Critical Infrastructure Security: The Emerging Smart Grid

Cyber Security Lecture 2: Cryptographic Basics
Carl Hauser & Adam Hahn
Overview

- Crypto principles
- Historical ciphers
- Modern symmetric ciphers
- Modern asymmetric ciphers
- Cryptographic Hash Functions
- Authentication
  - Message authentication code
  - Digital signature
Overview

• Crypto principles
• Historical ciphers
• Modern symmetric ciphers
• Modern asymmetric ciphers
• Cryptographic Hash Functions
• Authentication
  – Message authentication code
  – Digital signature
Cryptosystem Overview

Plaintext -> Encryption -> Channel -> Decryption -> Plaintext

Block
Intercept
Modify
Fabricate
Elements of a Cryptosystem

Cryptosystem

Algorithms

Encryption

Channel

Decryption

Plaintext

Ciphertext

Plaintext

C = E(K, P)

C = E(K_E, P)

P = D(K, C)

P = D(K_D, C)
Threats to cryptosystems

• Threats (theoretic)
  – Brute-force attack – try every possible key until you can find *intelligible translation*
  – Cryptanalysis – leverage knowledge about crypto algorithms, and/or some knowledge about the ciphertext and plaintext

• Threats (real-world)
  – System implementation – it is difficult to correctly design, implement, and deploy cryptosystems.
  – Rubber-hose cryptography – human factors/coercion

• Attacker capabilities:
  – Cipher only – attacker only knows cipher text
  – Known plaintext – attacker knows plain and ciphertext
  – Chosen plaintext – attacker can pick cipher text to encrypt (important in public key crypto)
  – Chosen ciphertext - attacker can choose ciphertext to be decrypted by system
Cryptosystem Principles

• Kerckhoff's principle (more specific definition)
  – made assumption that attacker knows how the cryptosystem algorithms work, only the key must be protected

• Perfect secrecy
  – Cipher text conveys no information about plain text
  – Can’t do cryptanalysis or brute force attacks
Overview

• Crypto principles
• Historical ciphers
• Modern symmetric ciphers
• Modern asymmetric ciphers
• Cryptographic Hash Functions
• Authentication
  – Message authentication code
  – Digital signature
Substitution Ciphers

Plain  abcdefghijklmnopqrstuvwxyz
Cipher mnbvcxzasdfghjklpoiuytrewq

bob. i love you. alice

nkn. s gktc wky. mgsbc

Monoalphabetic encryption
Transposition Ciphers

THIS IS A MESSAGE TO SHOW HOW A COLUMNAR TRANSPOSITION WORKS

TSSOHOANIWHAASOLRSTOIMGHW
UTPIRSEEOAMROOKISTWCNASNS
One Time Pad

“Information-theoretically secure”

• Keys are never re-used (one-time)
• Achieving this is hard: first to generate such key material and second to share it with your correspondent(s)
One-time Pad (cont.)

• Example...
  
  Message(P):  A T T A C K T O M O R R O W
  
  Random Key (k):  Z R Q W X O M G D E G F B I

• Algorithm:  \( C_i = (P_i + k_i) \mod 26 \)

  Key Elements:  
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  K:  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

  Cipher Text(C):  Z K J W Z Y F U P S X W P E

• Attacker’s dilemma:
  –  Multiples keys can be found the decrypt to many different credible plaintexts!!!

Option 1
  Key 1:  Z R Q W X O M G D E G F B I
  Plaintext 1:  A T T A C K T O M O R R O W

Option 2
  Key 2:  W W W I G Y M B P Q N Y L L
  Plaintext 2:  D O N O T A T T A C K Y E T
Overview

- Crypto principles
- Historical ciphers
- Modern symmetric ciphers
- Modern asymmetric ciphers
- Cryptographic Hash Functions
- Authentication
  - Message authentication code
  - Digital signature
Modern Symmetric Encryption

- Symmetric – same key used for enciphering and deciphering
- Combine multiple “rounds” of Substitution and Transposition

- Approaches:
  - Block cipher
  - Stream cipher
Block Ciphers

• Breaks plaintext into $n$-bit blocks, and then encrypts each block individually

Plain Text

<table>
<thead>
<tr>
<th>n-bits</th>
<th>K</th>
<th>K</th>
<th>K</th>
<th>K</th>
<th>K</th>
<th>K</th>
<th>K</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
</tr>
</tbody>
</table>

