most important, the sender would encrypt the data with her/his private key
  - This provides assurance to the receiver that the only person who could have encrypted the data is the individual in possession of the private key
  - If the sender encrypted the data with receiver's public key, authentication is not provided because the public key is available to anyone
  - Encrypting data with the sender's private key is called an "**open message format**" because anyone with a copy of the corresponding public key can decrypt the message
  - <u>Confidentiality is not assured</u>

Public Key Management



(a) Encryption with public key



(b) Encryption with private key

Figure 9.1 Public-Key Cryptography

Stallings, W. (2014) Cryptography and Network Security

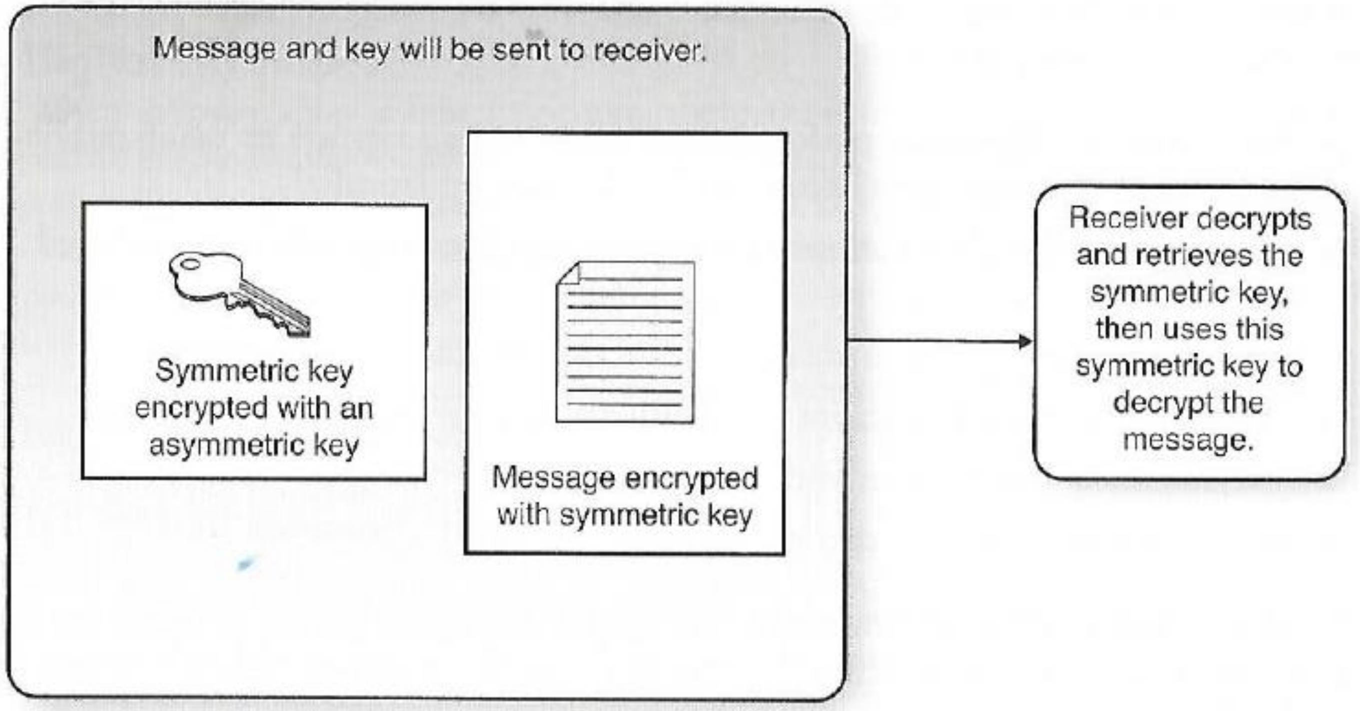
Agenda

- ✓Key sharing problem: Metcalf’s Law
- ✓Symmetric key sharing with Diffie-Hellman public key algorithm
- ✓Man-In-The-Middle Attack
- ✓RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- Hybrid Encryption
- Session key
- Digital Signature
- Finding algorithms in internet browser certificates

Hybrid Encryption (a.k.a. “digital envelope”)

Symmetric and asymmetric and algorithms are often used together

- Public key cryptography’s asymmetric algorithm is used to create public and private keys for secure automated key distribution
- Symmetric algorithm is used to create secret keys for rapid encryption/decryption of bulk data



Message and key will be sent to receiver.

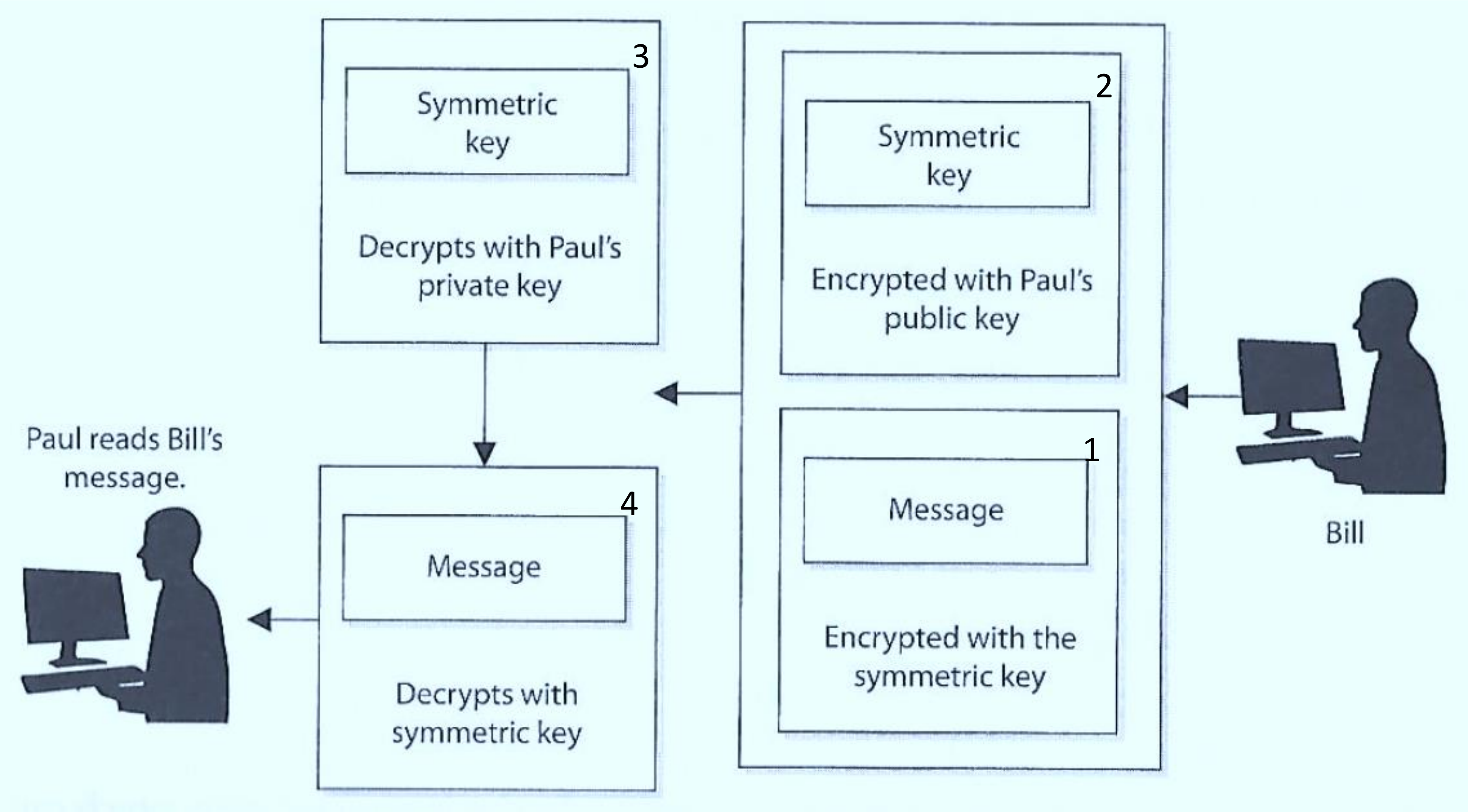
Symmetric key encrypted with an asymmetric key

Message encrypted with symmetric key

Receiver decrypts and retrieves the symmetric key, then uses this symmetric key to decrypt the message.

