Cryptography 101

Robert Boedigheimer
@boedie

About Me

- Web developer since 1995
- Pluralsight Author
- 3rd Degree Black Belt, Tae Kwon Do
- Microsoft MVP
- Progress Developer Expert - Fiddler

- boedie@outlook.com
- @boedie
- weblogs.asp.net/boedie

Background

- Cryptography is the science of keeping messages secure
- Why Cryptography?
  - **Confidentiality** – protect data from being read
  - Integrity – verify that data was not modified
  - Authentication – identify and validate a user
  - Non-repudiation – sender cannot deny later that he sent a message

- System.Security.Cryptography

Considerations

- What is your goal? (Confidentiality, etc.)
- How much is data worth?
- How long does it need to be secured?
- What are the primary threats?
  - In transit
  - Access configuration files
  - Dump of memory
  - Modify pages
  - Reverse engineer assemblies
  - …
- Company security policies?
- Regulatory compliance?
- Layered defenses, how many are enough?

- Don’t write own!!

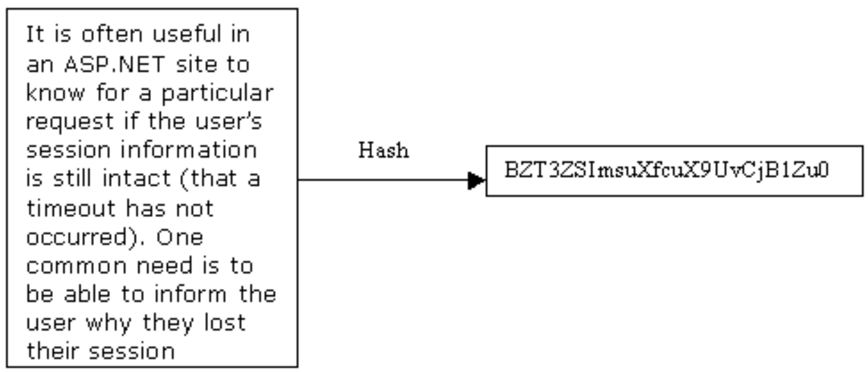
.NET Class Suffixes

- …Cng
  - Wrapper around Cryptography Next Generation (CNG)
    - Active development, newer OS required
- …CryptoServiceProvider
  - Wrapper around Windows Cryptography API (CAPI)
    - No longer developing but available on older OS
- …Managed
  - Written entirely in managed code
  - Need .NET framework
  - Not FIPS compliant

- https://tinyurl.com/o2zgbjk

Hash Functions

- One-way function – easy to compute but significantly harder to reverse
- Hash function – converts a variable length input to a fixed length
  - Creates a “data fingerprint” (digest)
  - Ok to see, don’t let it be tampered with
  - Be careful when limited value range!



Hash Algorithms

- Abstract base HashAlgorithm
  - ~~MD5 (128 bit hash)~~
  - SHA (Secure Hash Algorithm)
    - ~~SHA-1 (160 bit hash)~~
    - SHA-2
      - SHA256
      - SHA384
      - SHA512
  - KeyedHashAlgorithm
    - HMACSHA1 (up to 512)
    - MACTripleDES

(subset of derived classes shown)

Tamperproof Querystrings

- Goal is to protect **integrity** of querystring
- Use a Hash-based Message Authentication Code (HMAC)
  - Compute the hash of a querystring when constructed
  - Validate querystring was not modified by computing hash with querystring and comparing to original hash
  - Uses a key to ensure that attacker could not create own valid hash

Hashed Passwords

- Considered best practice for passwords since they cannot be retrieved
- Used for authentication

- Common attack against hashed passwords is “dictionary attack”
  - Pre-compute the hash values of an entire dictionary, compare hashed values to hashed password to look for matches

Salted Passwords

- Add some unique random data to each password
- Greatly increases work required to mount a dictionary attack against **all** passwords, need to pre-compute dictionary hash values for all salt values

- NOTE: This does nothing to increase security for an individual password if salt is easily found! (Add "random data" to do this…)

PBKDF2 (Password-Based Key Derivation Function 2)

- Compute power constantly increasing, so brute force attacks against hash functions are possible
- Add a “work factor” to the calculation based on a number of iterations
  - Set iterations to get acceptable time for login

- Rfc2898DeriveBytes

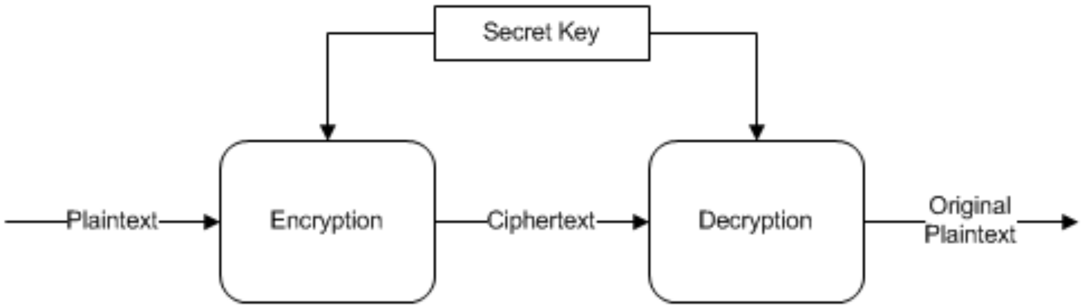
Terminology

- Plaintext – original data
- Encryption – process of obscuring data
- Ciphertext – encrypted data
- Decryption – process to recover original data

