Project Reading

Reading

20m - 40m

Status Incomplete

Introduction

This reading will not only help you review the encryption-related concepts but also introduces you to new examples to help you with the upcoming project.

What is Cryptography?

Cryptography is a collection of techniques for:

- Concealing data transmitted over insecure channels
- Validating message integrity and authenticity

Some Cryptographic Terms

- plain text: a message or other data in readable form.
- ciphertext: a message concealed for transmission or storage.
- encryption: transforming plain text into ciphertext.
- decryption: transforming ciphertext back into plain text.
- key: an input to an encryption or decryption algorithm that determines the specific transformation applied.
- hash: the output of an algorithm that produces a fixed N-bit output from any input of any size.
- entropy: the number of possible states of a system or the number of bits in the shortest possible description of a quantity of data. This may be less than the data size if it is highly redundant.

Basic Cryptographic Algorithms

- Symmetric ciphers: a symmetric cipher uses the same key for encryption and decryption. In semi-mathematical terms:

  encryption: ciphertext = E(plaintext, key)

  decryption: plaintext = D(ciphertext, key)

  Two parties that want to communicate via encryption must agree on a particular key. Sharing and protecting that key is often the most difficult part of protecting encryption security. The number of possible keys should be large enough that a third party can’t feasibly try all of the keys (“brute forcing”) to see if one of them decrypts a message.

- Block ciphers: a block cipher works on fixed-size units of plain text to produce (usually) identically-sized units of ciphertext, or vice-versa.

  Example block ciphers:

  DES with a 64-bit block and 56-bit keys, now obsolete because both the block size and key size are too small and allow for easy brute forcing.

  AES (formerly known as Rijndael) with 128-bit blocks and keys of 128-, 192-, or 256-bit.

- Stream ciphers: a stream cipher produces a stream of random bits based on a key that can be combined (usually using XOR) with data for encryption or decryption.

Example Public-Key Algorithms

RSA is based on modular arithmetic using large prime numbers and the difficulty of factoring large numbers. At this time, 2048-bit primes are considered necessary to create secure RSA keys (factorization of keys based on 512-bit primes has already been demonstrated, and 1024-bit keys appear feasible).

Elliptic curve (EC) algorithms based on integers and modular arithmetic satisfying an equation of the form y^2 = x^3 + a*x + b. EC keys can be much shorter (256-bit EC keys are roughly equivalent to 3072-bit RSA keys).

However, public key algorithms are many (hundreds to thousands) times slower than symmetric algorithms, making it expensive to send large amounts of data using only public key encryption. However, public key algorithms provide a secure way to transmit symmetric cipher keys.

- Diffie-Hellman key exchange: an algorithm that allows two parties to create a shared secret through a public exchange from which an eavesdropper cannot feasibly infer the secret. It is useful for establishing a shared symmetric key for encrypted communication. Diffie-Hellman can be performed using either modular arithmetic with large prime numbers or EC fields. Diffie-Hellman is also usually the basis of “forward secrecy”. One key exchange method possible in SSL/TLS is simply using a public key algorithm to send a key between a client and a server. However, if the private key of that SSL/TLS certificate is later exposed, someone who monitored and recorded session traffic could decrypt all the keys used in the sessions they recorded. Forward secrecy involves setting up unique, random session keys for each communication session and using an algorithm like Diffie-Hellman, which establishes those keys in a way that is inaccessible to an eavesdropper.

- Hash algorithms: a hash (or cryptographic checksum) reduces input data (of any size) to a fixed-size N-bit value. In particular, for cryptographic use, a hash has these properties:

- Two different inputs are unlikely to produce the same hash (“collision”).

- Finding another input that produces any specified hash value (“preimage”) should be very difficult. Even a one-bit change in the input should produce a different hash in about N/2 bits.

Example Hash Algorithms

MD5 produces a 128-bit hash from its input. It has demonstrated collisions and feasible preimage computation and should not be used.

SHA1 produces 160-bit hashes but has at least one demonstrated collision and is also deprecated for cryptographic use (however, it is still used in git because it is still working as a hash function).

SHA-256 produces 256-bit hashes. SHA-224 is an SHA-256 hash truncated to 224 bits.

SHA-512 produces a 512-bit hash, and SHA-384 truncates an SHA-512 hash to 384 bits.

- Cryptographic random number generators: many cryptographic methods require producing random numbers (for generating unique keys or identifiers). Traditional pseudo-random number generators produce output that can be highly predictable, as well as often starting from known states and having relatively short periods. A cryptographic random number generator must make it very difficult to determine the prior (or future) state of the generator from its current output, and have enough entropy to generate sufficiently large random numbers.

