

CSC 405

Introduction to Computer Security

Topic 2. Basic Cryptography (Part II)

AES (Advanced Encryption Standard)

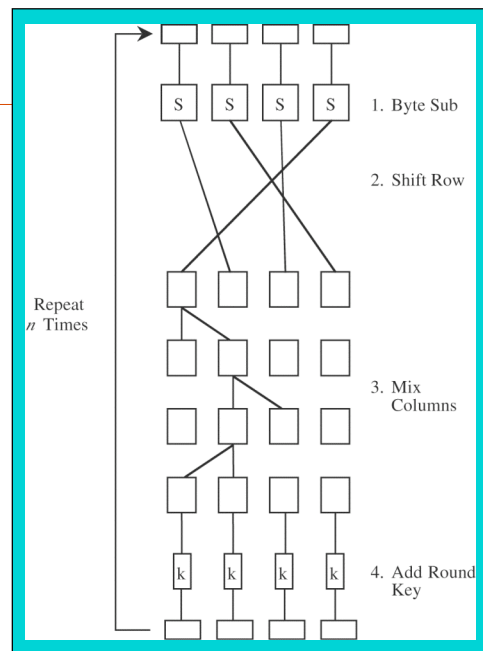
- January 1997: NIST called for AES contest
 - Requirements
 - Unclassified
 - Publicly disclosed
 - Available royalty-free for use worldwide
 - Symmetric block cipher, for blocks of 128 bits
 - Usable with key sizes of 128, 192, and 256 bits
- August 1998: 15 candidates submitted
- August 1999: 5 finalists
- Winning algorithm: Rijndael
 - Inventors: Vincent Rijmen and Joan Daemen (Dutch cryptographers)
 - The other four candidates are all security
 - Selection based on efficiency and implementation characteristics
- December 2001: AES adopted for use in the US government

Overview of AES

- Has a strong mathematical foundation
- Primarily use
 - Substitution, transposition, shift, XOR, and addition operations
- Use repeat rounds
 - 9 rounds for keys of 128 bits
 - 11 rounds for keys of 192 bits
 - 13 rounds for keys of 256 bits
- Rijndael can use any key length that is the multiple of 64
 - AES only recognizes 128, 192, and 256
- Rijndael is defined for blocks of 128, 192, and 256 bits
 - AES specifies 128 bit blocks

AES

- Each round consists of
 - Byte substitution
 - Shift row
 - Mix column
 - Add subkey




AES (Cont'd)

- Representation
 - 128 bits \Rightarrow 16 bytes \Rightarrow matrix $s[0,0]..s[3,3]$
- Byte substitution
 - Input b
 - Take the multiplicative inverse of b in $GF(2^8)$ defined by $P=x^8+x^4+x^3+x+1$
 - Ensure each value appears exactly once
 - XOR the result with 0x63 (0110 0011)
 - Help break up patterns

AES (Cont'd)

- Shift row
 - Row i is rotated left $(i - 1)$ bytes
 - Rijndael 256 bit blocks
 - Rows 3 and 4 are shifted an extra byte

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16



1	5	9	13
6	10	14	2
11	15	3	7
16	4	8	12

AES (Cont'd)

- Mix column
 - Each column is multiplied by a matrix
 - Arithmetic operations performed in GF(2⁸)

$$\begin{bmatrix} s'_{0,i} \\ s'_{1,i} \\ s'_{2,i} \\ s'_{3,i} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_{0,i} \\ s_{1,i} \\ s_{2,i} \\ s_{3,i} \end{bmatrix}$$

- Add subkey
 - XOR a variation of the key with the result so far

AES (Cont'd)

- Subkey generation
 - 128 bit key represented as four 32-bit words
 - $w_1 w_2 w_3 w_4$
 - Transformation of w_1 into w_1'
 - w_1 rotate one byte left
 - Byte substitution
 - XOR with a constant
 - The rest of the words are produced by XOR of the original word with w_1'
- First key is the original key
- Each later variation is generated from the previous one

Strength of AES

- Backed up by sound mathematical foundation
- Undergone extensive cryptanalysis by independent cryptographers
 - No flaw found

Public Key Cryptography (PKC)

- Requirements for Public-Key Algorithms
 - It is computationally easy to generate a pair of public key and private key
 - It is computationally easy to generate a ciphertext using the public key
 - It is computationally easy to decrypt the ciphertext using the private key
 - It is computationally infeasible to determine the private key from the public key
 - It is computationally infeasible to recover the message from the ciphertext and the public key