Cipher Text

| n-bits | ... | ... | ... | ... | ... |

• Elements
  – Key - random string of bytes, (56 < # of bits < 256 bits)
  – Encryption algorithm - Continued rounds of:
    • Substitutions
    • Transpositions
  – Decryption algorithm – inverse process of encryption
DES / 3DES / AES

- DES – Data Encryption Standard
  - Approved by NBS (NIST) in 1977
  - Based on Horst Feistel’s Lucifer cipher
  - 56 bit keys (8 bits for parity)
  - S-box (substitution box) developed by NSA
  - Key size too small for modern use

- 3DES
  - Implemented to extend life of DES
  - Uses three keys ($k_1, k_2, k_3$) to provide more security
  - Not meant to be long term solution!

- Advanced Encryption Standard (AES)
  - Won NIST sponsored competition for DES replacement in 2001, called AES
  - Really called “Rijndael cipher”, developed by Joan Daemen and Vincent Rijmen
  - Benefits
    - Performance, security

- Improved key sizes
  - 128, 192, 256 bits
Advanced Encryption Standard

128 bits

Input

128/192/256 bits

Cycle 1

Cycle 2

Cycle 10/12/14

Output

Add round key

Byte substitution
Shift row
Mix columns
Add round key
How secure is DES/AES?

• Larger key length
  – 56 bit key = $2^{56}$ (7.2 x $10^{16}$) possible keys
  – 128 bit key = $2^{128}$ (3.4 x $10^{38}$) possible keys
  – 256 bit key = $2^{256}$ (1.2 x $10^{77}$) possible keys

• How big is $2^{128}$???
  – 340,282,366,920,938,463,463,374,607,431,768,211,456
    • Or
  – “three hundred forty undecillion, two hundred eighty-two decillion, three hundred sixty-six nonillion, nine hundred twenty octillion, nine hundred thirty-eight septillion, four hundred sixty-three sextillion, four hundred sixty-three quintillion, three hundred seventy-four quadrillion, six hundred seven trillion, four hundred thirty-one billion, seven hundred sixty-eight million, two hundred eleven thousand, four hundred fifty-six”

Source: http://blog.agilebits.com/2013/03/09/guess-why-were-moving-to-256-bit-aes-keys/
Brute force decryption time

• Currently 33,862,700 GFlops/s or $3.4 \times 10^{16}$
  – Assume 1000 operations per decryption
  – Decryptions /second: $3.4 \times 10^{13}$
  – Seconds in year: $3.15 \times 10^7$
  – Decryptions/year: $\sim 1.1 \times 10^{21}$
    • $3.13 \times 10^{13}$ microseconds/year

• Decryption times:
  – DES (56 bit) = $2^{56}/(3.4\times10^{13}) = \sim 35$ minutes
  – AES (128 bit) = $2^{128}/(1.1\times10^{21}) = 3.1 \times 10^{17}$ years
  – AES (256 bit) = $2^{256}/(1.1\times10^{21}) = 1.1 \times 10^{56}$ years
Block Chaining

• How do we handle messages longer than the block size
• Example
  – Electronic Code Book (ECB) – each block encrypted independently from previous
    • Similar blocks encrypt to same output
  – Cipher Block Chaining (CBC) – each block XORs the output of the previous block
    • Similar blocks encrypt to different output
Problems with Symmetric Encryption

Key distribution

Scalability

\[ \frac{N(N - 2)}{2} \]

Existing Users

New Keys to Be Added

New User
Overview

• Crypto principles
• Historical ciphers
• Modern symmetric ciphers
• Modern asymmetric ciphers
• Cryptographic Hash Functions
• Authentication
  – Message authentication code
  – Digital signature
Fundamental Idea

• Encryption and Decryption use different keys
  – Also called “public key” crypto
• Invented in mid 1970s
• A completely different viewpoint on messages
  – Instead of consisting of symbols that could be substituted or rearranged, look at messages as numbers that can added, subtracted, multiplied, divided, exponentiated, etc.
• Based on one-way functions
  – Discrete logarithm
  – Integer factoring
  – Elliptic curve arithmetic
RSA Cryptosystem

Encryption

\[ P = 12 \]

\[ C = P^e \mod n \]

\[ C = 17 \]

