Cybersecurity

Architecture and Design

2.8.7 Common Cryptography Use Cases

What device factors should be considered when implementing cryptography methods?

Overview
The student will summarize the basics of cryptographic concepts.

Grade Level(s)
10, 11, 12

Cyber Connections

- **Threats & Vulnerabilities**
- **Networks & Internet**
- **Hardware & Software**

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Teacher Notes:

CompTIA SY0-601 Security+ Objectives

Objective 2.8

- Summarize the basics of cryptographic concepts.
    - Common use cases
        - Low power devices
        - Low latency
        - High resiliency
        - Supporting confidentiality
        - Supporting integrity
        - Supporting obfuscation
        - Supporting authentication
        - Supporting non-repudiation

Common Cryptography Use Cases

Using Cryptography

Privacy has always been important to people. In today's world with numerous electronic devices, many factors of the device itself must be considered when trying to provide "reasonable" security and protection. Different devices have different needs such as cell phones versus desktops versus servers.

Bring Balance to the Force, Don't Leave it in Darkness!

Consider mobile devices: cell phones, tablets, smart watches, etc. These devices would be considered "low power" devices; they are powered by a battery instead of being connected to a constant power source. Because of the power limitations, we would have to use cryptographic techniques that require less computing power, which requires less electric power. Some of the ways to use less power would be to use small symmetric keys or elliptic curve cryptography.

In addition to power consumption, you must consider the ***latency*** requirements of an application. Latency is the delay before data transfer begins after the transfer has been requested. If an application needs low latency (little delay), then we will have to use an encryption that does not require many resources (from the CPU). Stream ciphers (symmetric) are typically used in this case because of the speed of encryption and decryption.

Teacher Notes:

Lastly, we want our techniques to have high ***resiliency***, which is the ability to resume normal operations after some form of disruption. For mobile devices, this disruption may come in the loss of power from the battery or loss of connection from Wi-Fi or cellular service.

Useful or Useless

The entire purpose of cryptography is to protect the ***confidentiality*** (secrecy) of information. The better the cryptographic techniques used, the more likely we are to keep data private. Using large encryption keys or hashing (preferably a combination of both) ensures private data stays private.

If there are concerns about the ***integrity*** of the data, then a strong encryption method will be necessary. Integrity refers to protecting information from being modified by unauthorized parties. This is commonly used during file transfers or password storage. For file transfer, we will verify that data has not been altered. For passwords, we can use a hash to store the password in a way that does not show the original password but is still useful in verifying authentication.

In a previous section we discussed security through ***obfuscation***. Obfuscation allows us to hide data so it is not "visible" to the average viewer. Attackers use obfuscation to hide malware in your system. The malware will start encrypted, allowing it to avoid detection by an antivirus program. After execution, the malware will decrypt itself and infect your computer. Not all uses of obfuscation are for nefarious purposes. An example of using obfuscation for protection would be to change the order of a PIN to make it more difficult for a malicious user to determine your PIN.

We use the term ***authentication*** (confirmation) when storing passwords after hashing them. This will allow for validation of the user. Even if two people share the same password, if we combine the password with a random salt, hashing the combination will make the passwords look unique. Therefore, if an attacker gains access to our stored password and salt list, they should see a list that has no repetitions in the digests (stored hashes).

Along with confidentiality, integrity, obfuscation, and authentication, we have what is called ***non-repudiation***. Non-repudiation refers to the assurance that someone cannot dispute a contract or deny the authenticity of their digital signature. This helps us confirm that supposed information sent from another party actually came from that party.

Finally, we have the constant battle between resource and security constraints.

Teacher Notes:

As mentioned above, this is a balancing act. It's essential when planning the implementation of a cryptographic system that you provide appropriate security using the right type of resources. For example, consider a web server. If a web browser does not support the types of encryptions from the web server, the browser will be useless. Older computers also would not have the computing resources to run current browsers. Similarly, if you tried to access a modern website with an old, outdated browser, older browsers may not support the current security certificates issued by modern certificate authorities.