Chapter 5
Advanced Encryption Standard
Origins

• clear a replacement for DES was needed
  – have theoretical attacks that can break it
  – have demonstrated exhaustive key search attacks
• can use Triple-DES – pretty safe
  – but slow, small blocks
• issued call for ciphers in `97
• 15 candidates accepted in Jun 98
• 5 were short-listed in Aug-99
• AES selected in Oct-2000
• issued as FIPS PUB 197 standard in Nov-2001
AES Requirements

- private 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 (15 to 5):
  – security – effort to practically cryptanalyse
  – cost – computational, high-speed applications
  – algorithm & implementation characteristics
    • Flexibility, simplicity, maintainability

• final criteria
  – general security
  – software & hardware implementation ease
  – implementation attacks
  – flexibility (in changing en/decrypt, keying, #rounds, 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
- All were thought to be good – came down to best balance of attributes to meet criteria.
- Note mix of commercial (MARS, RC6, Twofish) verses academic (Rijndael, Serpent) proposals
The AES Cipher

• 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
  – rather than feistel (operate on halves at a time)
• designed to be:
  – resistant against known attacks
  – speed and code compactness on many CPUs
  – design simplicity
AES

- processes data as 4 groups of 4 bytes (state)
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
- initial XOR key material & incomplete last round
- all operations can be combined into XOR and table lookups - hence very fast & efficient
Rijndael

(a) Encryption

(b) Decryption
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 row
  - 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
- since state is processed by columns, this step permutes bytes between the columns
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

\[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02 \\
\end{bmatrix}
\begin{bmatrix}
s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\
s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\
s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\
s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\
\end{bmatrix}
= 
\begin{bmatrix}
s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\
s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\
s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\
s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\
\end{bmatrix}
\]
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 Round
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 places back
  - in 3 of 4 cases just XOR these together
  - every 4\textsuperscript{th} has S-box + rotate + XOR constant of previous before XOR together
- designed to resist known attacks
AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for 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
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 pre-compute 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