Decryption

\[ P = C^d \mod n \]

\[ P = 12 \]

\[ C = (P^e)^d \mod n = (P^d)^e \mod n \]
RSA Keys

\[(e, n) = (5, 35) \quad (d, n) = (29, 35)\]

- \( n \): a product of two primes, \( p \) and \( q \)
- \( >200 \) digits long (512 bits)
- \( e \): relatively prime to \( (p - 1)(q - 1) \)
- \( d \): \( e \times d = 1 \mod (p - 1)(q - 1) \)
  \[5 \times d = 1 \mod 24\]
is a public key; anyone, including Trudy can encrypt with it; (29,35) is Bob's private key – only he is able to decrypt messages created with the public key

Select two primes: \( p = 5, q = 7 \)
\[ n = pq = 35 \]
\[ e = 5, \text{ relatively prime to } (p-1)(q-1) = 24 \]
\[ d = 29, \text{ such that } e \times d = 1 \mod (p - 1)(q - 1) \]

\[ 12^5 \mod 35 = 17 \]
\[ 17^{29} \mod 35 = 2 \]
Asymmetric Encryption Challenges

- Asymmetric cryptosystems are many times more computationally expensive than good symmetric systems

<table>
<thead>
<tr>
<th>Device</th>
<th>Supported Algorithms</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel AT98SC016CU</td>
<td>SHA-1, SHA-256, RSA, RSA-CRT, RSA-DS, ECDSA, FIPS-140 Random number generator</td>
<td>RSA 2048-bit signature generation in less than 360ms RSA 2048-bit signature verification in less than 60ms</td>
</tr>
<tr>
<td>Broadcom BCM 5823</td>
<td>HMAC-SHA-1, RSA</td>
<td>550 1024-bit RSA transactions per second</td>
</tr>
<tr>
<td>SafeXcel 1741</td>
<td>HMAC-SHA-1, RSA</td>
<td>RSA 1024-bit sign in 8.4ms RSA 1024-bit verify in 0.85ms</td>
</tr>
<tr>
<td>SafeXcel 1840</td>
<td>HMAC-SHA-1, RSA</td>
<td>RSA 1024-bit sign in 0.82ms RSA 1024-bit verify in 0.26ms</td>
</tr>
</tbody>
</table>

- Solution
  - Use an asymmetric system to agree on a symmetric key
  - Encrypt large amounts of data using the symmetric key

Overview

• Crypto principles
• Historical ciphers
• Modern symmetric ciphers
• Modern asymmetric ciphers
• Cryptographic Hash Functions
• Authentication
  – Message authentication code
  – Digital signature
Cryptographic Hash Function

• Hash function: one way function mapping a variable length string to fixed length digest
• Common functions: SHA(1,2,3), MD5
• Used for:
  – Digital Signatures
  – HMACs
  – Password storage

• Three key requirements:
  – Preimage resistant
  – Second preimage resistant
  – Collision Resistant
Hash functions examples

• Unix commands
  
  # echo "This is a test message to demonstrate hash functions" | sha1sum  
  > 799772e0410af26e43ee279af9f630150c7bb8bb

  # echo "This is a test message to demonstrate hash functions." | sha1sum  
  > 5bd891f72d006b2eaff8b0ad36457fe7cce6f21c

  # echo "This is a tset message to demonstrate hash functions" | sha1sum  
  > d3e266656a24a986e6303ecccab4474447b887def

  # echo "This is a test message to demonstrate hash functions" | sha512sum  
  >72a26cb6e857cecbaf6b2b88bfaebd680de0551e091e87e7f822c10933ce58c696356d88124a114ff1eec089a3a2e8f2cecb0916c567f820c12d07c598d790c5
SHA-1

- Message
- Padding
- Message length
- 512 bits
- Block
- Block
- Block
- Block
- Block
- 160 bits
- (5 registers)
- Compression
- $H_{SHA}$
- $H_{SHA}$
- $H_{SHA}$
- $H_{SHA}$
- $H_{SHA}$
- $H$
- 160 bits
Overview

- Crypto principles
- Historical ciphers
- Modern symmetric ciphers
- Modern asymmetric ciphers
- Cryptographic Hash Functions
- Authentication
  - Message authentication code
  - Digital signature
Hash Message Authentication Code (HMAC)

