CSF 434/534: Advanced Network and System Security

Week 12 - Review

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Sources: Professor Messer's CompTIA SY0-501 Security+ Course Notes

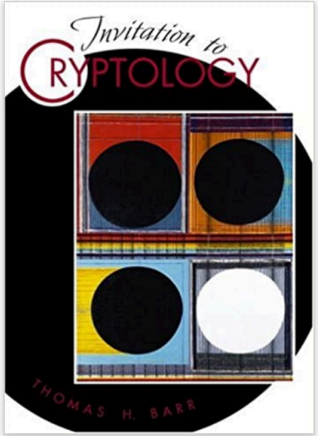
Cryptography Concepts

Cryptography Concepts

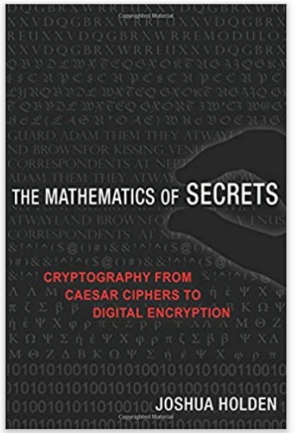
Cryptography

- Greek: "kryptos"
  - Hidden, secret
- Confidentiality
  - Especially with transport encryption
- Authentication and access control
  - I know it's you. I REALLY know it's you.
- Non-repudiation
  - You said it. You can't deny it.
- Integrity
  - Tamper-proof

Cryptography Concepts



ISBN-13: 978-0130889768
ISBN-10: 0130889768



ISBN-13: 978-0691141756
ISBN-10: 9780691141756

Cryptography Concepts

Cryptographic terms

- Plaintext
  - An unencrypted message (in the clear)
- Ciphertext
  - An encrypted message
- Cipher
  - The algorithm used to encrypt and/or decrypt
- Cryptanalysis
  - The art of cracking encryption
  - Researchers are constantly trying to find weaknesses in ciphers
  - A mathematically flawed cipher is bad for everyone

Cryptography Concepts

Cryptographic keys

- Keys
  - Add the key to the cypher to encrypt
  - Larger keys are ostensibly more secure
- Some encryption methods use one key
  - Some use more than one key
  - Every method is a bit different

Cryptography Concepts

Confusion

- Encryption is based on confusion and diffusion
- Confusion
  - The encrypted data is drastically different than the plaintext
  - The process should be non-linear, with no discernible patterns

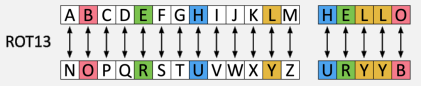
Diffusion

- Change one character of the input, and many characters change of the output

Cryptography Concepts

Security through obscurity

- Security should exist, even if the attacker knows everything about the system
  - Encryption key would be the only unknown
  - Cryptography is security through secrecy
- Substitution Cipher (Caesar cipher)
  - Substitute one letter with another
  - ROT13 - “URYYB” is “HELLO”



- Hack these ciphers with frequency analysis or brute force
  - If you know how the system works, you can decrypt it

Cryptography Concepts

Random numbers

- Cryptography relies on randomness
  - Used to generate keys, salt hashes, and much more
- Random number generation
  - It's very difficult to create true randomness with a program
  - Usually includes some type of natural input
  - Mouse movements, atmospheric noise, lava lamp
- Pseudo-randomness doesn't rely on the natural world
  - Approximate true randomness
  - Based on a starting seed

Cryptography Concepts

App development and cryptography

- Developers don't need to be cryptographers
  - They write to an API (application programming interface)
  - Crypto modules
- The API library does all of the heavy lifting
  - Send plaintext into the box, get ciphertext back
  - No extra programming required
- The Windows software library is the
- Cryptographic Service Provider (CSP)
  - The Microsoft CryptoAPI is the bridge between the application and the CSP

