

# **ECOR 1055C**

## **Engineering Disciplines I**

Introduction to Security and Cryptography

**Mostafa Taha**

Assistant Professor Systems & Computer Engineering  
Carleton University

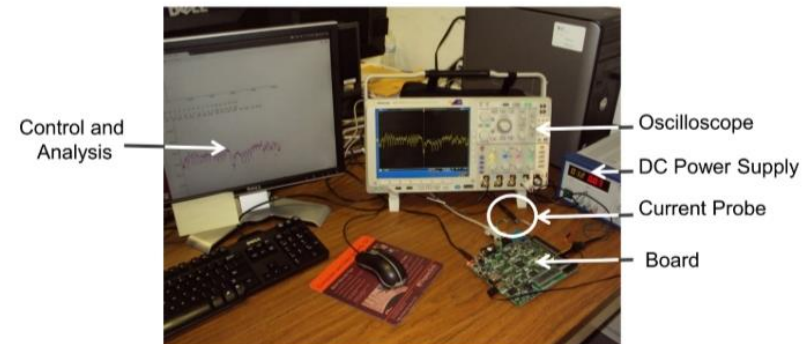
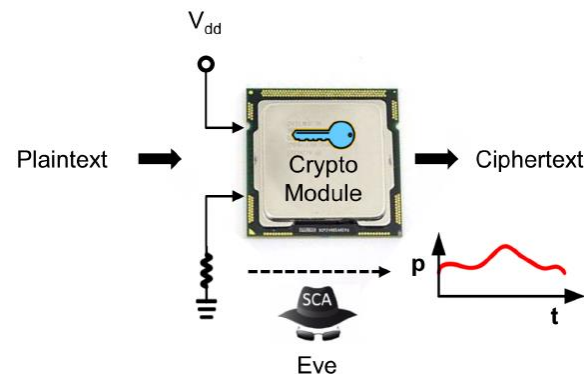
<https://carleton.ca/mtaha/>

# Lecture Outline

- About the instructor
- Introduction to Security and Cryptography
- Key Players in Cryptography
- User's Perspective
- Developer's Perspective

# About the instructor

- Mostafa Taha
  - Systems and Computer Engineering Department
  - Website: <https://carleton.ca/mtaha/> , Email: [mtaha@sce.carleton.ca](mailto:mtaha@sce.carleton.ca)
  - Currently teaching SYSC4810B and SYSC4805
- Research Areas:
  - Security of Embedded Systems and the Internet of Things.
  - Implementation of Security and Cryptographic Algorithms.