- MAC – cryptographic digest appended to a communication
  - Cannot be manipulated by attack without the key used to create it

- HMAC - MAC based on hash functions
  - Stronger security properties
    - Usually faster than traditional symmetric crypto algs (e.g., DES)
    - Less vulnerable to collisions than just hash functions
      - Used with modern hashes (MD5, SHA, etc)

- HMAC output is appended to message
  - Receiver verifies message by computing the HMAC and compare with the appended HMAC

- HMAC basic example:
  - $\text{HMAC}(k, m) = H(K \mid m)$
    - Length extension attack

- Better example
  - $\text{HMAC}(k, m) = H(K \mid H(K \mid m))$
HMAC usage

Alice

Key (k) → Plaintext → HMAC Function → HMAC

Network

Bob

Key (k) → Plaintext → HMAC Function → HMAC

Yes = OK
No = modified/spoofed
Overview

• Crypto principles
• Historical ciphers
• Modern symmetric ciphers
• Modern asymmetric ciphers
• Cryptographic Hash Functions
• Authentication
  – Message authentication code
  – Digital signature
Digital Signatures

• Utilize public key crypto in reverse, to provide authenticity
  – Sender encrypts (signs) with private key (usually a message hash)
  – Receive decrypts (verifies) with public key
• Upon successful decryption (verify), recipient knows message came from someone holding the corresponding private key
• Anyone can verify (because verifying uses the public key)
Digital Signatures

Alice

Plaintext

Hash

Hash

Key (PRₐ)

Sign

DS

Network

Plaintext

Bob

Key (PUₐ)

Hash

Hash

???
Authenticating *Principals*
Trusted third parties

• Some entity to attest to the relationship between the cryptographic keys and the real-world principals to which they belong

• Examples:
  – Key server (symmetric keys)
  – Certificate authority (asymmetric keys)

• About “trust”:
  – trust is reliance on another to perform as expected. To say something is “trusted” is not a statement about its reliability but rather a statement about risk!
Needham-Schroeder Protocol using Symmetric Keys

The Key Server is the trusted third party.
PKI: Public Key Infrastructure

• A system of protocols and third parties to make public-key encryption systems useful in practice

• Certificates
  – Help develop “root of trust” for issuing public keys
  – Based on trusted certificate authorities (CA)
    • E.g., Verisign, Comodo, GoDaddy
  – Challenges include key issuing and revocation
What are the trust assumption being made here?
Certificate Creation

1. Bob creates public/private keys
2. Prepares a certificate (unsigned) with information about identity (Bob), public key, and the CA
3. Submit certificate to CA
4. CA hashes certificate, signs it with CA private key, attaches signature to certificate
5. CA returns *signed* to the Bob
Certificate Usage

1. Alice request communication with Bob
2. Bob sends signed certificate to Alice
3. Alice (already has CA public key) creates hash of certificate, checks to see if the hash matches the CA signed part of certificate
4. If equal, cert has not been tampered with and public key in certificate can be trusted
5. If not equal, potentially spoofed/tampered communication!
Certificate Example

Certificate:
  Data:
  Version: 1 (0x0)
  Serial Number: 7829 (0ixe95)
  Signature Algorithm: md5WithRSAEncryption
  Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
  OU=Certification Services Division,
  CN=Thawte Server CA/emailAddress=server-certs@thawte.com
  Validity:
  Not Before: Jul 9 16:04:02 1998 GMT
  Not After: Jul 9 16:04:02 1999 GMT
  Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
  OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
  Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (1024 bit)
  Modulus (1024 bit):
  66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
  8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
  e8:35:1c:9e:27:52:7e:41:8f
  Exponent: 65537 (0x10001)
  Signature Algorithm: md5WithRSAEncryption
  ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:fc:8e:9c:67:
  68:9f
Certification Hierarchies
Crypto Warnings

• Strong crypto != strong security
  – Crypto usually bypassed not broken

• Vulnerabilities in
  – Design
  – Implementation
  – Installation

Avoid rolling your own crypto!!!

• I haven’t told you enough to implement ANY of this securely
  – There are MANY PITFALLS in the practicalities

• My best recommendation
  – Use well-researched, standard techniques
  – Use library implementations and pay attention to details about how to use them correctly
  – Investigate the literature for recent status of both the library and underlying technique
  – Consult an active expert
END