Cryptographic Protocols

- Cipher modes: the simplest thing you can do with a block cipher is break plain text up into blocks, then encrypt each block with your chosen key (also called ECB for “Electronic Code Book”, by analogy with codes that simply substituted code words). Unfortunately, this leads to weakness: if a particular plain text block is repeated in the input, the ciphertext block also repeats in the output. This can quickly happen in English text if a phrase just happens to line up with a block the same way more than once.
- Message signing: someone who has created a public key pair (K1, K2) and published a public key (let’s say that’s K2) can encrypt a message using their private key K1, and anyone can validate that the message came from that sender by decrypting it with the public key K2.

Due to the much higher computational cost of encrypting data with public key algorithms, usually, the signer encrypts only a cryptographic hash of the original message. A sender can also send a plain text message along with a signature created with their private key if the privacy of the message is not essential, but validating the sender's identity is.

Message signing is also the basis of SSL/TLS certificate validation. A certificate contains a public key and a signature of that key generated with the private key of a trusted Certificate Authority. An SSL/TLS client (such as a web browser) can confirm the authenticity of the public key by validating the certificate signature using the public key of the Certificate Authority that signed it. “Self-signed” certificates are public keys signed with the corresponding private key. This isn’t as trustworthy (assuming you have reasons to trust a certificate authority) but doesn’t require interaction with a Certificate Authority. However, the buck has to stop somewhere; even Certificate Authority “root certificates” are self-signed.

Rather than the centralized Certificate Authority model (where certain authorities are trusted to sign certificates), email encryption tools like GPG have a “web of trust” model where many other individuals or entities can sign someone’s public key so that if you trust at least some of those others, it gives you greater assurance that a public key is valid and belongs to the indicated person. Without such signatures, someone could presumably publish a key purporting to be someone else, and there’d be no easy way to validate it.

- Secure email: if you want people to be able to send you a secure email (such as with PGP, GPG, or S/MIME), you create a public key pair (K1, K2) and publish the public key K2. Someone who wants to send you mail picks a cipher and generates a unique, random key for that cipher. They encrypt their plain text message with that cipher and key, encrypt the key with your public key, and send you a message containing the ciphertext, the cipher algorithm they used, and the encrypted cipher key.
- SSL/TLS: SSL and TLS use cryptographic primitives to secure data sent over a network. As a result, the protocol could be simpler. Client and server agree on a “cipher suite” to use, which consists of the following:
- A method for key exchange (via the public/private key pair in a certificate or Diffie-Hellman key exchange).
- A technique for server validation (based on the public key algorithm used in its certificate).
- A symmetric cipher for bulk data encryption.
- A hash algorithm to use for message authentication, actually an HMAC that hashes a combination of a secret key and the data.
- Establish a random shared key for the symmetric cipher and HMAC using the specified key exchange method.
- Transmits data using the specified symmetric cipher and HMAC algorithms.

Cryptanalysis

Cryptanalysis is the study of weaknesses in cryptographic algorithms and protocols. In general, good algorithms and protocols have been subjected to many public cryptanalyses that have yet to result in significantly better attacks than brute force.

Cryptographic Tools

- OpenSSL: although it’s taken a lot of heat for some of its previous security issues (particularly “Heartbleed”), it’s still the most widely used cryptographic library because of its portability and completeness. The OpenSSL command-line utility also provides a lot of useful functionality. It can create certificate requests or sign certificates, encrypt/decrypt files, transform several file formats used for cryptographic data, and more.
- GnuTLS: the GNU Project’s SSL/TLS library includes a gnutls-cli utility with similar (but less extensive) functionality for SSL/TLS client connections and encryption/decryption.

- GnuPG: primarily intended for encrypting or decrypting secure mail messages, it also provides some functionality for encrypting or decrypting files and creating or validating signatures.

References

- https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic*Storage*Cheat_Sheet.html

---

✓ Mark Completed

Previous
Data Integrity Email

How well did this activity help you to understand the content?

Let us know how we're doing

W07D3

Wed Aug 7

- Lectures (1)
- Work (7)

  6 hrs

  - Configure Apache Web Server
  - Capture an HTTP and an HTTPS Wireshark PCAP
  - Report to IT Manager
  - Hash Functions, Data Integrity and Digital Signatures
  - Key Establishment Protocols and Management Techniques
  - Data Integrity Email
  - Project Reading

W07D3 Schedule »