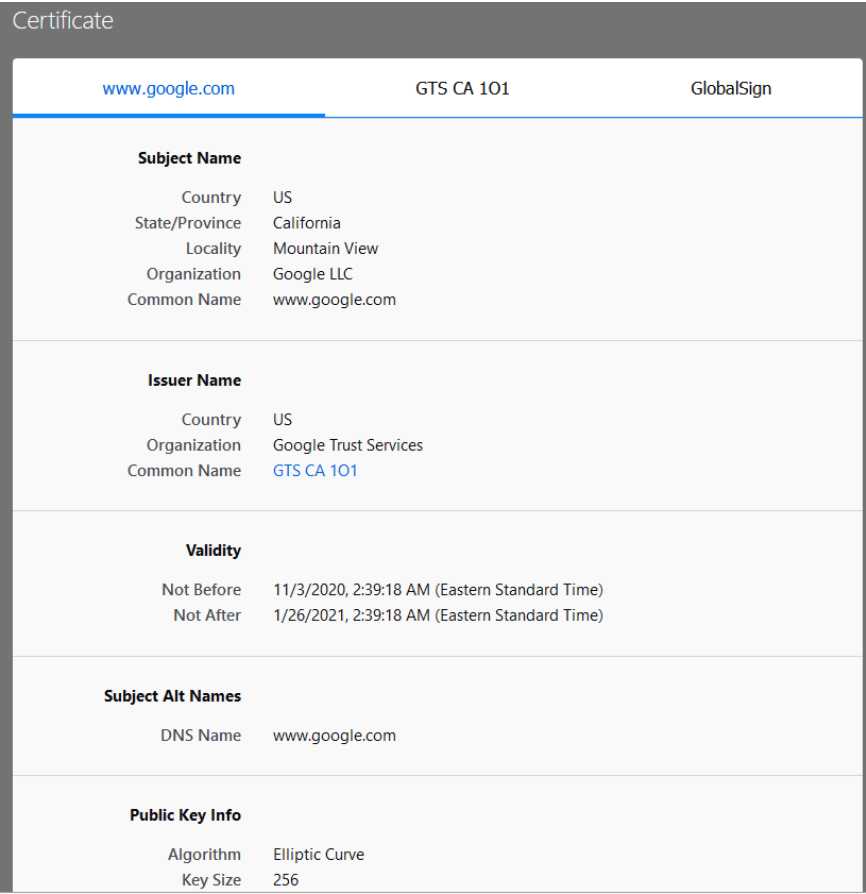
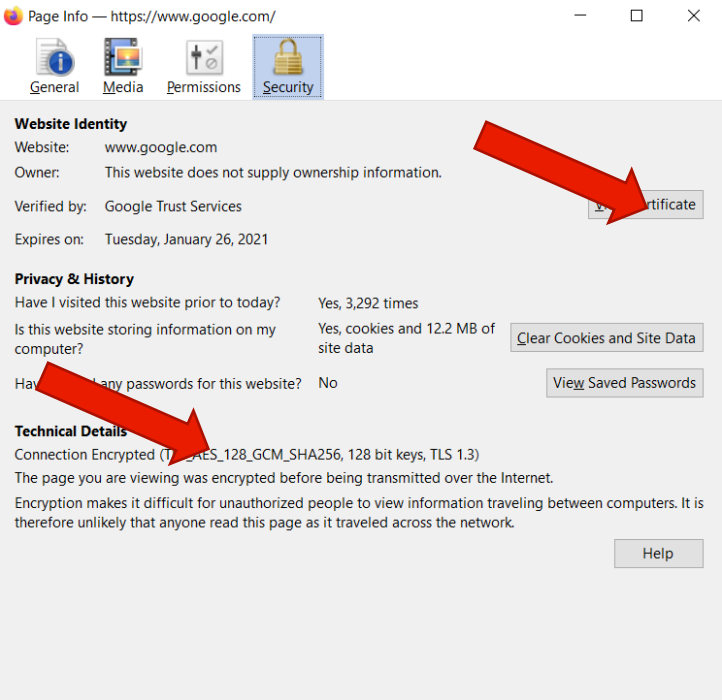
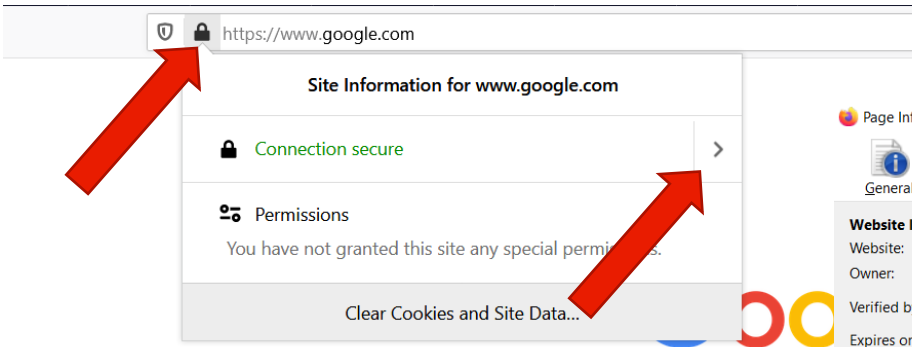


# Security and Cryptography

- History of Cryptography
- A fundamental component of our modern life
  - Confidentiality
    - No one is able to decipher your data.
  - Integrity
    - Ensure that the data was not altered.
  - Availability of Data
    - Data is available when needed.
  - Authenticity
    - Message came from the legitimate sender.
  - And many other...

# Security and Cryptography

- In Firefox browser



# Security and Cryptography

- In Chrome browser

The image illustrates the process of checking a website's security in Google Chrome. It shows the address bar with 'google.com', the Chrome menu with 'Developer tools' highlighted, the 'Security' tab in the developer tools showing a 'Security overview' with a green lock icon and a 'View certificate' button, and the 'Certificate Viewer' window for '\*.google.com'.