Symmetric and Asymmetric Encryption

Symmetric and Asymmetric Encryption

Symmetric encryption

- A single, shared key
  - Encrypt with the key
  - Decrypt with the same key
  - If it gets out, you'll need another key
- Secret key algorithm
  - A shared secret
- Doesn't scale very well
  - Can be challenging to distribute
- Very fast to use
  - Less overhead than asymmetric encryption
  - Often combined with asymmetric encryption

Symmetric and Asymmetric Encryption

Asymmetric encryption

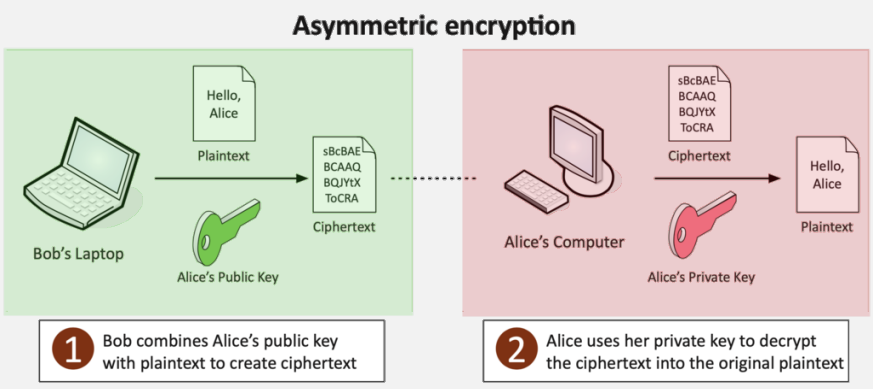
- Public key cryptography
  - Two keys
- Private key
  - Keep this private
- Public key
  - Anyone can see this key
  - Give it away
- The private key is the only key that can decrypt data encrypted with the public key
  - You can't derive the private key from the public key

Symmetric and Asymmetric Encryption

The key pair

- Asymmetric encryption
  - Public Key Cryptography
- Key generation
- Build both the public and private key at the same time
  - Lots of randomization
  - Large prime numbers
  - Lots and lots of math
- Everyone can have the public key
  - Only Alice has the private key

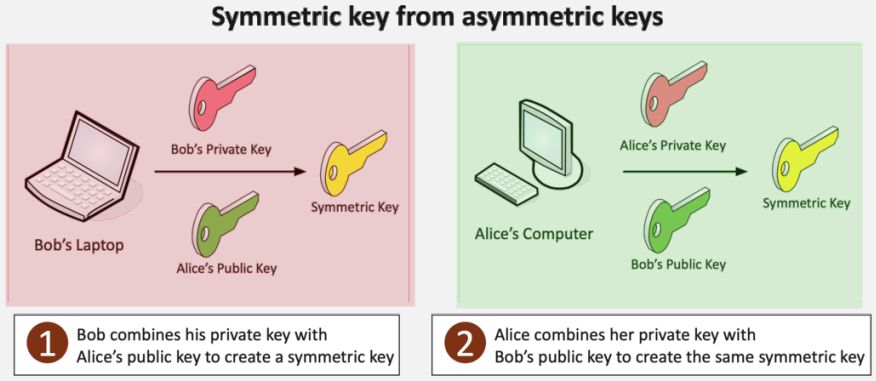
Symmetric and Asymmetric Encryption



Symmetric and Asymmetric Encryption

Symmetric key from asymmetric keys

- Use public and private key cryptography to create a symmetric key
  - Math is powerful



Symmetric and Asymmetric Encryption

Elliptic curve cryptography (ECC)

- Asymmetric encryption
  - Need large integers composed of two or more large prime factors
- Instead of numbers, use curves!
  - Uses smaller keys than non-ECC asymmetric encryption
  - Smaller storage and transmission requirements
  - Perfect for mobile devices

Hashing and Digital Signatures

Hashing and Digital Signatures

Hashes

- Represent data as a short string of text
  - A message digest
- One-way trip
  - Impossible to recover the original message from the digest
  - Used to store passwords / confidentiality
- Verify a downloaded document is the same as the original
  - Integrity
- Can be a digital signature
  - Authentication, non-repudiation, and integrity
- Will not have a collision (hopefully)
  - Different messages will not have the same hash