- Cipher – algorithm for performing encryption and decryption

Symmetric Algorithms

- Encryption and decryption use the same (secret) key
- Primary attack is “brute force” key search, try all possible keys
- Key distribution is difficult



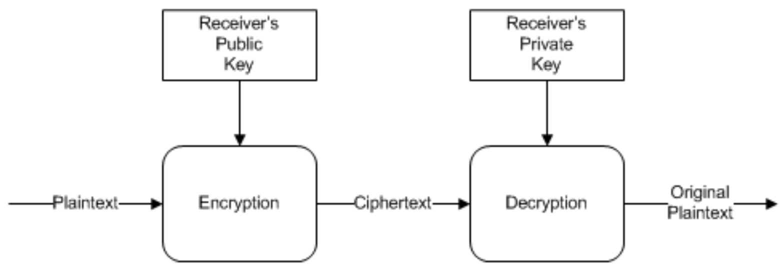
- Abstract class SymmetricAlgorithm
  - Rijndael (AES)
  - ~~DES~~
  - TripleDES

Symmetric Algorithms (cont.)

- .NET symmetric algorithms are "block ciphers"
- Padding – data added to fill to block size
  - Zeros
  - PKC27
  - **ISO10126**
- Mode
  - ECB
  - **CBC** (recommend)
- IV (Initialization Vector)
  - Random data used to seed first block
  - Does not need to be secret
  - Never reuse, always unique for each set of data!

Asymmetric Algorithms

- Utilizes two complimentary keys (public key and private key)
- Generally 1,000 times slower than symmetric algorithms
- Often use asymmetric to encrypt a “session” symmetric key



- Abstract class AsymmetricAlgorithm
  - RSA
  - DSA (digital signatures only)
  - ECDiffieHellman

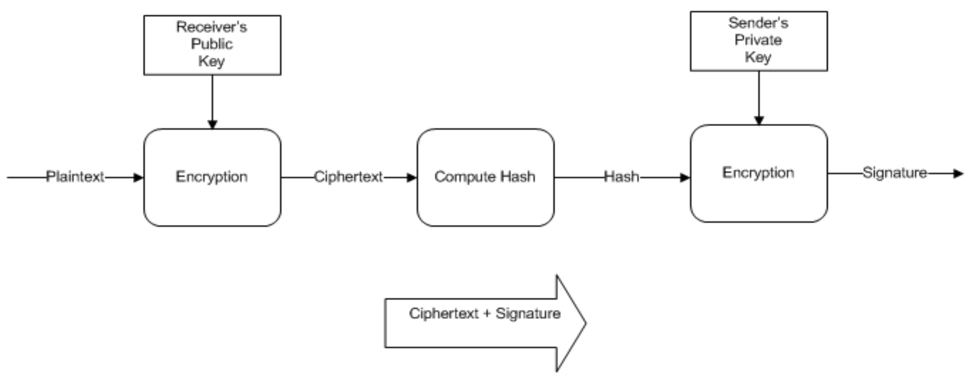
Website Encrypting Safely

- Generate an RSA key pair
  - Store only the public key on web servers
  - Store the private key on an internal secured system that needs the data
- Meant for small amounts of data

Digital Signatures

- Provides integrity and non-repudiation
- Hash the contents of a message, sign it (encrypt) with senders private key

- By default, does not provide confidentiality, can encrypt with receivers public key before signing



HTTPS

- Certificate (relies on asymmetric encryption)
  - Server's **public** key is digitally signed by a Certificate Authority (CA)
- Browser knows "well-known" CA's and will trust certificates signed by them

- TLS handshake
  - Browser gets server certificate
  - Browser chooses symmetric key to encrypt traffic, encrypts with server's public key

Key Sizes and Storage

- Key sizes
  - Tradeoff performance and security
  - Symmetric AES use 256 bits
  - Asymmetric RSA use 2048 or 4096
- Key storage
  - Hardcoded strings are visible if use a disassembler (like ILDASM)
  - Encrypted <appSetting> section of web.config
  - Split key in code, registry, and config files

Summary

- **Don't write own!**

- Use trusted algorithms and implementations
  - https://tinyurl.com/o2zgbjk
- Use hashing to validate the integrity of data or to prove both know the same secret
- Use symmetric algorithms unless have special needs for asymmetric (digital signatures, key exchange, etc)
- Know threats, choose the proper countermeasures

Resources

- Pluralsight – Introduction to Cryptography
  - https://tinyurl.com/kkn3coq

- Applied Cryptography - Bruce Schneier
- Cryptography Engineering – Ferguson, Schneier, Kohno
- Understanding Cryptography – Paar, Pelzl

- The Code Book – Simon Singh
- The Code-Breakers – Kahn

Questions

- boedie@outlook.com
- @boedie
- weblogs.asp.net/boedie

- Code and slides - https://tinyurl.com/ybygpvdz