**Security overview**

This page is secure (valid HTTPS).

- **Certificate - valid and trusted**  
The connection to this site is using a valid, trusted server certificate issued by GTS CA 1C3.  
[View certificate](#)
- **Connection - secure connection settings**  
The connection to this site is encrypted and authenticated using QUIC, X25519, and AES\_128\_GCM.
- **Resources - all served securely**  
All resources on this page are served securely.

**Certificate Viewer: \*.google.com**

**General** Details

**Issued To**

Common Name (CN)	*.google.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

**Issued By**

Common Name (CN)	GTS CA 1C3
Organization (O)	Google Trust Services LLC
Organizational Unit (OU)	<Not Part Of Certificate>

**Validity Period**

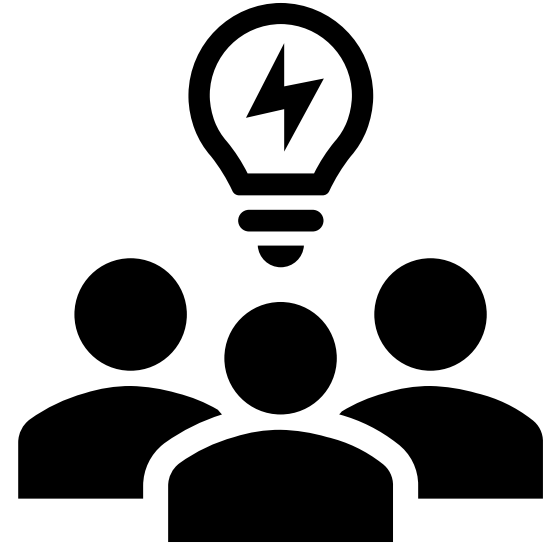
Issued On	Monday, October 17, 2022 at 4:16:44 AM
Expires On	Monday, January 9, 2023 at 3:16:43 AM

**Fingerprints**

SHA-256 Fingerprint	28 00 26 27 A6 9B D1 40 16 52 9C 3D E4 B5 94 E7 0A 4B E7 75 79 D6 95 FD 2B 4A 8C 72 57 2C 7C 31
SHA-1 Fingerprint	8D C5 65 10 71 12 6B 24 9B 99 8C 1F A9 0C D1 0E 07 C0 4D A4

# Key Players in Cryptography

- There are three players in this game
  - Mathematicians and Cryptographers
  - Cryptographic Engineers
  - Users



# User's Perspective

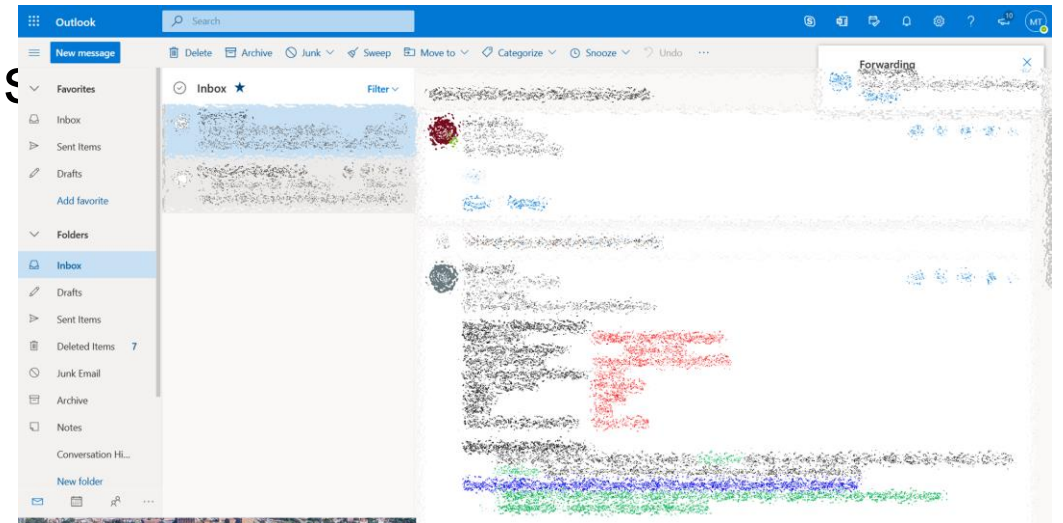


# User's Perspective

- Some helpful hints:
  1. Always initiate your own secure channels

# User's Perspective

- Some helpful hints:
  1. Always initiate your own secure channels
  2. Protect your email



Welcome to the Carleton SSO Portal.  
Enter your [MyCarletonOne](#) username and password.