Harris, S. and Maymi, F. (2016) All-In-One CISSP Exam Guide, McGraw Hill Education

Hybrid Encryption


3
Symmetric key
Decrypts with Paul's private key
2
Symmetric key
Encrypted with Paul's public key
1
Message
Encrypted with the symmetric key
Bill
Paul reads Bill's message.
4
Message
Decrypts with symmetric key

Symmetric algorithm uses a secret key to encrypt the message and the asymmetric key encrypts the secret key for transmission (SSL/TLS uses hybrid)

Harris, S. and Maymi, F. (2016) All-In-One CISSP Exam Guide, McGraw Hill Education

Quick review

1. If a symmetric key is encrypted with a receiver’s public key, what security service is provided?
    - **Confidentiality**: only the receiver’s private key can be used to decrypt the symmetric key, and only the receiver should have access to this private key

Quick review

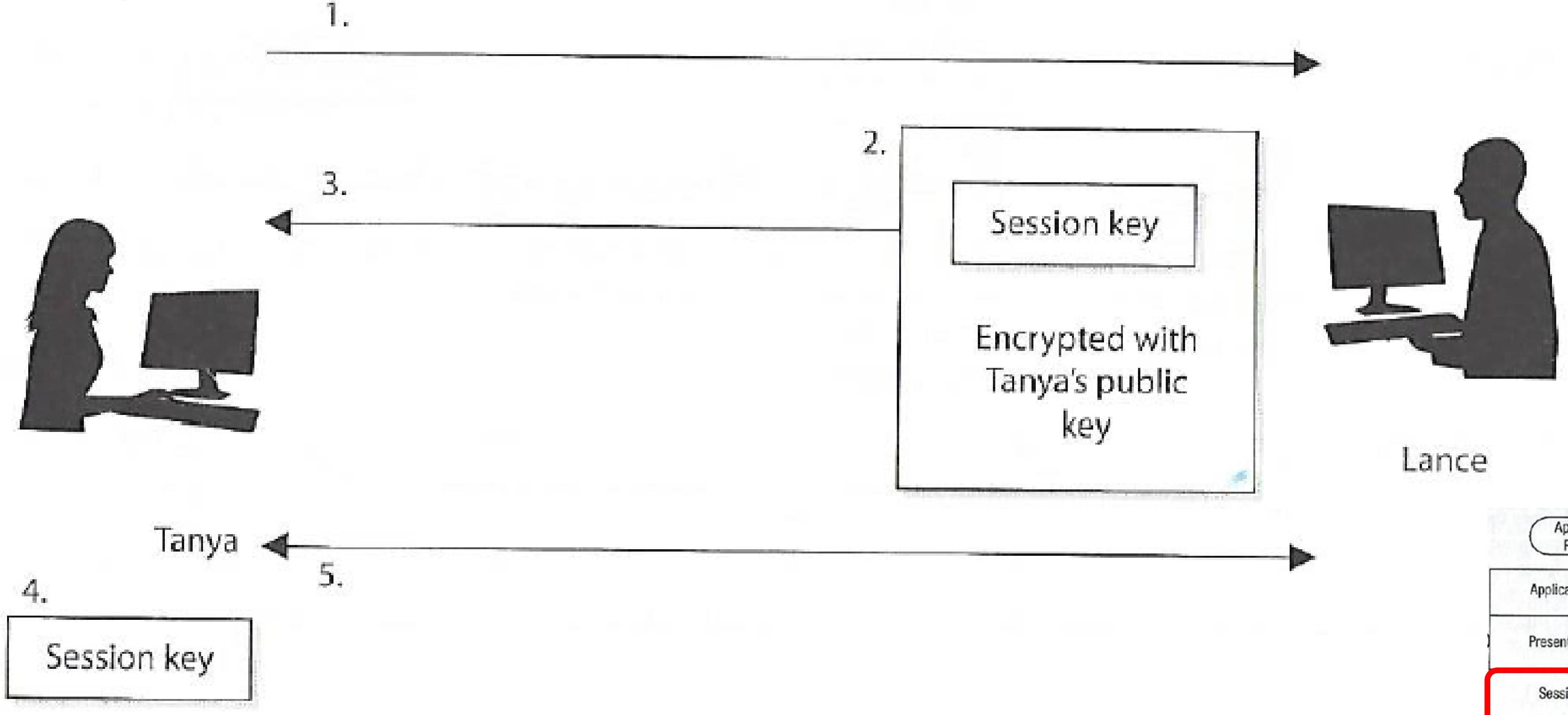
2. If data is encrypted with the sender’s private key, what security services are provided?
    - **Authenticity** of the sender and nonrepudiation. If the receiver can decrypt the encrypted data with the sender’s public key, then receiver knows the data was encrypted with the sender’s private key

Quick review

3. Why do we encrypt the message with the symmetric key rather than the asymmetric key?
   - **Because the asymmetric key algorithm is too slow**

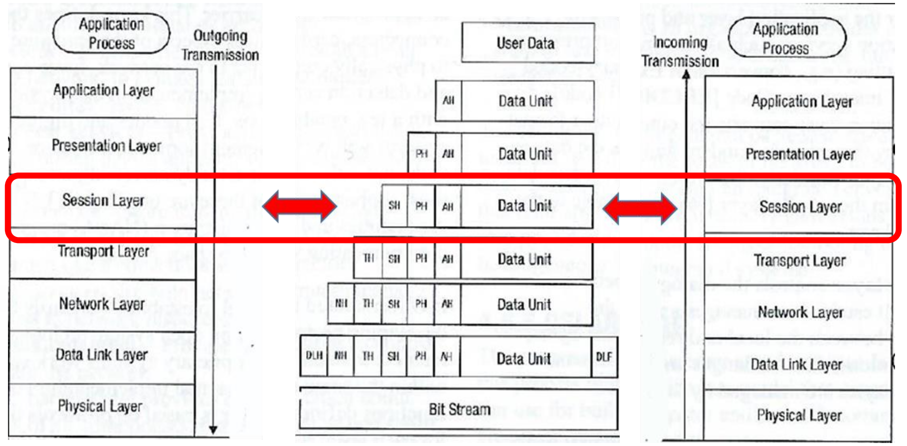
Session keys

<u>Single-use</u> symmetric keys used to encrypt messages between two users in an individual communication session



1) Tanya sends Lance her public key.
2) Lance generates a random session key and encrypts it using Tanya's public key.
3) Lance sends the session key, encrypted with Tanya's public key, to Tanya.
4) Tanya decrypts Lance's message with her private key and now has a copy of the session key.
5) Tanya and Lance use this session key to encrypt and decrypt messages to each other.

This is how secure web client applications communicate with server-side services



Harris, S. and Maymi, F. (2016) <u>All-In-One CISSP Exam Guide</u>, McGraw Hill Education