Hashing and Digital Signatures

Collision

- Hash functions
  - Take an input of any size
  - Create a fixed size string
  - Message digest, checksum
- The hash should be unique
  - Different inputs should never create the same hash
  - If they do, it's a collision
- MD5 has a collision problem
  - Found in 1996 - Don't use MD5

Hashing and Digital Signatures

Practical hashing

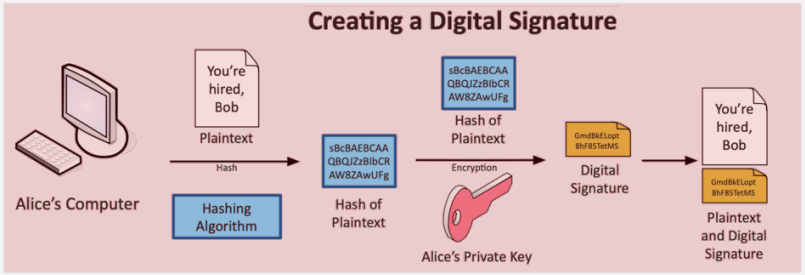
- Verify a downloaded file
  - Hashes may be provided on the download site
  - Compare the downloaded file hash with the posted hash value
- Password storage
  - Instead of storing the password, store the hash
  - Compare hashes during the authentication process
  - Nobody ever knows your actual password

Hashing and Digital Signatures

Digital signatures

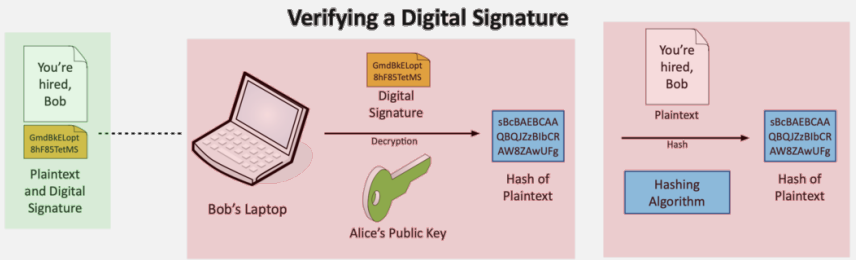
- Prove the message was not changed
  - Integrity
- Prove the source of the message
  - Authentication
- Make sure the signature isn't fake
  - Non-repudiation
- Sign with the private key
  - The message doesn't need to be encrypted
  - Nobody else can sign this (obviously)
- Verify with the public key
  - Any change in the message will invalidate the signature

Symmetric and Asymmetric Encryption



1. Alice creates a hash of the original plaintext
2. Alice encrypts the hash with her private key
3. The encrypted hash (digital signature) is included with the plaintext

Symmetric and Asymmetric Encryption



1. Bob decrypts the digital signature to obtain the plaintext hash
2. Bob hashes the plaintext and compares it to the decrypted hash

Randomizing Cryptography

Randomizing Cryptography

Initializing vectors

- A type of nonce
  - Used for randomizing an encryption scheme
  - The more random the better
- Used in encryption ciphers, WEP, and older SSL implementations

Randomizing Cryptography

Cryptographic nonce

- Arbitrary number
  - Used once
  - “For the nonce” - For the time being
- A random or pseudo-random number
  - Something that can’t be reasonably guessed
  - Can also be a counter
- Use a nonce during the login process
  - Server gives you a nonce
  - Calculate your password hash using the nonce
- Each password hash sent to the host will be different, so a replay won’t work

Randomizing Cryptography

Salt

- A nonce most commonly associated with password randomization
  - Make the password hash unpredictable
- Password storage should always be salted
  - Each user gets a different salt
- If the password database is breached, you can’t correlate any passwords
  - Even users with the same password have different hashes stored