☐ Keep me signed in

[Sign in](#)

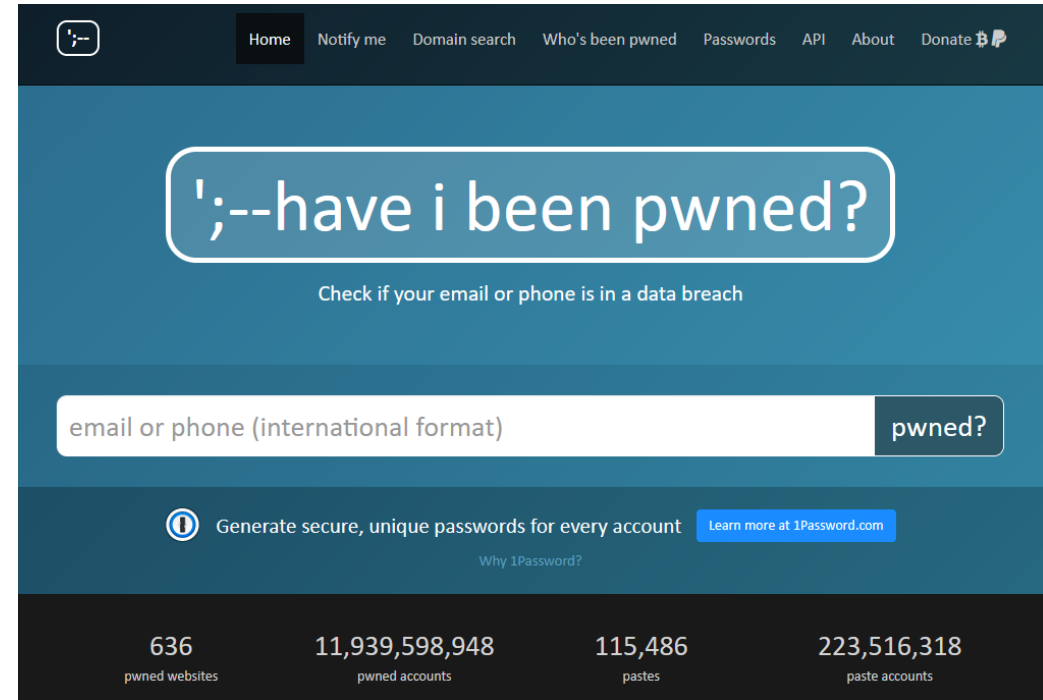
[Forgotten Username?](#) or [Forgotten Password?](#)

[New to Carleton and need a MyCarletonOne account?](#)

[Carleton.ca](#) [Privacy Policies](#) [Contact ITS](#)

# User's Perspective

- Some helpful hints:
  1. Always initiate your own secure channels
  2. Protect your email
  3. Understand Data Breaches
- Check: <https://haveibeenpwned.com/>



mtaha@vt.edu is hacked ➤

**mtaha@vt.edu**

to me ▼

Hello!

My nickname in darknet is xavier24.

I hacked this mailbox more than six months ago, through it I infected your operating system with a virus (trojan) created by me and have been monitoring you for a long time.

So, your password from [mtaha@vt.edu](mailto:mtaha@vt.edu) is **bronzedoor78**

Even if you changed the password after that - it does not matter, my virus intercepted all the caching data on your computer and automatically saved access for me.

I have access to all your accounts, social networks, email, browsing history.

Accordingly, I have the data of all your contacts, files from your computer, photos and videos.

