Security Engineer

- Intro to Cryptography
  - Symmetric Encryption
  - Asymmetric Encryption
  - Authenticating With Passwords
  - Diffie-Hillman Key Exchange
  - Hashing
  - PKI & SSL/TLS
- Identity & Access Management
  - IAAA Model
  - Identity
  - Authentication

Intro to Cryptography

Symmetric Encryption

Overview

Symmetric encryption, or secret-key encryption, is a fundamental cryptographic method where the same key (the secret key) is used for both encryption (converting plaintext to ciphertext) and decryption (recovering the plaintext from the ciphertext). The communication parties must agree upon and securely exchange this secret key beforehand.

Key Information

Terminology:

- **Cryptographic Algorithm or Cipher** This algorithm defines the encryption and decryption processes.
- **Key** The cryptographic algorithm needs a key to convert the plaintext into ciphertext and vice versa.
- **plaintext** is the original message that we want to encrypt
- **ciphertext** is the message in its encrypted form

A symmetric encryption algorithm uses the same key for encryption and decryption.

Encryption Algorithm	Notes
AES, AES192, and AES256	AES with a key size of 128, 192, and 256 bits
IDEA	International Data Encryption Algorithm (IDEA)
3DES	Triple DES (Data Encryption Standard) and is based on DES. We should note that 3DES will be deprecated in 2023 and disallowed in 2024.

Encryption Algorithm	Notes
CAST5	Also known as CAST-128. Some sources state that CAST stands for the names of its authors: Carlisle Adams and Stafford Tavares.
BLOWFISH	Designed by Bruce Schneier
TWOFISH	Designed by Bruce Schneier and derived from Blowfish
CAMELLIA128, CAMELLIA192, and CAMELLIA256	Designed by Mitsubishi Electric and NTT in Japan. Its name is derived from the flower camellia japonica.

Notes

Popular tools for symmetric encryption:

1. **GNU Priacy Guard**: The GNU Privacy Guard, also known as GnuPG or GPG, implements the OpenPGP standard.
2. **OpenSSL Project**: The OpenSSL Project maintains the OpenSSL software.

GNU Privacy Guard

- Command to ecnrypt `gpg --symmetric --cipher-algo CIPHER message.txt`
- Ascii armorded output `gpg --armor --symmetric --cipher-algo CIPHER message.tx`
- command to decryp `gpg --output original_message.txt --decrypt message.gpg`

OpenSSL Project

- command to encrypt `openssl aes-256-cbc -e -in message.txt -out encrypted_message`
- command to decrypt `openssl aes-256-cbc -d -in encrypted_message -out original_message.txt`
- Password-Based Key Derivation Function 2 `openssl aes-256-cbc -pbkdf2 -iter 10000 -e -in message.txt -out encrypted_message`

Task

1. Decrypt the file `quote01` encrypted (using AES256) with the key `s!kR3T55` using `gpg`. What is the third word in the file?
    1. `gpg --output quote1.txt --decrypt quote01.txt.gpg`
    2. Third Word `waste`
2. Decrypt the file `quote02` encrypted (using AES256-CBC) with the key `s!kR3T55` using `openssl`. What is the third word in the file?
    1. `openssl aes-256-cbc -d -in quote02 -out quote2`
    2. Third Word `science`
3. Decrypt the file `quote03` encrypted (using CAMELLIA256) with the key `s!kR3T55` using `gpg`. What is the third word in the file?
    1. `gpg --output quote3.txt --decrypt quote03.txt.gpg`
    2. Third Word `understand`

Conclusion

Symmetric encryption is a cryptographic method where a single secret key is used to encrypt plaintext into ciphertext and decrypt it back. While historical algorithms like DES (56-bit key) were broken, modern standards like AES (128/192/256-bit keys) remain secure and provide confidentiality, integrity, and authenticity. Popular implementations include GnuPG (GPG) and OpenSSL. Despite its security benefits, symmetric encryption suffers from a scalability problem because the number of required keys grows quadratically with the number of users, making it impractical for large-scale key distribution.

Resources

- **TryHackMe**: Intro to Cryptography
- **OpenSSL Project**: OpenSSL
- **GNU Privacy Guard**: GPG

Asymmetric Encryption

Authenticating With Passwords

Overview

Cryptography is essential for protecting passwords both in transit (via SSL/TLS) and at rest (in databases), where the latter requires secure storage methods to prevent exposure during a breach. Simply hashing passwords is insufficient due to rainbow tables, necessitating the use of a unique salt for each password before hashing to significantly improve security.