Obfuscation

Obfuscation

Obfuscation

- The process of making something unclear
  - It's now much more difficult to understand
- But it's not impossible to understand
  - If you know how to read it
- Make source code difficult to read
  - But it doesn't change the functionality of the code
- Hide information inside of an image
  - Steganography

Obfuscation

Substitution ciphers

- Simple substitution
  - Plaintext alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - Ciphertext alphabet: ZEBRASCDFGHIJKLMNOPQTUVWXY
  - WE ARE DISCOVERED enciphers to: VA ZOA RFPBLUAOAR
- Caesar cipher
  - Substitute one letter with another at a fixed position
- Hack these ciphers with frequency analysis or brute force
  - If you know the system, you can decrypt it

Obfuscation

XOR (Exclusive OR)

- Outputs true when inputs differ
- Used extensively in cryptography
  - With a truly random key, the results are theoretically unbreakable

x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0

```
  01010111 01101001 01101011 01101001  Plaintext
⊕ 11110011 11110011 11110011 11110011  Key
  10100100 10011010 10011000 10011010  Ciphertext

  10100100 10011010 10011000 10011010  Ciphertext
⊕ 11110011 11110011 11110011 11110011  Key
  01010111 01101001 01101011 01101001  Plaintext
```

Weak Encryption

Weak Encryption

The strength of encryption

- Strong cryptography vs. weak cryptography
  - It's all relative
- Practically everything can be brute forced
  - Try every possible key
- Strong algorithms have been around for a while
  - That's part of the reason that they are strong
  - Wired Equivalent Privacy (WEP) had design flaws
- Strong algorithms
  - PGP, AES
- Weak algorithms
  - DES (56-bit keys), WEP (design flaw)

Weak Encryption

Give weak keys a workout

- A weak key is a weak key
  - By itself, it's not very secure
- Make a weak key stronger by performing multiple processes
  - Hash a password. Hash the hash of the password. And continue...
  - Key stretching, key strengthening
- Brute force attacks would require reversing each of those hashes
  - The attacker has to spend much more time, even though the key is small

Cryptographic Keys

Cryptographic Keys

Cryptographic keys

- There's very little that isn't known about the cryptographic process
  - The algorithm is usually a known entity
  - The only thing you don't know is the key
- The key determines the output
  - Encrypted data, hash value, digital signature
- Keep your key private!
  - It's the only thing protecting your data

Cryptographic Keys

Key strength

- Larger keys tend to be more secure
  - Prevent brute-force attacks
  - Attackers can try every possible key combination
- Symmetric encryption
  - 128-bit or larger symmetric keys are common
  - These numbers get larger as time goes on
- Asymmetric encryption
  - Complex calculations of prime numbers
  - Larger keys than symmetric encryption
  - Common to see key lengths of 3,072 bits or larger

Cryptographic Keys

Key exchange

- A logistical challenge
  - How do you transfer an encryption key across an insecure medium without having an encryption key?
- Out-of-band key exchange
  - Don't send the symmetric key over the 'net
  - Telephone, courier, in-person, etc.
- In-band key exchange - It's on the network
  - Protect the key with additional encryption
  - Often uses asymmetric encryption to deliver a symmetric key

Cryptographic Keys

Real-time encryption/decryption

- There's a need for fast security
  - Without compromising the security part
- Share a symmetric session key using asymmetric encryption
  - Client encrypts a random (symmetric) key with a server's public key
  - The server decrypts this shared key, uses it to encrypt data
- Implement session keys carefully
  - Need to be changed often (ephemeral keys)
  - Need to be unpredictable

Steganography

Steganography

Steganography

- Greek for “concealed writing” - security through obscurity
- Message is invisible - But it’s really there
- The covertext - The container document or file

Steganography

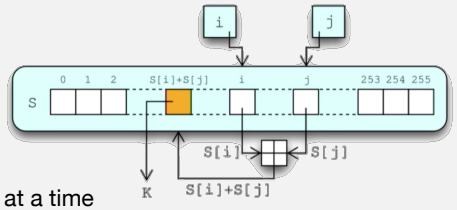
Common steganography techniques

- Network based
  - Embed messages in TCP packets
- Use an image
  - Embed the message in the image itself
- Invisible watermarks
  - Yellow dots on printers