# User's Perspective

- Some helpful hints:
  1. Always initiate your own secure channels
  2. Protect your email
  3. Understand Data Breach
    - Check: <https://haveibeenpwned.com/>
  4. Use unique passwords with a password manager
    1. Using “forgot your password” feature
    2. Using your email as a password manager
    3. Using offline password manager
    4. Using online password manager

# Developer's Perspective

# Developer's Perspective

- Coding for Security and Cryptographic Applications
  - Cryptographic Requirement:
  - Design a password verification code that accepts 8-bytes of user input and compares it against an 8-byte stored value.
  - Theoretical security:
    - The stored secure key is one of  $256^8 = 2^{64} = 18,446,744,073,709,551,616 = 18.4 \times 10^{18} = 18 \text{ exa}$
    - If a computer can check 1 million tries each second, it would need around 584,542 years to finish testing all the cases.
    - This seems to be a pretty good security.

# Developer's Perspective

- Coding for Security and Cryptographic Applications
  - Implementation
- Can you spot any problem?
- Hint: assume the adversary can measure the response time.

---

## Algorithm 4 Password verification.

---

**Input:**  $\tilde{P} = (\tilde{P}[0], \dots, \tilde{P}[7])$  (and  $P = (P[0], \dots, P[7])$ )

**Output:** 'true' or 'false'

```
1: for  $j = 0$  to  $7$  do  
2:   if  $(\tilde{P}[j] \neq P[j])$  then return 'false'  
3: end for  
4: return 'true'
```

---

Marc Joye, "Basics of Side-Channel Analysis"

# Developer's Perspective

- Coding for Security and Cryptographic Applications
  - Execution-Time Attack
- $\tilde{P}[0]$  is wrong:  
return 'false' very quickly
- $\tilde{P}[0]$  is correct and  $\tilde{P}[1]$  is wrong  
return 'false' after some time.
- $\tilde{P}[0]$  and  $\tilde{P}[1]$  are correct,  
but  $\tilde{P}[2]$  is wrong  
return 'false' after more time.

---

**Algorithm 4** Password verification.

---

**Input:**  $\tilde{P} = (\tilde{P}[0], \dots, \tilde{P}[7])$  (and  $P = (P[0], \dots, P[7])$ )

**Output:** 'true' or 'false'

```
1: for  $j = 0$  to  $7$  do
2:   if  $(\tilde{P}[j] \neq P[j])$  then return 'false'
3: end for
4: return 'true'
```

---

Marc Joye, "Basics of Side-Channel Analysis"



# Developer's Perspective

- Coding for Security and Cryptographic Applications

- For  $0 \leq n \leq 255$ , test  $\tilde{P}(n) = (n, 0, 0, 0, 0, 0, 0, 0)$  and measures the corresponding running time,  $\tau[n]$ .
- Find the maximum running time.

$$\tau[n_0] := \max_{0 \leq n \leq 255} \tau[n]$$

Assign  $P[0]$  to  $\tilde{P}(n_0)$ .

- Repeat for the second byte:  
For  $0 \leq n \leq 255$ , test  $\tilde{P}(n) = (P[0], n, 0, 0, 0, 0, 0, 0)$
- And so on.
- Assume using the same machine as previous (1 million test / sec). 255 trials can be tested in 1 msec. **The entire password could be found in 8 msec!!!!**

---

**Algorithm 4** Password verification.

---

**Input:**  $\tilde{P} = (\tilde{P}[0], \dots, \tilde{P}[7])$  (and  $P = (P[0], \dots, P[7])$ )

**Output:** 'true' or 'false'

```
1: for  $j = 0$  to 7 do
2:   if  $(\tilde{P}[j] \neq P[j])$  then return 'false'
3: end for
4: return 'true'
```

---

Marc Joye, "Basics of Side-Channel Analysis"

# Developer's Perspective

- Coding for Security and Cryptographic Applications
  - Part of computing an RSA Digital Signature  
First proposed in 1977, still in use today.

$d_j$  is the secret value.