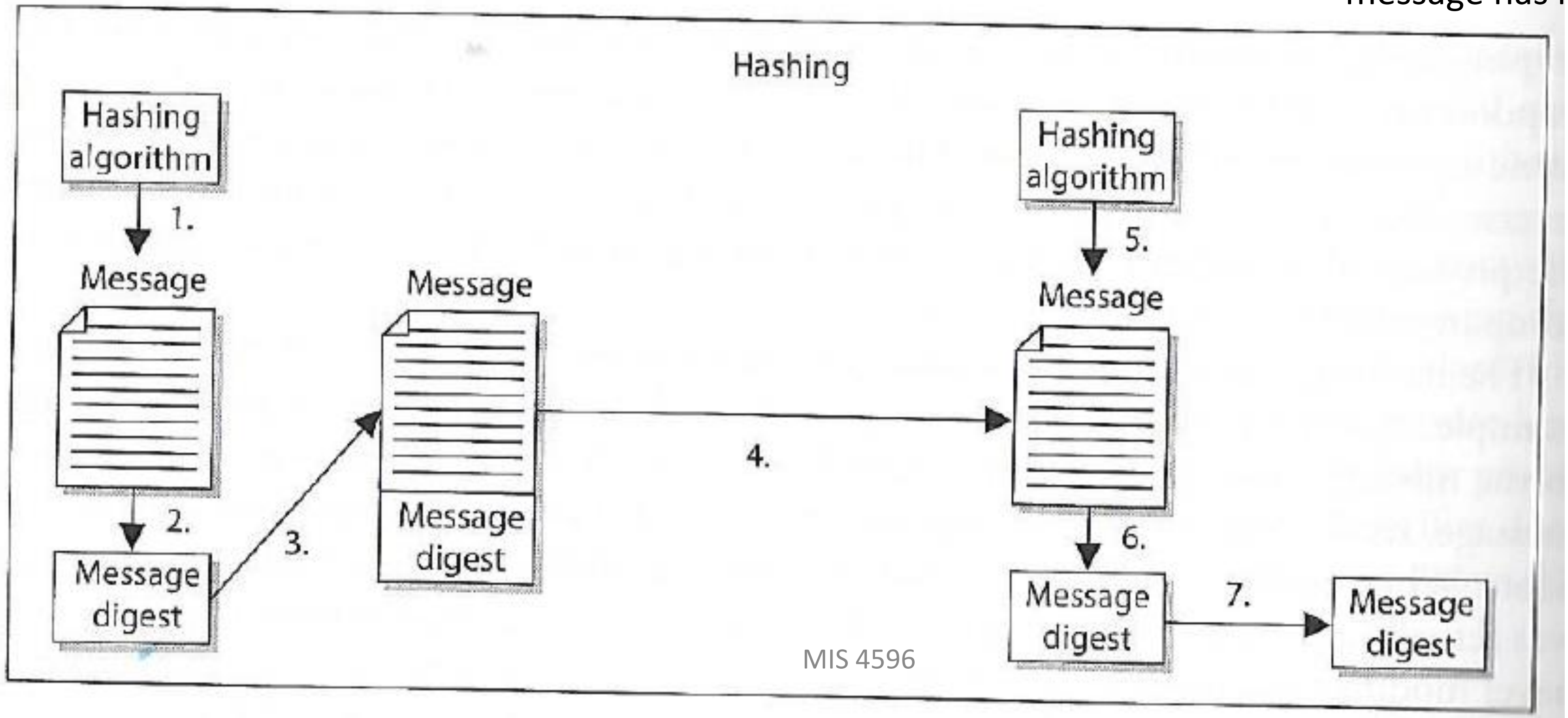
Agenda

- ✓Key sharing problem: Metcalf’s Law
- ✓Symmetric key sharing with Diffie-Hellman public key algorithm
- ✓Man-In-The-Middle Attack
- ✓RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- ✓Hybrid Encryption
- ✓Session key
- Digital Signature
- Finding algorithms in internet browser certificates

Quick Review: One-way Hash

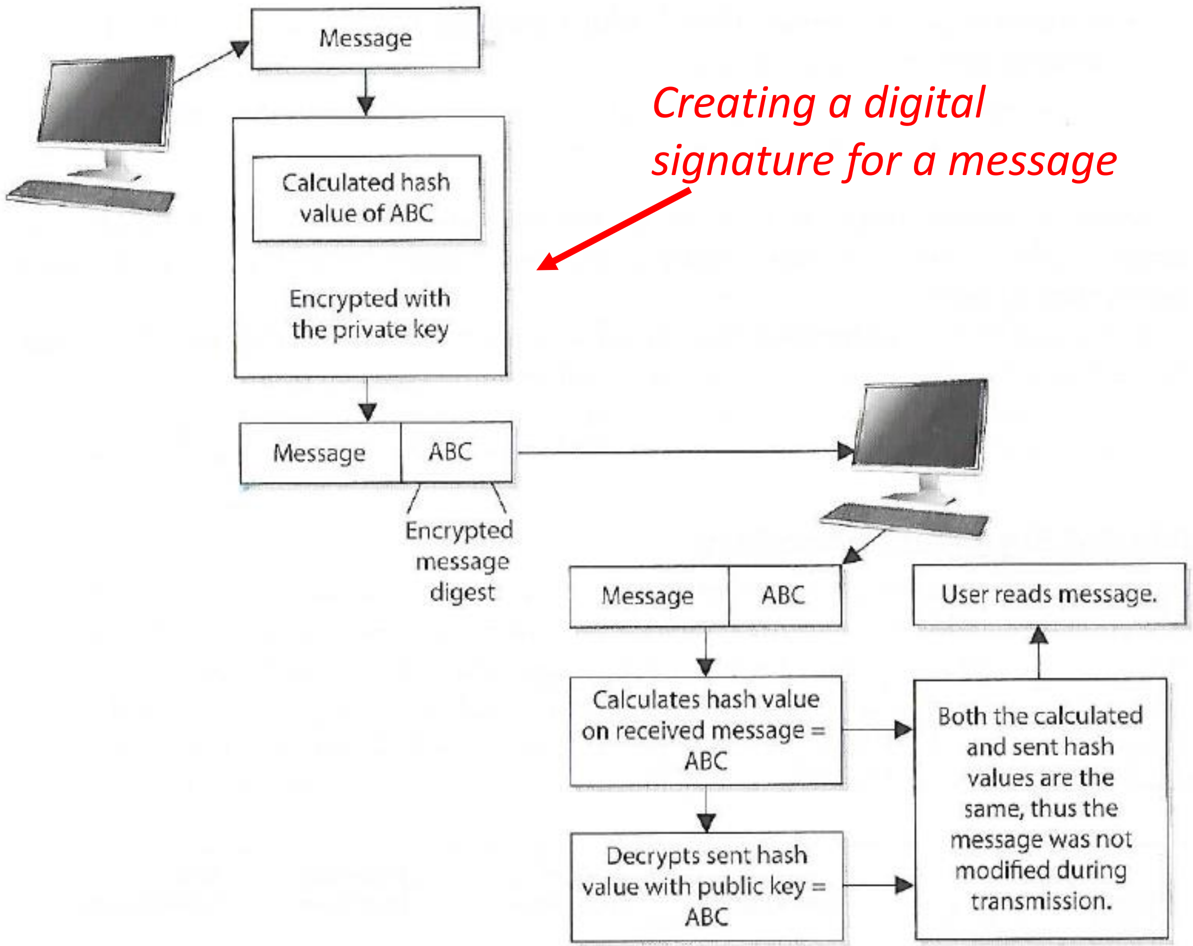
- Assures message **integrity**
- A function that takes a variable-length string (i.e. message) and produces a fixed-length value called a hash value
- Does not use keys

1. Sender puts message through hashing function
2. Message digest generated
3. Message digest appended to the message
4. Sender sends message to receiver
5. Receiver puts message through hashing function
6. Receiver generates message digest value
7. Receiver compares the two message digests values. If they are the same, the message has not been altered



Digital Signature

- A hash value encrypted with the sender’s private key
- The act of signing means encrypting the message’s hash value with the private key


Message
Calculated hash value of ABC
Encrypted with the private key
Message
ABC
Encrypted message digest
Message
ABC
User reads message.
Calculates hash value on received message = ABC
Both the calculated and sent hash values are the same, thus the message was not modified during transmission.
Decrypts sent hash value with public key = ABC

Creating a digital signature for a message

Harris, S. and Maymi, F. (2016) All-In-One CISSP Exam Guide, McGraw Hill Education

Reasons to Use Cryptography

Reason	How achieved
Confidentiality	The message can be encrypted
Integrity	The message can be hashed and/or digitally signed
Authentication	The message can be digitally signed
Nonrepudiation	The message can be digitally signed

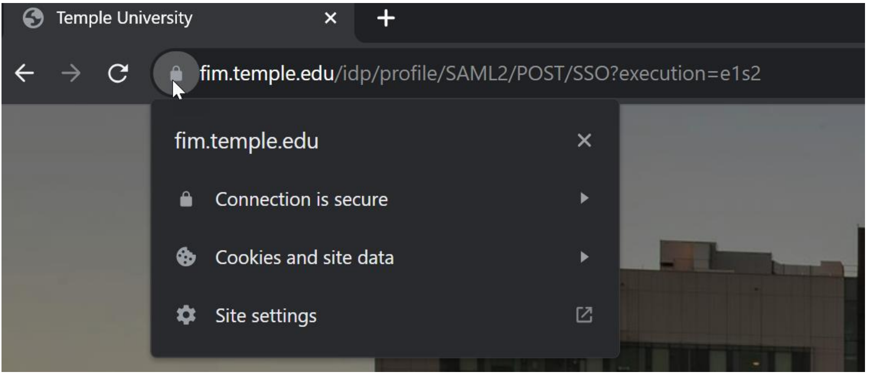
Agenda

- ✓Key sharing problem: Metcalf’s Law
- ✓Symmetric key sharing with Diffie-Hellman public key algorithm
- ✓Man-In-The-Middle Attack
- ✓RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- ✓Hybrid Encryption
- ✓Session key
- ✓Digital Signature
- Finding algorithms in internet browser certificates