Stream and Block Ciphers

Stream and Block Ciphers

Stream ciphers

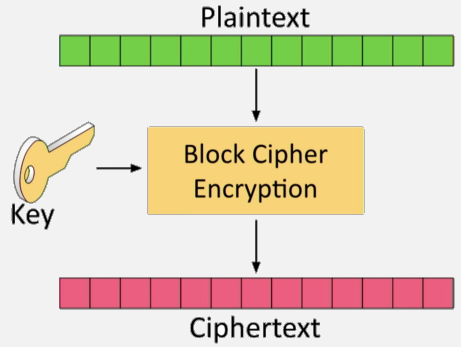


- Used with symmetric encryption
  - Not used in asymmetric encryption
- Encryption is done one bit or byte at a time
  - High speed, low hardware complexity
- The starting state should never be the same twice
  - Key is often combined with an initialization vector (IV)

Stream and Block Ciphers

Block ciphers

- Symmetric encryption - Similar to stream ciphers
- Encrypt fixed-length groups
  - Often 64-bit or 128-bit blocks
  - Pad added to short blocks
  - Each block is encrypted or decrypted independently
- Block cipher modes of operation
  - Avoid patterns in the encryption
- Many different modes to choose from



Block Cipher Modes

Block Cipher Modes

Block Cipher mode of operation

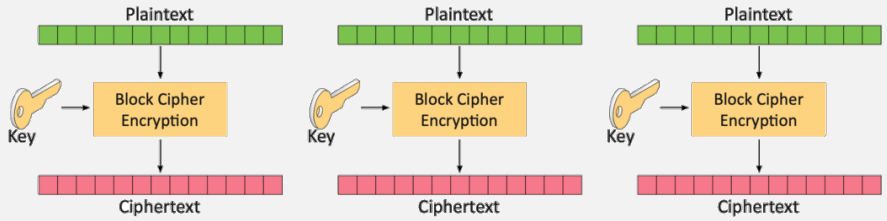
- Encrypt one fixed-length group of bits at a time
  - A block
- Mode of operation
  - Defines the method of encryption
  - May provide a method of authentication
- The block size is a fixed size
  - Not all data matches the block size perfectly
  - Split your plaintext into smaller blocks
  - Some modes require padding before encrypting

Block Cipher Modes

ECB (Electronic Code book)

- The simplest encryption mode
  - Too simple for most use cases
- Each block is encrypted with the same key
  - Identical plaintext blocks create identical ciphertext blocks

ECB (Electronic Code book) cipher mode

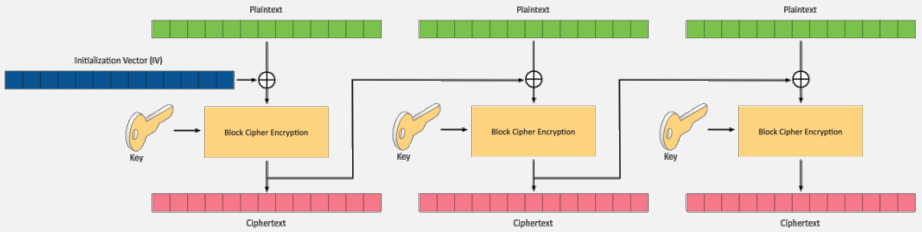


Block Cipher Modes

CBC (Cipher Block Chaining)

- A popular mode of operation
  - Relatively easy to implement
- Each plaintext block is XORed with the previous ciphertext block
  - Adds additional randomization
  - Use an initialization vector for the first block