If  $d_j = 0$ , do only  $(R_0 \leftarrow R_0^2)$

If  $d_j = 1$ , do, do both  
 $(R_0 \leftarrow R_0^2)$  and  $(R_0 \leftarrow R_0 \times R_1)$

---

**Algorithm 5** Computation of an RSA signature.

---

**Input:**  $m, N, d = (d_{k-1}, \dots, d_0)_2$ , and  $\mu : \{0, 1\}^* \rightarrow \mathbb{Z}/N\mathbb{Z}$

**Output:**  $S = \mu(m)^d \pmod{N}$

1:  $R_0 \leftarrow 1; R_1 \leftarrow \mu(m)$

2: **for**  $j = k - 1$  **downto** 0 **do**

3:      $R_0 \leftarrow R_0^2 \pmod{N}$

4:     **if**  $(d_j = 1)$  **then**  $R_0 \leftarrow R_0 \cdot R_1 \pmod{N}$

5: **end for**

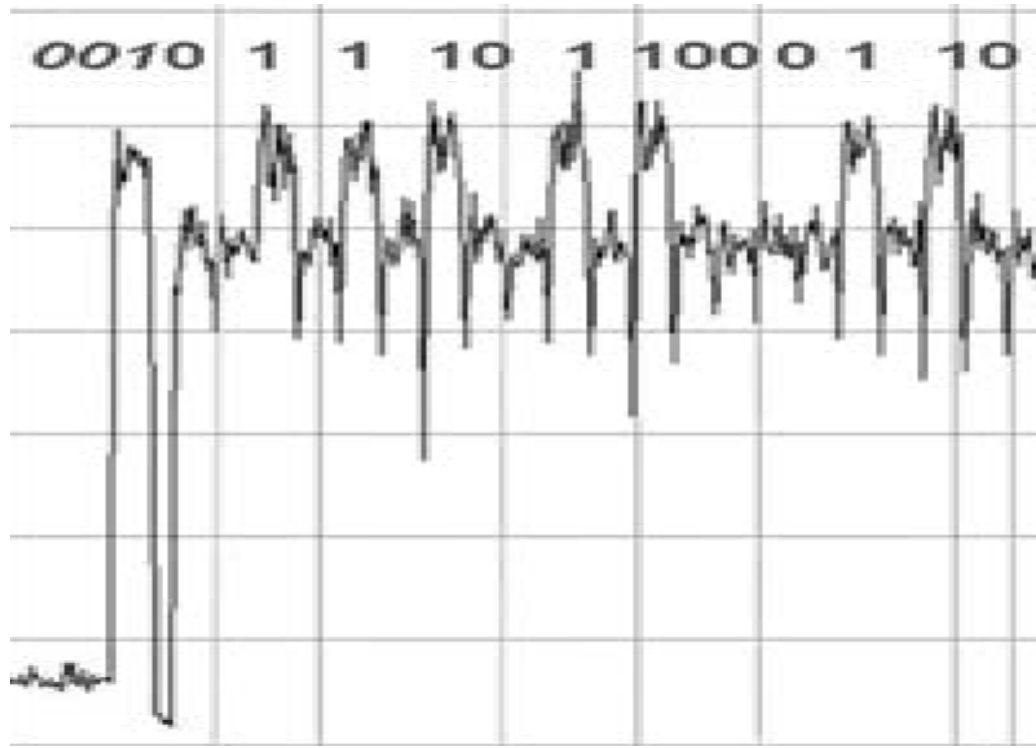
6: **return**  $R_0$

---

Marc Joye, "Basics of Side-Channel Analysis"

# Developer's Perspective

- Coding for Security and Cryptographic Applications
  - Power-Consumption Attack



---

**Algorithm 5** Computation of an RSA signature.

---

**Input:**  $m, N, d = (d_{k-1}, \dots, d_0)_2$ , and  $\mu : \{0, 1\}^* \rightarrow \mathbb{Z}/N\mathbb{Z}$

**Output:**  $S = \mu(m)^d \pmod{N}$

1:  $R_0 \leftarrow 1; R_1 \leftarrow \mu(m)$

2: **for**  $j = k - 1$  **downto** 0 **do**

3:      $R_0 \leftarrow R_0^2 \pmod{N}$

4:     **if**  $(d_j = 1)$  **then**  $R_0 \leftarrow R_0 \cdot R_1 \pmod{N}$

5: **end for**

6: **return**  $R_0$

---

Marc Joye, “Basics of Side-Channel Analysis”

# Introduction to Security and Cryptography

- Lecture Summary:
  - Do your part as a user
    - Initiate your own secure channels
    - Protect your email
    - Understand Data Breaches
    - Use unique passwords
  - As a developer
    - Be cautious about coding/modeling for security/cryptography applications.
    - Efficiency may be your worst enemies.

**Carleton**  
University