Viewing Digital Certificates to see algorithms used

Chrome Browser

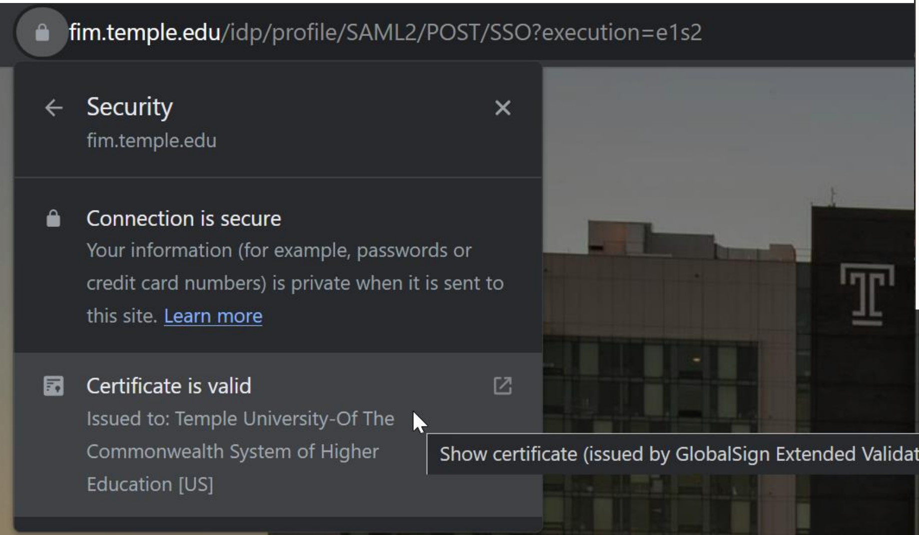
1. Click the padlock in the URL bar
2. Click the "Connection is secure"
3. Click "Certificate is valid"



Temple University

fim.temple.edu/idp/profile/SAML2/POST/SSO?execution=e1s2

fim.temple.edu



fim.temple.edu/idp/profile/SAML2/POST/SSO?execution=e1s2

Security

fim.temple.edu

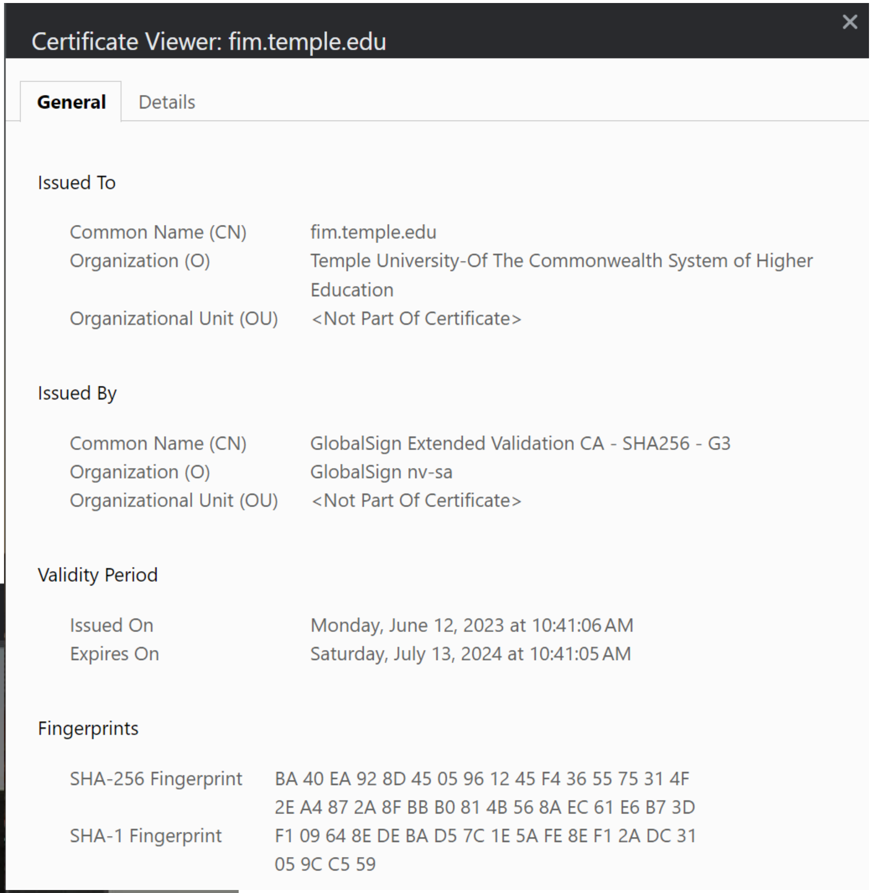
Connection is secure

Your information (for example, passwords or credit card numbers) is private when it is sent to this site. Learn more

Certificate is valid

Issued to: Temple University-Of The Commonwealth System of Higher Education [US]

Show certificate (issued by GlobalSign Extended Validation CA - SHA256 - G3)



Certificate Viewer: fim.temple.edu

Issued To

Common Name (CN): fim.temple.edu

Organization (O): Temple University-Of The Commonwealth System of Higher Education

Organizational Unit (OU): <Not Part Of Certificate>

Issued By

Common Name (CN): GlobalSign Extended Validation CA - SHA256 - G3

Organization (O): GlobalSign nv-sa

Organizational Unit (OU): <Not Part Of Certificate>

Validity Period

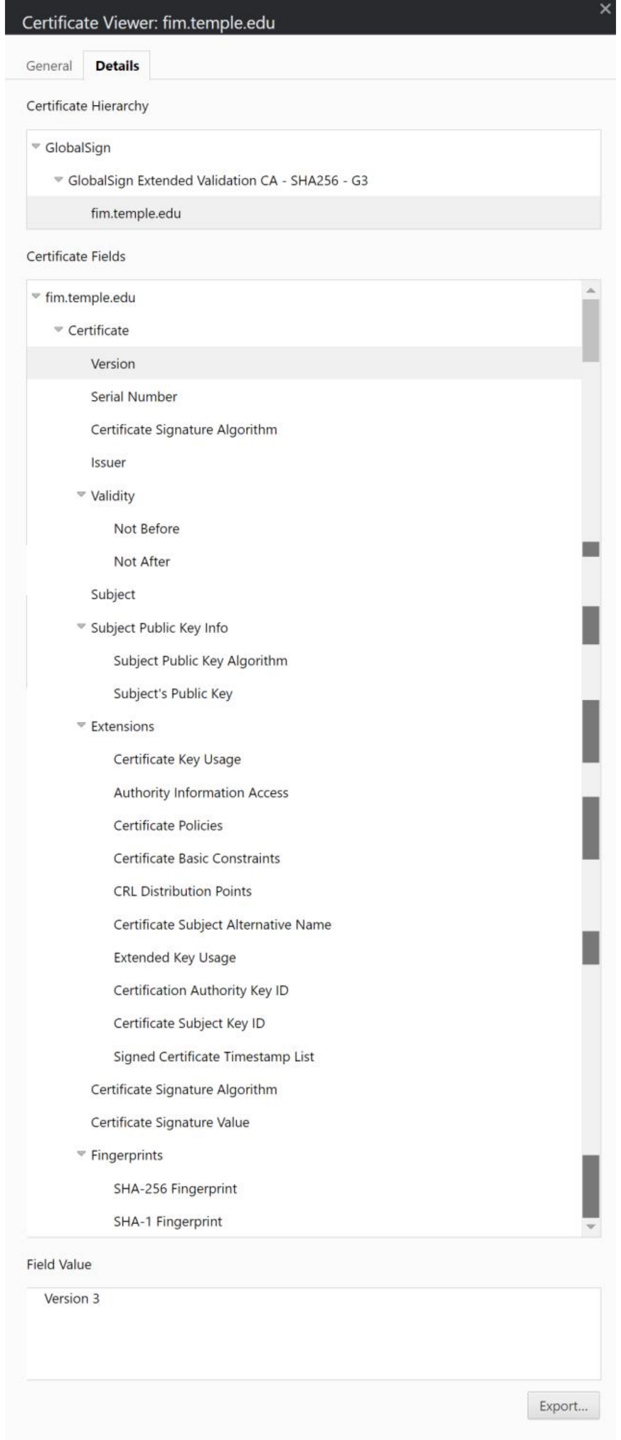
Issued On: Monday, June 12, 2023 at 10:41:06 AM