Key Information

- Data at Rest vs. In Transit: Passwords must be protected in transit (e.g., HTTPS/SSL/TLS) and when stored as data at rest in a database.
- Simple Hashing Flaw: Storing passwords as simple hashes is vulnerable because attackers can use pre-computed lists called rainbow tables to reverse the hash back to the original password.
- Salting: To counter rainbow tables, a unique, random value called a salt must be appended (or mixed) with the password before the hashing process.
- Improved Hashing: The combination of the password and its unique salt is then hashed, making the pre-computation of a rainbow table infeasible for all possible salt combinations.
- Key Derivation Functions (KDFs): For maximum security, functions like PBKDF2 are recommended; they use hashing but incorporate a high number of iterations to intentionally slow down the hashing process, which frustrates brute-force attacks.

Notes

Ways to store a Password

Least Secure (Plain Password)

UserName	Password
alice	qwerty

Better (Hash)

UserName	Hash (Password)
alice	d8578edf8458ce06fbc5bb76a58c5ca4

Best (Hash + Salt)

User	Hash (Password + salt)	Salt
alice	8a43db01d06107fcad32f0bcfa651f2f	12742

PBKDF2

PBKDF2 (Password-Based Key Derivation Function 2) takes the password and the salt and submits it through a certain number of iterations, usually hundreds of thousands

---

Task

1. You were auditing a system when you discovered that the MD5 hash of the admin password is `3fc0a7acf087f549ac2b266baf94b8b1`. What is the original password?
    1. Used Crack Station to get the value of the plain md5 hash.
    2. **qwerty123**

Conclusion

Protecting stored passwords requires more than simple hashing, which is easily defeated by rainbow tables, but mandates the use of a unique salt for every password to ensure a data breach only yields unique, un-invertible hash-salt pairs. For future-proofing against increased computing power, best practice involves utilizing Key Derivation Functions like PBKDF2, which introduce high computational cost through iterative hashing.

Resources

- **TryHackMe**: Intro to Cryptography
- **Password Storage**: Cheat Sheet
- **MD5 Checker**: Crack Station

Diffie-Hillman Key Exchange

Hashing

Overview

Cryptographic hash functions transform data of any size into a fixed-length message digest or checksum, with SHA256 producing a 256-bit (64 hexadecimal digit) output regardless of input size. These functions are deterministic and demonstrate the avalanche effect—even a single-bit change in input produces a completely different hash value. Hash functions serve critical security purposes including secure password storage and detecting file modifications or tampering. Older algorithms like MD5 and SHA-1 are now cryptographically broken and vulnerable to collision attacks.

Key Information

- Fixed Output Size: Hash functions always produce the same length output regardless of input file size (e.g., SHA256 always produces 256 bits or 64 hex digits)
- Avalanche Effect: A single-bit change in input data produces a drastically different hash value, enabling reliable tamper detection
- Primary Applications: Password storage (with salting), file integrity verification, and detecting both intentional tampering and transfer errors
- Secure vs. Broken Algorithms: SHA224, SHA256, SHA384, SHA512, and RIPEMD160 are currently secure; MD5 and SHA-1 are cryptographically broken and susceptible to hash collisions
- HMAC Authentication: HMAC combines a hash function with a secret key using inner/outer padding (ipad/opad) to provide message authentication.

Notes

`sha256sum file`

`hmac256 key file`

Task

1. What is the SHA256 checksum of the file `order.json`?
    1. `sha256sum order.json`
    2. `2c34b68669427d15f76a1c06ab941e3e6038dacdfb9209455c87519a3ef2c660`
2. Open the file `order.json` and change the amount from `1000` to `9000`. What is the new SHA256 checksum?
    1. `sha256sum order.json`
    2. `11faeec5edc2a2bad82ab116bbe4df0f4bc6edd96adac7150bb4e6364a238466`
3. Using SHA256 and the key `3RfDFz82`, what is the HMAC of `order.txt`?
    1. `hmac256 3RfDFz82 order.json`
    2. `c7e4de386a09ef970300243a70a444ee2a4ca62413aeaeb7097d43d2c5fac89f`

Conclusion

Cryptographic hash functions are fundamental security tools that provide both data integrity verification and secure password storage mechanisms. Understanding the difference between secure algorithms (SHA-256 family) and broken ones (MD5, SHA-1) is essential for implementing modern cybersecurity solutions. HMAC extends basic hashing by incorporating secret keys, making it suitable for message authentication in scenarios requiring both integrity and authenticity verification.