Trapdoor One-Way Function

- Essential requirement: **Trapdoor one-way function.**
- One-way function f
 - One-to-one mapping
 - $Y=f(X)$: easy
 - $X=f^{-1}(Y)$: infeasible
- Trapdoor one-way function
 - One-to-one mapping
 - $Y=f_k(X)$: easy if k and X are known
 - $X=f_k^{-1}(Y)$: easy if k and Y are known
 - $X=f_k^{-1}(Y)$: infeasible if Y is known but k is unknown.
- Designing public-key algorithm is to find appropriate trapdoor one-way function

Public-Key Cryptanalysis

- Brute-force attack
 - Try all possible keys
- Derivation of private key from public key
 - Try to find the relationship between the public key and the private key and compute the private key from the public one
- Probable-message attack
 - The public key is known
 - Encrypt all possible messages
 - Try to find a match between the ciphertext and one of the above encrypted messages

RSA (Rivest, Shamir, Adleman)

- The most popular one
- Support both public key encryption and digital signature
- Assumption/theoretical basis:
 - Factorization of large integers is hard
- Variable key length (usually 1024 bits)
- Variable plaintext block size
 - Plaintext must be “smaller” than the key
 - Ciphertext block size is the same as the key length

Modulo Operator

- Given any positive integer n and any integer a , we have $a = qn + r$, where $0 \leq r < n$ and $q = \lfloor a/n \rfloor$
 - We write $r = a \bmod n$
 - The remainder r is often referred to as a **residue**
 - Example:
 - $2 = 12 \bmod 5$
- Two integer a and b are said to be **congruent modulo n** if $a \bmod n = b \bmod n$
 - We write $a \equiv b \bmod n$
 - Example:
 - $7 \equiv 12 \bmod 5$

Modulo Operator (Cont'd)

- Properties of modulo operator
 - $a \equiv b \pmod n$ if $n|(a - b)$
 - $(a \pmod n) = (b \pmod n)$ implies $a \equiv b \pmod n$.
 - $a \equiv b \pmod n$ implies $b \equiv a \pmod n$.
 - $a \equiv b \pmod n$ and $b \equiv c \pmod n$ imply $a \equiv c \pmod n$.

Modular Arithmetic

- Observation:
 - The $(\pmod n)$ operator maps all integers into the set of integers $\{0, 1, 2, \dots, (n-1)\}$
- Modular addition
 - $[(a \pmod n) + (b \pmod n)] \pmod n = (a+b) \pmod n$
- Modular subtraction
 - $[(a \pmod n) - (b \pmod n)] \pmod n = (a - b) \pmod n$
- Modular multiplication
 - $[(a \pmod n) \times (b \pmod n)] \pmod n = (a \times b) \pmod n$

Totient Function

- Totient function $\phi(n)$: number of integers less than n and relatively prime to n
 - If p is prime, $\phi(p)=p-1$
 - If $n=p*q$, and p, q are primes, $\phi(n)=(p-1)(q-1)$
- Examples:
 - $\phi(7)=$ _____
 - $\phi(21)=$ _____

Euler's Theorem

- For relatively prime numbers a and n ,
 - $a^{\phi(n)} \equiv 1 \pmod{n}$
- Examples
 - $a=3, n=10, \phi(10)=$ _____, $3^{\phi(10)} \pmod{10} =$ _____
 - $a=2, n=11, \phi(11)=$ _____, $2^{\phi(11)} \pmod{11} =$ _____.

RSA Algorithm

- To generate key pair:
 - Pick large primes p and q
 - Let $n = p * q$, keep p and q to yourself!
 - For public key, choose e that is relatively prime to $\phi(n) = (p-1)(q-1)$
 - Let **pub** = $\langle e, n \rangle$
 - For private key, find d that is the multiplicative inverse of $e \bmod \phi(n)$, i.e., $e * d = 1 \bmod \phi(n)$
 - Let **pri** = $\langle d, n \rangle$

How Does RSA Work?

- Given **pub** = $\langle e, n \rangle$ and **priv** = $\langle d, n \rangle$
 - encryption: $c = m^e \bmod n, m < n$
 - decryption: $m = c^d \bmod n$
 - signature: $s = m^d \bmod n, m < n$
 - verification: $m = s^e \bmod n$

An Example

- Choose $p = 7$ and $q = 17$.
- Compute $n = p * q = \underline{\quad}$.
- Compute $\phi(n) = (p-1)(q-1) = \underline{\quad}$.
- Select $e = 5$, which is relatively prime to $\phi(n)$.
- Compute $d = \underline{77}$ such that $e * d = 1 \pmod{\phi(n)}$.
- Public key: $\langle \underline{\quad}, \underline{\quad} \rangle$
- Private key: $\langle \underline{\quad}, \underline{\quad} \rangle$
- Encryption: $19^5 \pmod{119} = 66$
- Decryption: $66^{77} \pmod{119} = 19$.