Expires On: Saturday, July 13, 2024 at 10:41:05 AM

Fingerprints

SHA-256 Fingerprint: BA 40 EA 92 8D 45 05 96 12 45 F4 36 55 75 31 4F 2E A4 87 2A 8F BB B0 81 4B 56 8A EC 61 E6 B7 3D

SHA-1 Fingerprint: F1 09 64 8E DE BA D5 7C 1E 5A FE 8E F1 2A DC 31 05 9C C5 59



Certificate Viewer: fim.temple.edu

Certificate Hierarchy

```
GlobalSign
  GlobalSign Extended Validation CA - SHA256 - G3
    fim.temple.edu
```

Certificate Fields

```
fim.temple.edu
  Certificate
    Version
    Serial Number
    Certificate Signature Algorithm
    Issuer
    Validity
      Not Before
      Not After
    Subject
    Subject Public Key Info
      Subject Public Key Algorithm
      Subject's Public Key
    Extensions
      Certificate Key Usage
      Authority Information Access
      Certificate Policies
      Certificate Basic Constraints
      CRL Distribution Points
      Certificate Subject Alternative Name
      Extended Key Usage
      Certification Authority Key ID
      Certificate Subject Key ID
      Signed Certificate Timestamp List
    Certificate Signature Algorithm
    Certificate Signature Value
    Fingerprints
      SHA-256 Fingerprint
      SHA-1 Fingerprint
```

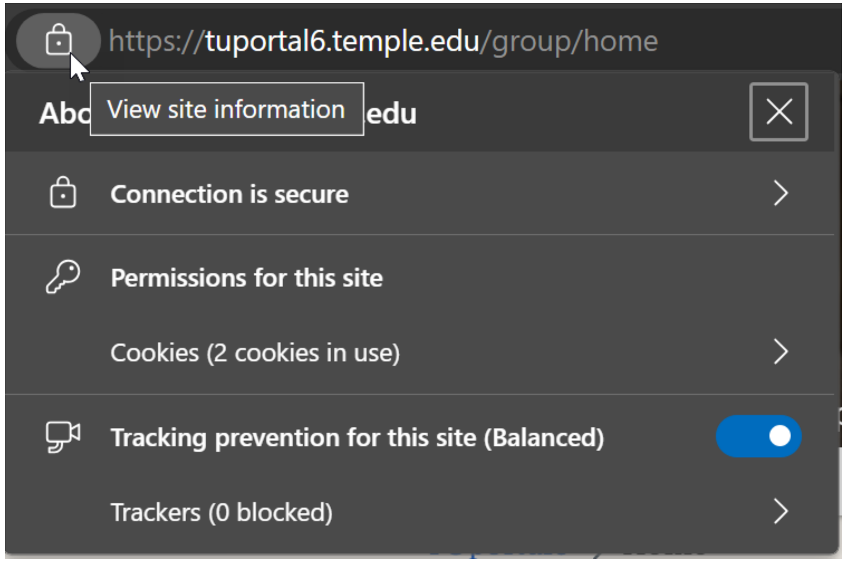
Field Value

Version 3

Viewing Digital Certificate

Microsoft Edge

1. Click the padlock on the URL bar
2. Click Connection is secure
3. Click Certificate icon…



https://tuportal6.temple.edu/group/home

Abc View site information edu

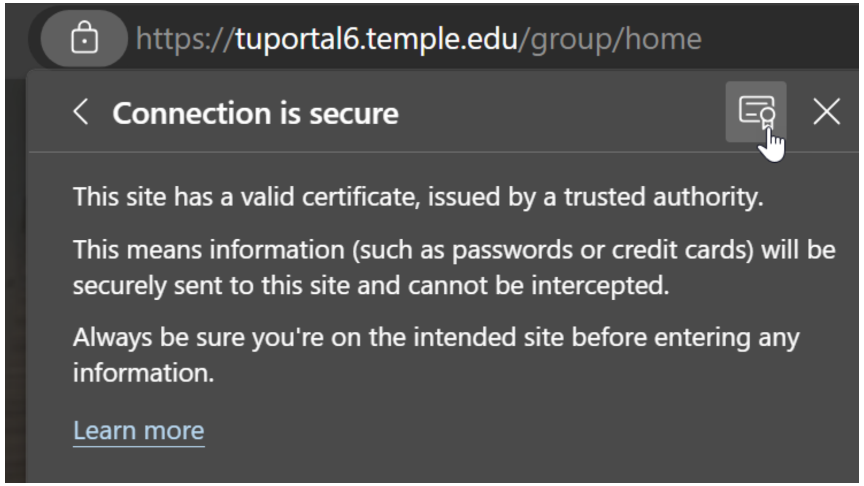
Connection is secure

Permissions for this site

Cookies (2 cookies in use)

Tracking prevention for this site (Balanced)

Trackers (0 blocked)



https://tuportal6.temple.edu/group/home

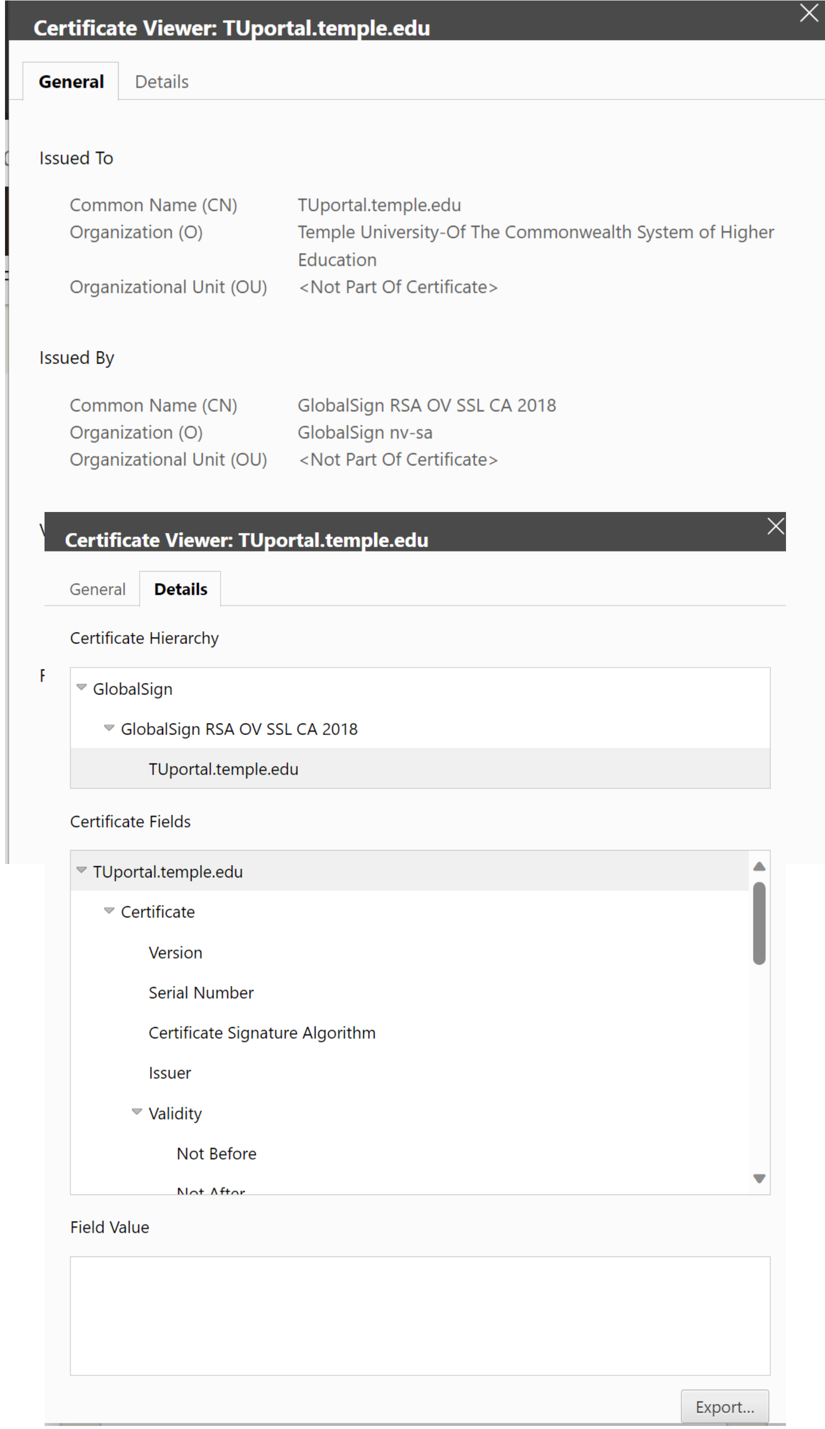
Connection is secure

This site has a valid certificate, issued by a trusted authority.