Resources

- **TryHackMe**: Intro to Cryptography

PKI & SSL/TLS

Overview

The fundamental Diffie-Hellman key exchange is susceptible to a Man-in-the-Middle (MITM) attack because it lacks a mechanism for participants to authenticate each other's identity, allowing an attacker to establish two separate secret keys and decrypt all communication. This critical security gap is filled by Public Key Infrastructure (PKI), which introduces trust by using digital certificates signed by a universally trusted third party called a Certificate Authority (CA). Consequently, modern protocols like HTTPS rely on the client's ability to verify the server's certificate signature, ensuring that the initial key exchange and subsequent encrypted communication are indeed with the legitimate intended party.

Key Information

- Diffie-Hellman (DH) Flaw: The standard DH key exchange lacks authentication, leaving it vulnerable to Man-in-the-Middle (MITM) attacks.
- MITM Attack: An attacker can intercept public values and establish two separate secret keys (one with Alice, one with Bob), allowing them to read and modify all communication.
- PKI Solution: Public Key Infrastructure (PKI) resolves this by providing an identity verification mechanism.
- Digital Certificates: PKI uses digital certificates which cryptographically bind a public key to an identity.
- Trusted CAs: These certificates are signed by a Certificate Authority (CA) (a trusted third party), enabling protocols like HTTPS to ensure clients are communicating with the genuine server.

Notes

Creating a certificate with openssl

```
openssl req -new -nodes -newkey rsa:4096 -keyout key.pem -out cert.csr
```

- `req -new` create a new certificate signing request
- `-nodes` save private key without a passphrase
- `-newkey` generate a new private key
- `rsa:4096` generate an RSA key of size 4096 bits
- `-keyout` specify where to save the key
- `-out` save the certificate signing request

Viewing a certificate and its information

```
openssl x509 -in cert.pem -text
```

- `x509` Specifies that you want to perform operations related to X.509 digital certificates
- `-in` Specifies the input file
- `-text` output the certificate content in a human-readable, detailed text format, rather than the raw encoded form

---

Task

1. What is the size of the public key in bits?
    1. `openssl x509 -in cert.pem -text`
    2. **Public Key: (4096 bits)**
2. Till which year is this certificate valid?
    1. `Not After : Feb 25 11:34:19 2039 GMT`
    2. **2039**

Conclusion

The inherent lack of identity verification in the basic Diffie-Hellman key exchange leaves it vulnerable to a crippling MITM attack where all communication is compromised. This fundamental flaw is securely mitigated by PKI, which leverages CA-signed digital certificates to authenticate the

server's identity, thereby guaranteeing the integrity and confidentiality of modern communication protocols like HTTPS.

Resources

- **TryHackMe**: Intro to Cryptography

Identity & Access Management

IAAA Model

Overview

The IAAA model consists of four essential pillars—Identification, Authentication, Authorization, and Accountability—that work together to protect sensitive information and resources in an organization. Identification establishes who a user claims to be through unique identifiers like usernames or email addresses, while Authentication verifies that claim through methods such as passwords or verification codes. Authorization then determines what resources and operations the authenticated user is permitted to access based on their role and privileges. Together, these three elements form a security foundation that is reinforced by Accountability, which logs and tracks all user activity for incident investigation and responsibility enforcement.