CBC (Cipher Block Chaining) cipher mode

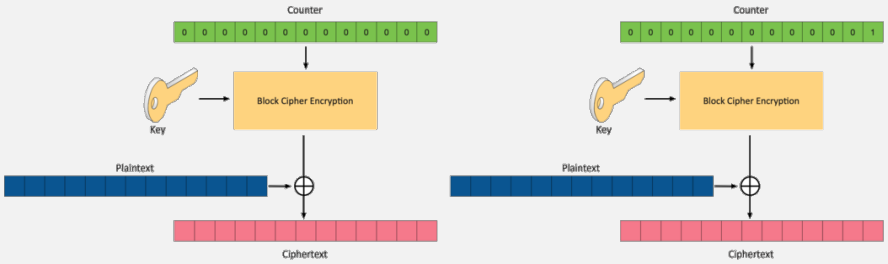


Block Cipher Modes

CTR (Counter)

- Block cipher mode / acts like a stream cipher
  - Encrypts successive values of a “counter”
- Plaintext can be any size, since it’s part of the XOR
  - i.e., 8 bits at a time (streaming) instead of a 128-bit block

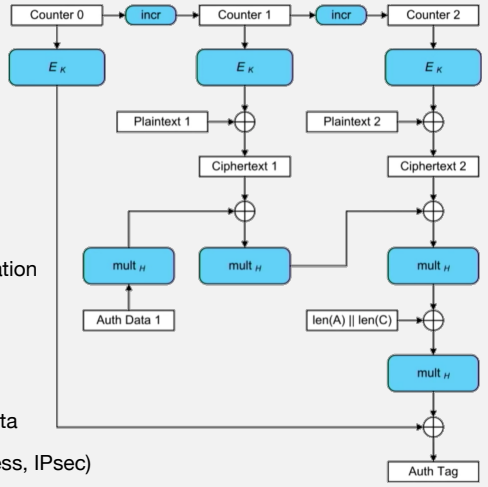
CTR (Counter) cipher mode



Block Cipher Modes

GCM (Galois/Counter Mode)

- Encryption with authentication
  - Authentication is part of the block mode
  - Combines Counter Mode with Galois authentication
- Minimum latency, minimum operation overhead
  - Very efficient encryption and authentication
- Commonly used in packetized data
  - Network traffic security (wireless, IPsec)
  - SSH, TLS



States of Data

States of Data

Data in-transit

- Data transmitted over the network
  - Also called data in-motion
- Not much protection as it travels
  - Many different switches, routers, devices
- Network-based protection - Firewall, IPS
- Provide transport encryption
  - TLS (Transport Layer Security)
  - IPsec (Internet Protocol Security)

States of Data

Data at-rest

- The data is on a storage device
  - Hard drive, SSD, flash drive, etc.
- Encrypt the data
  - Whole disk encryption, database encryption
  - File or folder-level encryption
- Apply permissions - Access control lists
  - Only authorized users can access the data

States of Data

Data in-use

- The data is in memory
  - System RAM, CPU registers and cache
- The data is almost always decrypted
  - Otherwise, you couldn't do anything with it
- The bad guys can pick the decrypted information out of RAM
  - A very attractive option
- Target Corporation breach - November 2013
  - 110 million credit cards
  - Data in-transit encryption and data at-rest encryption
  - Bad guys picked the credit card numbers out of the point-of-sale RAM (data in-use)

Perfect Forward Secrecy

Perfect Forward Secrecy

Traditional web server encryption

- SSL/TLS uses encryption keys to protect web server communication
  - Traditionally, this has been based on the web server's RSA key pair
  - One key that encrypts all symmetric keys
- This server's private key can rebuild everything
  - If you capture all of the traffic, you can decrypt all of the data
- One point of failure for all of your web site encryption

Perfect Forward Secrecy

Perfect Forward Secrecy (PFS)

- Change the method of key exchange
  - Don't use the server's private RSA key
- Elliptic curve or Diffie-Hellman ephemeral
  - The session keys aren't kept around
- Can't decrypt with the private server key
  - Every session uses a different private key
- PFS requires more computing power
  - Not all servers choose to use PFS
- The browser must support PFS