Why Does RSA Work?

- Given $\text{pub} = \langle e, n \rangle$ and $\text{priv} = \langle d, n \rangle$
 - $n = p * q, \phi(n) = (p-1)(q-1)$
 - $e * d = 1 \pmod{\phi(n)}$
 - $x^{e*d} = x \pmod{n}$
 - encryption: $c = m^e \pmod{n}$
 - decryption: $m = c^d \pmod{n} = m^{e*d} \pmod{n} = m \pmod{n}$
 $= m$ (since $m < n$)
 - digital signature (similar)

The Security of RSA

- Attacks against RSA
 - Brute force: Try all possible private keys
 - Can be defeated by using a large key space
 - Mathematical attacks
 - Factor n into $n=p*q$.
 - Determine $\phi(n)$ directly: equivalent to factoring n .
 - Determine d directly: at least as difficult as factoring n .
 - Timing attacks
 - Recover the private key according to the running time of the decryption algorithm.

Using PKC for Key Exchange

Goal: Use PKC to establish a shared symmetric key

Alice (Pub-A, Pri-A)

Bob (Pub-B, Pri-B)

————— E (Pub-B, K) —————>

Problem?

Using PKC for Key Exchange (Cont'd)

Alice (Pub-A, Pri-A)

Bob (Pub-B, Pri-B)

———— Sign (Pub-A, K) ———→

Problem?

Using PKC for Key Exchange (Cont'd)

Alice (Pub-A, Pri-A)

Bob (Pub-B, Pri-B)

———— E (Pub-B, K, Sig (Pub-A, K)) ———→

Problem?

Using PKC for Key Exchange (Cont'd)

Alice (Pub-A, Pri-A)

Bob (Pub-B, Pri-B)

———— E (Pub-B, K), Sig (Pub-A, E (Pub-B, K)) —————>

Public Key Certificates

To create Diana's certificate:

Diana creates and delivers to Edward:

Name: Diana
Position: Division Manager
Public key: 17EF83CA ...

Edward adds:

Name: Diana	hash value
Position: Division Manager	128C4
Public key: 17EF83CA ...	

Edward signs with his private key:

Name: Diana	hash value
Position: Division Manager	128C4
Public key: 17EF83CA ...	

Which is Diana's certificate.

Use known public key to establish unknown ones (Assume Edward's public key is known.)

To create Delwyn's certificate:

Delwyn creates and delivers to Diana:

Name: Delwyn
Position: Dept Manager
Public key: 3AB3882C ...

Diana adds:

Name: Delwyn	hash value
Position: Dept Manager	48CFA
Public key: 3AB3882C ...	

Diana signs with her private key:

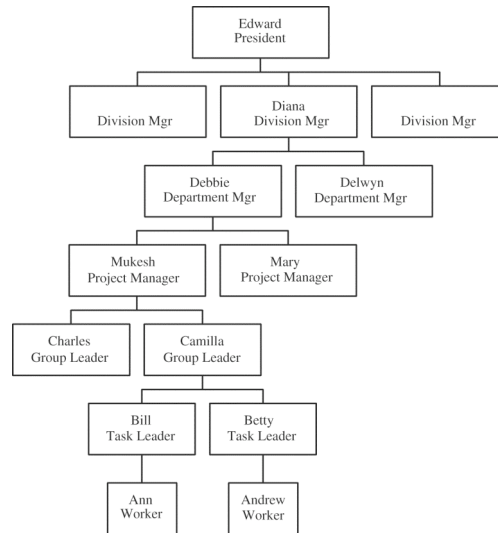
Name: Delwyn	hash value
Position: Dept Manager	48CFA
Public key: 3AB3882C ...	

And appends her certificate:

Name: Delwyn	hash value
Position: Dept Manager	48CFA
Public key: 3AB3882C ...	
Name: Diana	hash value
Position: Division Manager	128C4
Public key: 17EF83CA ...	

Which is Delwyn's certificate.

Hierarchy of Certificates



Certificate Authority (CA)

- A CA is a trusted node that maintains the public keys for all nodes
- Example
 - Edward on the previous slide
- CA's public key is well known
- A CA is involved in authenticating users' public keys by generating certificates