This means information (such as passwords or credit cards) will be securely sent to this site and cannot be intercepted.

Always be sure you're on the intended site before entering any information.

Learn more



Certificate Viewer: TUportal.temple.edu

General | Details

Issued To

Common Name (CN)	TUportal.temple.edu
Organization (O)	Temple University-Of The Commonwealth System of Higher Education
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	GlobalSign RSA OV SSL CA 2018
Organization (O)	GlobalSign nv-sa
Organizational Unit (OU)	<Not Part Of Certificate>

Certificate Viewer: TUportal.temple.edu

General | Details

Certificate Hierarchy

- GlobalSign
  - GlobalSign RSA OV SSL CA 2018
    - TUportal.temple.edu

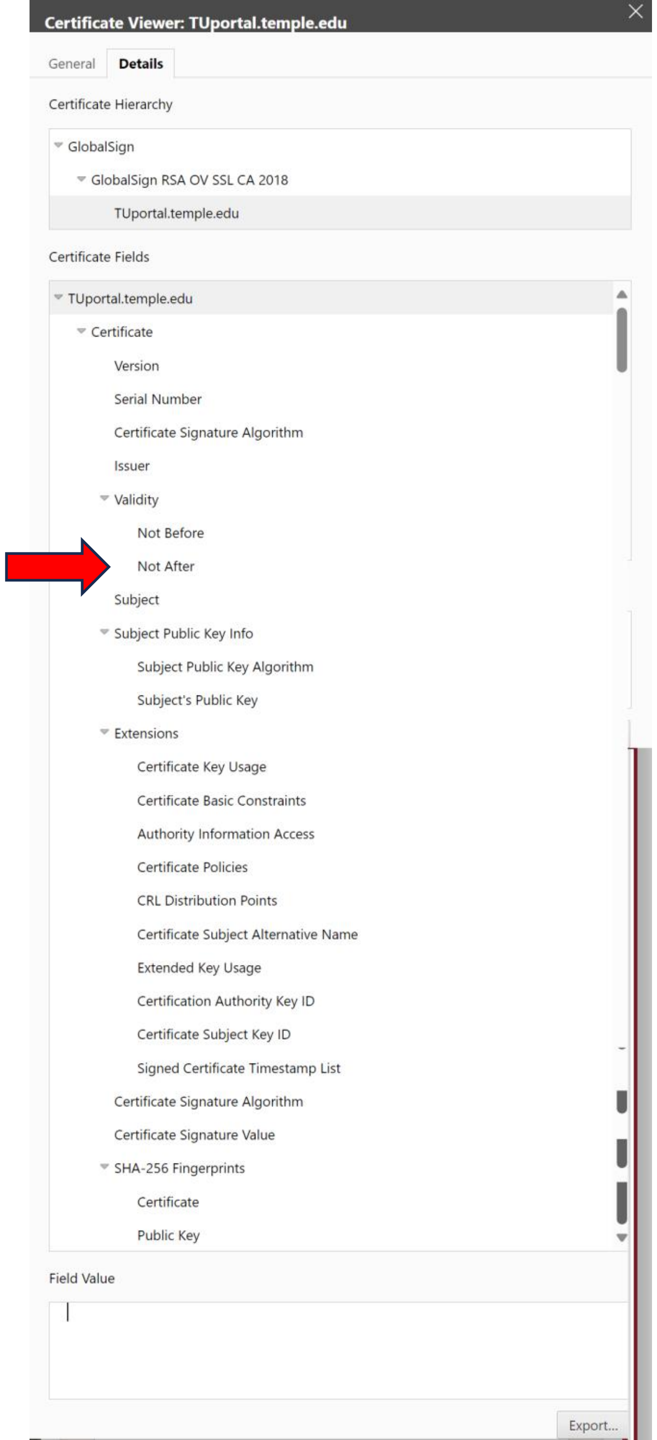
Certificate Fields

- TUportal.temple.edu
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
      - Not Before
      - Not After

Field Value

Export...

Certificate Details



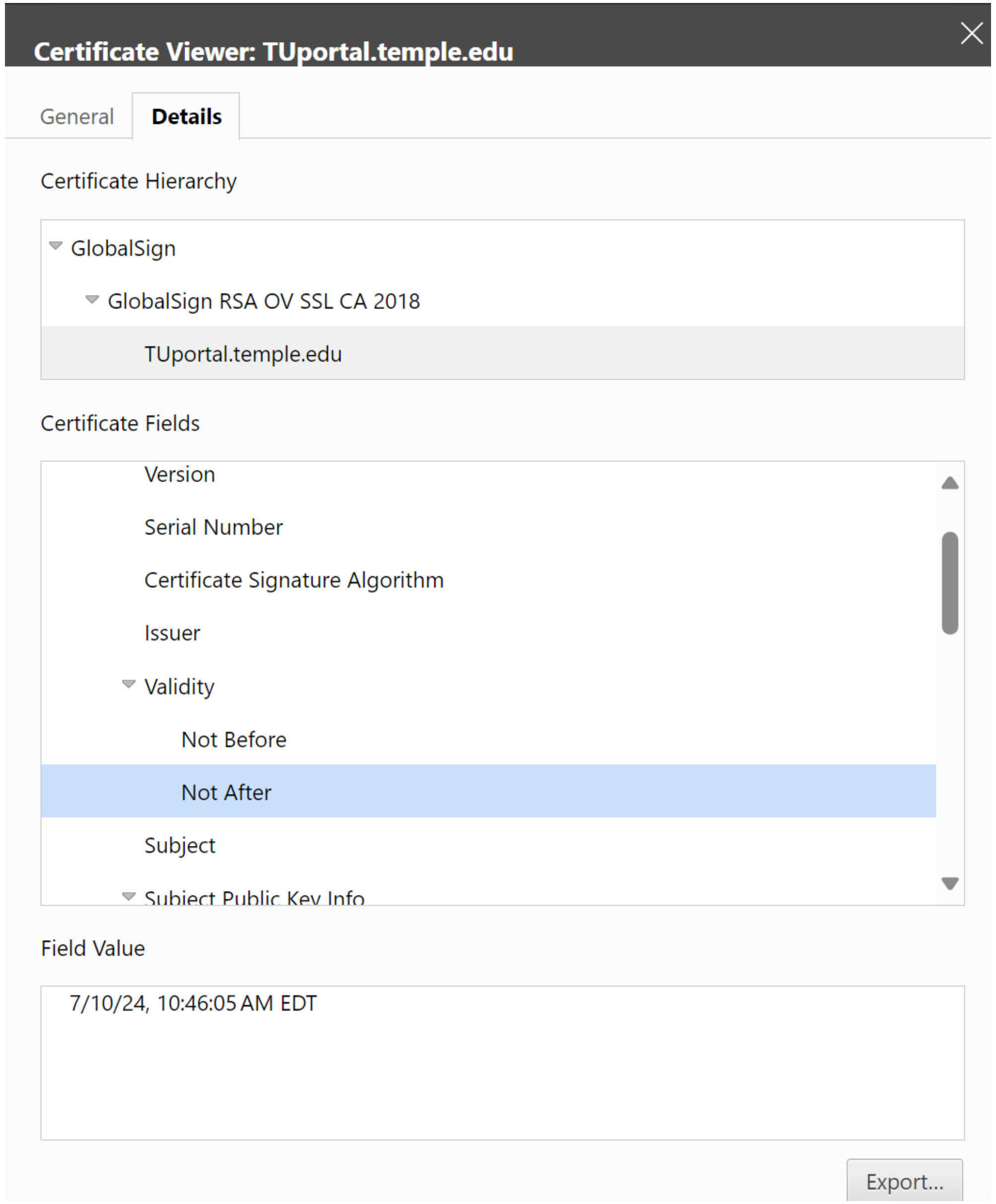
Certificate Hierarchy

- GlobalSign
  - GlobalSign RSA OV SSL CA 2018
    - TUportal.temple.edu

Certificate Fields

- TUportal.temple.edu
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
      - Not Before
      - Not After
    - Subject
    - Subject Public Key Info
      - Subject Public Key Algorithm
      - Subject's Public Key
    - Extensions
      - Certificate Key Usage
      - Certificate Basic Constraints
      - Authority Information Access
      - Certificate Policies
      - CRL Distribution Points
      - Certificate Subject Alternative Name
      - Extended Key Usage
      - Certification Authority Key ID
      - Certificate Subject Key ID
      - Signed Certificate Timestamp List
  - Certificate Signature Algorithm
  - Certificate Signature Value
  - SHA-256 Fingerprints
    - Certificate
    - Public Key