Common Cryptography Use Cases

Common Cryptography Use Cases

Finding the balance

- Low power devices
  - Mobile devices, portable systems
  - Smaller symmetric key sizes
  - Use elliptic curve cryptography (ECC) for asymmetric encryption
- Low latency
  - Fast computation time
  - Symmetric encryption, smaller key sizes
- High resiliency
  - Larger key sizes
  - Encryption algorithm quality
  - Hashing provides data integrity

Common Cryptography Use Cases

Use cases

- Confidentiality
  - Secrecy and privacy
  - Encryption (file-level, drive-level, email)
- Integrity
  - Prevent modification of data
  - Validate the contents with hashes
  - File downloads, password storage
- Non-Repudiation
  - Confirm the authenticity of data
  - Digital signature provides both integrity and non-repudiation

Common Cryptography Use Cases

Use cases

- Obfuscation
  - Modern malware
  - Encrypted data hides the active malware code
  - Decryption occurs during execution
- Authentication
  - Password hashing
  - Protect the original password
  - Add salts to randomize the stored password hash
- Resource vs. security constraints
  - An ongoing battle
  - Browser support vs. supported encryption
  - VPN software support vs. supported algorithms

Symmetric Algorithms

Symmetric Algorithms

AES (Advanced Encryption Standards)

- US Federal Government Standard
  - FIPS 197 in 2001
  - It took five years to standardize on this!
  - Developed by two Belgian cryptographers
    - Joan Daemen and Vincent Rijmen
- 128-bit block cipher - 128-, 192-, and 256-bit keys
- Used in WPA2 - Powerful wireless encryption

Symmetric Algorithms

DES

- Data Encryption Standard - DES and Triple DES
- Developed between 1972 and 1977 by IBM for the NSA
  - One of the Federal Information Processing Standards (FIPS)
- 64-bit block cipher
  - 56-bit key (very small in modern terms)
- Easily brute-forced with today's technology

Symmetric Algorithms

RC4

- Rivest Cipher 4 - Ron Rivest (Ron's Code 4)
- Part of the ill-fated WEP standard
  - Also part of SSL, but removed from TLS
- RC4 has "biased output"
  - If the third byte of the original state is zero and the second byte is not equal to two, then the second output byte is always zero
- Not common to see RC4 these days
  - WPA2 moved to AES

Symmetric Algorithms

Blowfish and Twofish

- Blowfish
  - Designed in 1993 by Bruce Schneier
  - 64-bit block cipher, variable length key (1 to 448 bits)
  - No known way to break the full 16 rounds of encryption
  - One of the first secure ciphers not limited by patents
- Twofish
  - Successor to Blowfish
  - 128-bit block size, key sizes up to 256
  - Designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson, Stefan Lucks, Tadayoshi Kohno, and Mike Stay
  - No patent, public domain

Asymmetric Algorithms

Asymmetric Algorithms

Diffie-Hellman key exchange

- A key exchange method
  - Over an insecure communications channel
- Published in 1976
  - Witfield Diffie and Martin Hellman (and Ralph Merkle)
- DH does not itself encrypt or authenticate
  - It's an anonymous key-agreement protocol
- Used for Perfect Forward Secrecy
  - Ephemeral Diffie-Hellman (EDH or DHE)
  - Combine with elliptic curve cryptography for ECDHE

m = 11292367, e = 94321, x = 08041115, p*q = m, n = (p-1)(q-1), d = e^{-1} mod n
n = (2860)(3946) = 11285560

n = 11285560

d = e^{-1} mod n = 94321^{-1} mod 11285560 → 6327241

d = 6327241
σ = x^d = $08041115^{6327241}$ mod 11292367 = 8338987

Asymmetric Algorithms

RSA

- Ron Rivest, Adi Shamir, and Leonard Adelman
  - Published the RSA cipher in 1977
- The first practical public-key cryptography system
  - Encrypt, decrypt, digital signatures
  - You must know the factors to decode
- Now released into the public domain
  - Used extensively for web site encryption and digital rights management