Key Information

- Identification: User claims an identity using unique identifiers (email, username, ID number) to establish who they are in the system
- Authentication: Verification process confirming the user's claimed identity through credentials (passwords, codes, multi-factor methods) to ensure they are who they claim to be
- Authorization: Access control mechanism that grants or restricts user permissions based on assigned roles and job functions, limiting access to only necessary resources
- Accountability: Logging and monitoring system that tracks all user activities in a centralized location, enabling incident investigation and ensuring users are responsible for their actions
- Security Benefit: IAAA implementation prevents unauthorized access, reduces data breach risk, and enables organizations to respond effectively to security incidents through audit trails

---

Task

1. You are granted access to read and send an email. What is the name of this process?
    1. **Authorisation**
2. Which process would require you to enter your username?
    1. **Identification**
3. Although you have write access, you should only make changes if necessary for the task. Which process is required to enforce this policy?
    1. **Accountability**

Conclusion

The IAAA model provides a comprehensive security framework that addresses both access control and audit requirements essential for modern cybersecurity. By systematically implementing identification, authentication, authorization, and accountability mechanisms, organizations can significantly reduce vulnerability to internal and external security threats. Understanding each component's distinct role is fundamental for 4th-year cybersecurity students designing secure systems and developing security policies.

Resources

- **TryHackMe**: Intro to Cryptography

Identity

Overview

Identification is the process by which a user, process, or system claims a specific identity through a unique identifier, without any verification of that claim's truthfulness. Identifiers can take various forms including usernames (such as tanderson, neo, or thomas01), email addresses, national ID numbers, student IDs, passport numbers, or phone numbers—any attribute that is reasonably unique within a given context. The key distinction is that identification is purely a claim of identity, similar to someone at a party stating their name; the system accepts this claim at face value without confirming its authenticity. This process sets the foundation for subsequent security measures like authentication, which verify whether the claimed identity is legitimate.

Key Information

- Claim-Based Process: Identification involves a user stating who they are through a unique identifier without requiring proof or verification of that claim
- Identifier Types: Common identifiers include usernames, email addresses, national ID numbers, student IDs, passport numbers, and mobile phone numbers
- Organizational Variation: Different organizations and platforms use different identifier formats (e.g., tanderson, thomasa, ta001, or neo could all represent the same person)
- Email as Identifier: Many websites use email addresses for identification because they are globally unique and eliminate the burden of users creating and remembering usernames
- Foundation for Security: Identification alone is insufficient for system security; it must be followed by authentication to prevent unauthorized access and fraud (such as someone falsely claiming to be a gym member or loan applicant)

Task

1. Which of the following cannot be used for identification? `Email address` `Mobile number with international code` `Year of birth` `Passport number`
    1. **Year of Birth**
2. Which of the following cannot be used for identification? `Landline phone number` `Street number` `Health insurance card number` `Student ID number`
    1. **Street Number**

Conclusion

Identification serves as the initial step in the IAAA model, establishing a claimed identity within a system, but provides no security guarantee on its own. Without subsequent authentication mechanisms, systems become vulnerable to impersonation and fraud, making proper identification combined with strong authentication critical for protecting sensitive resources and maintaining system integrity. Understanding this distinction is essential for designing secure systems that prevent unauthorized access and protect legitimate users.

Resources

- **TryHackMe**: Intro to Cryptography

Authentication

Overview

Authentication is the process of verifying a user's or system's claimed identity, distinct from identification which is simply claiming that identity. The primary mechanisms for authentication include something you know (passwords, PINs), something you have (security keys, phones), and something you are (biometrics). Multi-factor authentication (MFA) combines two or more of these mechanisms to significantly enhance security against compromised single factors.

Key Information

- Something You Know – Includes passwords, passphrases, and PINs that users memorize; examples include complex strings like "4SNoPawKkdFiCdnm" and numeric codes like "25063"
- Something You Have – Physical objects such as hardware security keys (Yubico, Titan Security Key), SIM cards, or mobile phones used to receive verification codes via SMS or NFC
- Something You Are – Biometric authentication methods including fingerprint readers, facial recognition, retina scanners, and voice recognition that are becoming increasingly affordable and reliable
- Multi-Factor Authentication (MFA) – Combines two or more authentication mechanisms to provide layered security; classic example is an ATM requiring both a debit card (something you have) and a PIN (something you know)
- Real-World Applications – Authentication is essential in everyday scenarios like gym membership verification, mobile phone unlocking, banking systems, and instant messaging app registration

---

Notes

Task

1. .

Conclusion

Understanding the three primary authentication mechanisms and their combinations through MFA is critical for designing secure systems. Organizations and individuals should implement MFA where possible, as it substantially reduces the risk of unauthorized access even if one authentication factor is compromised. The evolving affordability and reliability of biometric technologies make MFA increasingly practical for widespread deployment across both enterprise and consumer applications.

Resources

- **TryHackMe**: Intro to Cryptography