Field Value



Certificate Hierarchy

- GlobalSign
  - GlobalSign RSA OV SSL CA 2018
    - TUportal.temple.edu

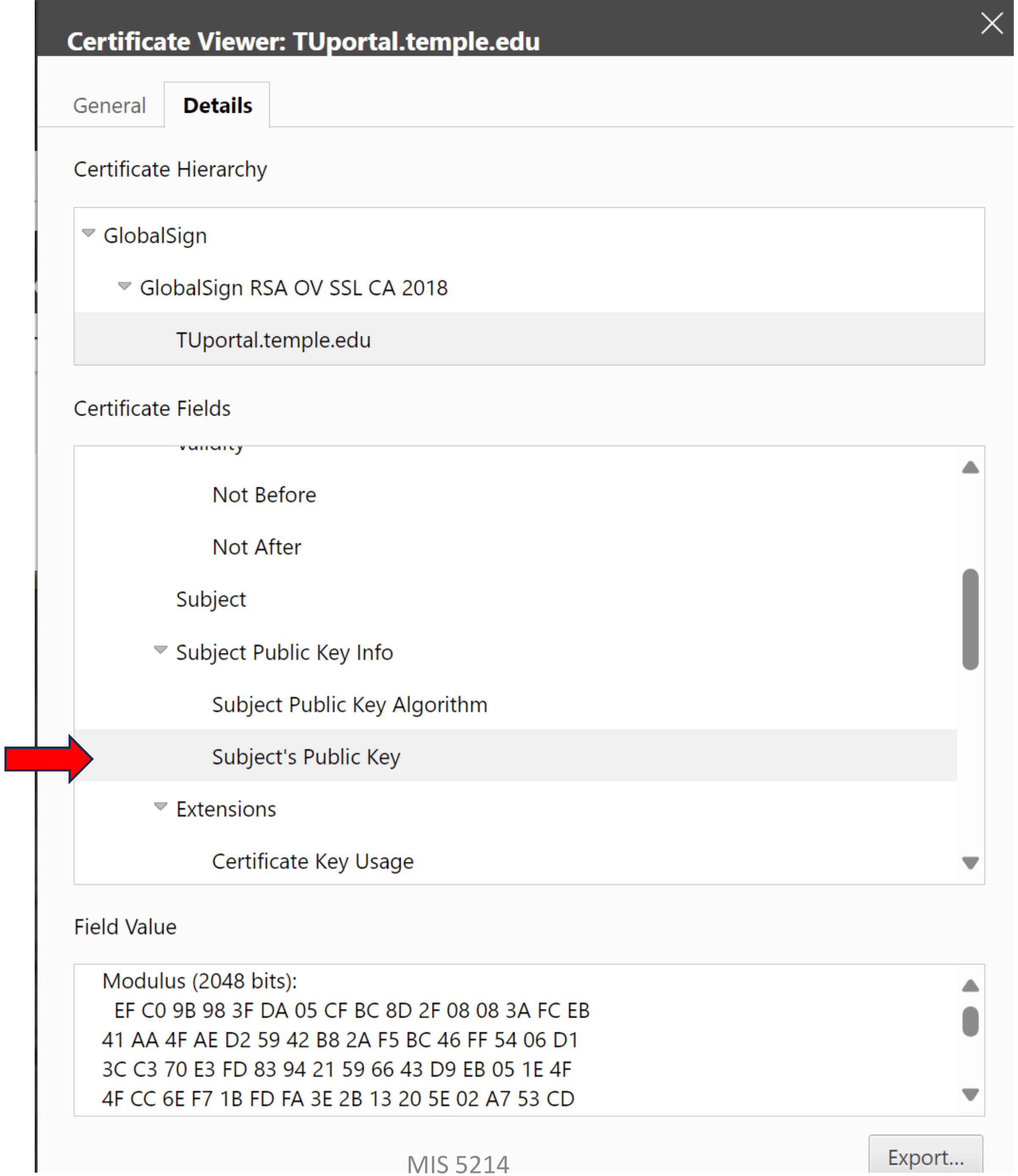
Certificate Fields

- Version
- Serial Number
- Certificate Signature Algorithm
- Issuer
- Validity
  - Not Before
  - Not After
- Subject
- Subject Public Key Info

Field Value

7/10/24, 10:46:05 AM EDT

Certificate Details



Certificate Viewer: TUportal.temple.edu

Certificate Hierarchy

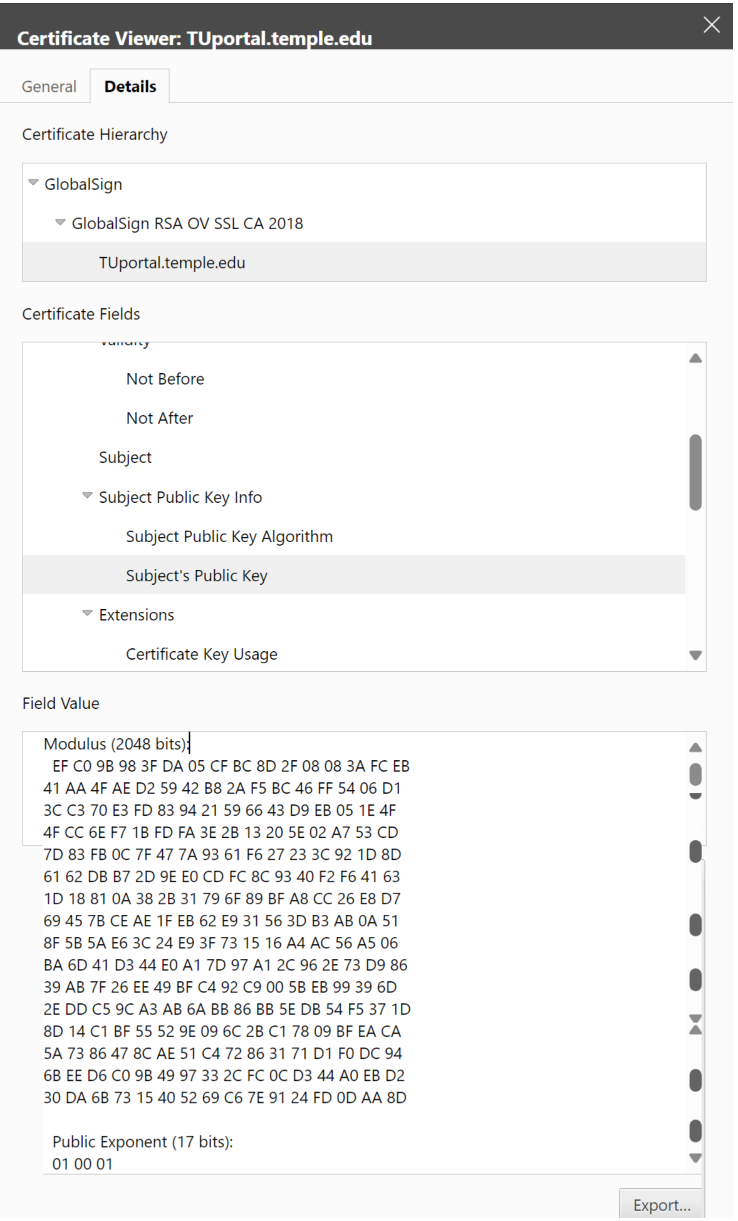
- GlobalSign
  - GlobalSign RSA OV SSL CA 2018
    - TUportal.temple.edu

Certificate Fields

- Validity
  - Not Before
  - Not After
- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- Extensions
  - Certificate Key Usage