Asymmetric Algorithms

DSA (Digital Signature Algorithm)

- A standard for digital signatures
  - Modifies Diffie-Hellman for use in digital signatures
- A Federal Information Processing Standard for digital signatures
- Combine with elliptic curve cryptography
  - Fast and efficient digital signatures - ECDSA

Asymmetric Algorithms

Elliptic curve cryptography (ECC)

- Used for encryption, digital signatures, pseudo-random generators, and more
- Asymmetric encryption
  - Traditionally need large integers composed of two or more large prime factors
- Instead of numbers, use curves!
  - Just as infeasible to find the discrete logarithm of a random elliptic curve element with respect to a publicly known base point
  - Uses smaller keys than non-ECC encryption
  - Elliptic Curve Digital Signature Algorithm (ECDSA)

Asymmetric Algorithms

PGP (Pretty Good Privacy) and GPG

- Popular asymmetric encryption
  - Created by Phil Zimmerman in 1991
  - "Why I Wrote PGP" http://professormesser.link/pgp
  - Commercial software
  - Owned by Symantec
- Open standard - OpenPGP (RFC 4880)
  - Implemented as software: Gnu Privacy Guard (GPG)
    - Linux, Windows, Mac OS, others
    - Very compatible with commercial PGP

Hashing Algorithms

Hashing Algorithms

MD5 Message Digest Algorithms

- Designed by Ronald Rivest
  - One of the "fathers" of modern cryptography
- First published: April 1992
  - Replaced MD4
  - 128-bit hash value
- 1996: Vulnerabilities found
  - Not collision resistant
- December 2008: Researchers created CA certificate that appeared legitimate when MD5 is checked
  - Built other certificates that appeared to be legit and issued by RapidSSL

Hashing Algorithms

Secure Hash Algorithm (SHA)

- Developed by the National Security Agency (NSA)
  - A US Federal Information Processing Standard
- SHA-1
  - Widely used - 160-bit digest
  - 2005: Collision attacks published
- SHA-2
  - The preferred SHA variant
  - Up to 512-bit digests
  - SHA-1 is now retired for most US Government use

Hashing Algorithms

HMAC

- Hash-based Message Authentication Code
  - Combine a hash with a secret key
  - e.g., HMAC-MD5, HMAC-SHA1
- Verify data integrity and authenticity
  - No fancy asymmetric encryption required
- Used in network encryption protocols
  - IPsec, TLS

Hashing Algorithms

RIPEMD

- A family of message digest algorithms
  - RACE Integrity Primitives Evaluation Message Digest
- RACE
  - Research and Development in Advanced Communications Technologies in Europe
  - Created to help with Integrated Broadband Communications in Europe
  - Centralized cryptographic standards and management
- Original RIPEMD was found to have collision issues (2004)
  - Effectively replaced with RIPEMD-160 (no known collision issues)
  - Based upon MD4 design but performs similar to SHA-1
  - RIPEMD-128, RIPEMD-256, RIPEMD-320

Key Stretching Algorithms

Key Stretching Algorithms

Give weak keys a workout

- A weak key is a weak key
  - By itself, it's not very secure
- Make a weak key stronger by performing multiple processes
  - Hash a password. Hash the hash of the password. And continue...
  - Key stretching, key strengthening
- Brute force attacks would require reversing each of those hashes
  - The attacker has to spend much more time, even though the key is small

Key Stretching Algorithms

Key stretching libraries

- Already built for your application
  - No additional programming involved
- bcrypt
  - Generates hashes from passwords
  - An extension to the UNIX crypt library
  - Uses Blowfish cipher to perform multiple rounds of hashing
- Password-Based Key Derivation Function 2 (PBKDF2)
  - Part of RSA public key cryptography standards (PKCS #5, RFC 2898)

CSF 434/534: Advanced Network and System Security

Week 12 - Review

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Sources: Professor Messer's CompTIA SY0-501 Security+ Course Notes