Advanced Encryption Standard

CSCI 454/554

Background

- clear a replacement for DES was needed
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow with small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001
AES Requirements

♦ secret key symmetric block cipher
♦ 128-bit data, 128/192/256-bit keys
♦ stronger & faster than Triple-DES
♦ active life of 20-30 years (+ archival use)
♦ provide full specification & design details
♦ both C & Java implementations
♦ NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

♦ initial criteria:
  – security – effort to practically cryptanalyse
  – cost – computational
  – algorithm & implementation characteristics
♦ final criteria
  – general security
  – software & hardware implementation ease
  – implementation attacks
    • Timing attacks and Power analysis
  – flexibility (in en/decrypt, keying, other factors)
AES Shortlist

♦ after testing and evaluation, shortlist in Aug-99:
  – MARS (IBM) - complex, fast, high security margin
  – RC6 (USA) - v. simple, v. fast, low security margin
  – Rijndael (Belgium) - clean, fast, good security margin
  – Serpent (Euro) - slow, clean, v. high security margin
  – Twofish (USA) - complex, v. fast, high security margin

♦ then subject to further analysis & comment

♦ saw contrast between algorithms with
  – few complex rounds verses many simple rounds
  – which refined existing ciphers verses new proposals

The AES Cipher - Rijndael

♦ designed by Rijmen-Daemen in Belgium
♦ has 128/192/256 bit keys, 128 bit data
♦ an iterative rather than feistel cipher
  – treats data in 4 groups of 4 bytes
  – operates an entire block in every round
♦ designed to be:
  – resistant against known attacks
  – speed and code compactness on many CPUs
  – design simplicity
Rijndael

- processes data as 4 groups of 4 bytes (state)
- The key is expanded into 44/52/60 (32-bit) words
- has 10/12/14 rounds in which state undergoes:
  - byte substitution (one S-box used on every byte)
  - shift rows (simple permutation)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with a portion of expanded key)
- all operations can be combined into XOR and table lookups - hence very fast & efficient

AES Data Structure
Encryption/decryption Operations

AES Round
**Byte Substitution**

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte \{95\} is replaced by row 9 col 5 byte
  - which is the value \{2A\}
- S-box is constructed using a defined transformation of the values in GF(2^8)
- designed to be resistant to all known attacks

**Shift Rows**

- a circular byte shift in each each
  - 1\textsuperscript{st} row is unchanged
  - 2\textsuperscript{nd} row does 1 byte circular shift to left
  - 3\textsuperscript{rd} row does 2 byte circular shift to left
  - 4\textsuperscript{th} row does 3 byte circular shift to left
- decrypt does shifts to right
**Mix Columns**

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF($2^8$) using prime poly $m(x) = x^8 + x^4 + x^3 + x + 1$
Add Round Key
♦ XOR state with 128-bits of the round key
♦ again processed by column (though effectively a series of byte operations)
♦ inverse for decryption is identical since XOR is own inverse, just with correct round key
♦ designed to be as simple as possible

AES Key Expansion
♦ takes 128-bit (16-byte) key and expands into array of 44/52/60 (32-bit) words
♦ start by copying key into first 4 words
♦ then loop creating words that depend on values in previous & 4 positions back
  – in 3 of 4 cases just XOR these together
  – every 4th has S-box + rotate + XOR constant of previous before XOR together
♦ designed to resist known attacks
AES Decryption

♦ AES decryption: the order of steps done in reverse is different from that of in encryption
♦ but can define an equivalent inverse cipher with the same order of steps as encryption
  – but using inverses of each step
  – with a different key schedule
♦ works since result is unchanged when
  – swap byte substitution & shift rows
  – swap mix columns & add (tweaked) round key
Encryption/decryption Operations

(a) Encryption

(b) Decryption
Implementation Aspects

♦ can efficiently implement on 8-bit CPU
  – byte substitution works on bytes using a table of 256 entries
  – shift rows is simple byte shifting
  – add round key works on byte XORs
  – mix columns requires matrix multiply in GF(2^8) which works on byte values, can be simplified to use a table lookup

Implementation Aspects

♦ can efficiently implement on 32-bit CPU
  – redefine steps to use 32-bit words
  – can precompute 4 tables of 256-words
  – then each column in each round can be computed using 4 table lookups + 4 XORs
  – at a cost of 16Kb to store tables
♦ designers believe this very efficient implementation was a key factor in its selection as the AES cipher
Summary

♦ have considered:
  – the AES selection process
  – the details of Rijndael – the AES cipher
  – looked at the steps in each round
  – the key expansion
  – implementation aspects