Field Value

```
Modulus (2048 bits):
  EF C0 9B 98 3F DA 05 CF BC 8D 2F 08 08 3A FC EB
41 AA 4F AE D2 59 42 B8 2A F5 BC 46 FF 54 06 D1
3C C3 70 E3 FD 83 94 21 59 66 43 D9 EB 05 1E 4F
4F CC 6E F7 1B FD FA 3E 2B 13 20 5E 02 A7 53 CD
```



Certificate Viewer: TUportal.temple.edu

Certificate Hierarchy

- GlobalSign
  - GlobalSign RSA OV SSL CA 2018
    - TUportal.temple.edu

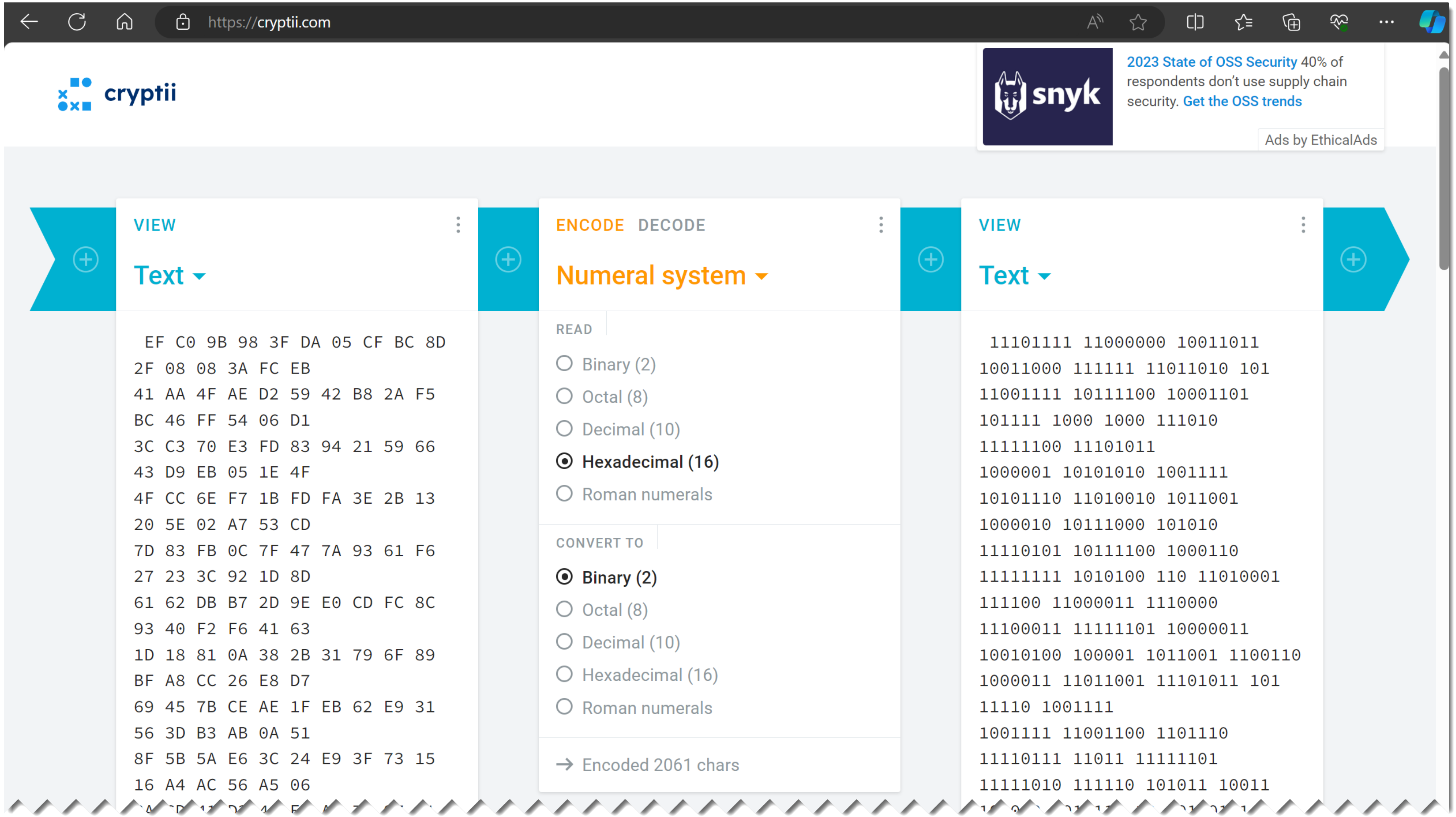
Certificate Fields

- Not Before
- Not After
- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- Extensions
  - Certificate Key Usage

Field Value

```
Modulus (2048 bits):
  EF C0 9B 98 3F DA 05 CF BC 8D 2F 08 08 3A FC EB
41 AA 4F AE D2 59 42 B8 2A F5 BC 46 FF 54 06 D1
3C C3 70 E3 FD 83 94 21 59 66 43 D9 EB 05 1E 4F
4F CC 6E F7 1B FD FA 3E 2B 13 20 5E 02 A7 53 CD
7D 83 FB 0C 7F 47 7A 93 61 F6 27 23 3C 92 1D 8D
61 62 DB B7 2D 9E E0 CD FC 8C 93 40 F2 F6 41 63
1D 18 81 0A 38 2B 31 79 6F 89 BF A8 CC 26 E8 D7
69 45 7B CE AE 1F EB 62 E9 31 56 3D B3 AB 0A 51
8F 5B 5A E6 3C 24 E9 3F 73 15 16 A4 AC 56 A5 06
BA 6D 41 D3 44 E0 A1 7D 97 A1 2C 96 2E 73 D9 86
39 AB 7F 26 EE 49 BF C4 92 C9 00 5B EB 99 39 6D
2E DD C5 9C A3 AB 6A BB 86 BB 5E DB 54 F5 37 1D
8D 14 C1 BF 55 52 9E 09 6C 2B C1 78 09 BF EA CA
5A 73 86 47 8C AE 51 C4 72 86 31 71 D1 F0 DC 94
6B EE D6 C0 9B 49 97 33 2C FC 0C D3 44 A0 EB D2
30 DA 6B 73 15 40 52 69 C6 7E 91 24 FD 0D AA 8D

  Public Exponent (17 bits):
  01 00 01
```



VIEW
Text

```
 EF C0 9B 98 3F DA 05 CF BC 8D
2F 08 08 3A FC EB
41 AA 4F AE D2 59 42 B8 2A F5
BC 46 FF 54 06 D1
3C C3 70 E3 FD 83 94 21 59 66
43 D9 EB 05 1E 4F
4F CC 6E F7 1B FD FA 3E 2B 13
20 5E 02 A7 53 CD
7D 83 FB 0C 7F 47 7A 93 61 F6
27 23 3C 92 1D 8D
61 62 DB B7 2D 9E E0 CD FC 8C
93 40 F2 F6 41 63
1D 18 81 0A 38 2B 31 79 6F 89
BF A8 CC 26 E8 D7
69 45 7B CE AE 1F EB 62 E9 31
56 3D B3 AB 0A 51
8F 5B 5A E6 3C 24 E9 3F 73 15
16 A4 AC 56 A5 06
```

ENCODE DECODE
Numeral system

READ
○ Binary (2)
○ Octal (8)
○ Decimal (10)
● Hexadecimal (16)
○ Roman numerals

CONVERT TO
● Binary (2)
○ Octal (8)
○ Decimal (10)
○ Hexadecimal (16)
○ Roman numerals

→ Encoded 2061 chars

VIEW
Text

```
 11101111 11000000 10011011
10011000 111111 11011010 101
11001111 10111100 10001101
101111 1000 1000 111010
11111100 11101011
1000001 10101010 1001111
10101110 11010010 1011001
1000010 10111000 101010
11110101 10111100 1000110
11111111 1010100 110 11010001
111100 11000011 1110000
11100011 11111101 10000011
10010100 100001 1011001 1100110
1000011 11011001 11101011 101
11110 1001111
1001111 11001100 1101110
11110111 11011 11111101
11111010 111110 101011 10011
```

https://cryptii.com/

Agenda

- ✓Key sharing problem: Metcalf’s Law
- ✓Symmetric key sharing with Diffie-Hellman public key algorithm
- ✓Man-In-The-Middle Attack
- ✓RSA - Public Key Encryption with confidentiality, authentication and non-repudiation
- ✓Hybrid Encryption
- ✓Session key
- ✓Digital Signature
- ✓Finding algorithms in internet browser certificates