CS 465

AES
Programming Lab #1

- Implement AES
- Use the FIPS 197 spec as your guide
  - Avoid looking at code on the Internet
  - Challenge yourself to implement the algorithm based on sources mentioned in the lab specification
  - The standard provides programming language independent pseudo-code
  - 20 pages at the end of the spec has complete, step-by-step debugging information to check your solution
AES Parameters

• **Nb** – Number of columns in the State
  - For AES, Nb = 4

• **Nk** – Number of 32-bit words in the Key
  - For AES, Nk = 4, 6, or 8

• **Nr** – Number of rounds (function of Nb and Nk)
  - For AES, Nr = 10, 12, or 14
AES methods

- Convert to state array

- Transformations (and their inverses)
  - AddRoundKey
  - SubBytes
  - ShiftRows
  - MixColumns

- Key Expansion
Inner Workings

• See Flash demo URL on course Lectures page
Finite Fields

• AES uses the finite field \( GF(2^8) \)
  o Polynomials of degree 8
  o \( b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 \)
    • \( \{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\} \)

• Byte notation for the element: \( x^6 + x^5 + x + 1 \)
  o \( 0x^7 + 1x^6 + 1x^5 + 0x^4 + 0x^3 + 0x^2 + 1x + 1 \)
  o \( \{01100011\} \) – binary
  o \( \{63\} \) – hex

• Has its own arithmetic operations
  o Addition
  o Multiplication
Finite Field Arithmetic

• Addition (XOR)
  o \((x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2\)
  o \(\{01010111\} \oplus \{10000011\} = \{11010100\}\)
  o \(\{57\} \oplus \{83\} = \{d4\}\)

• Multiplication is tricky
  o Study section 4.2 in the spec
  o In 4.2.1, a paragraph describes what your implementation will do. Study it. Difficult to interpret.
Finite Field Multiplication (\(\cdot\))

\[(x^6 + x^4 + x^2 + x + 1) (x^7 + x + 1) =
\]

\[x^{13} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1\]

\[= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\]

and

\[x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)\]

\[= x^7 + x^6 + 1.\]
There’s a better way
  o Patterned after the divide and conquer modular exponentiation algorithm (CS 312)
  o xtime() – very efficiently multiplies its input by \{02\}
    • This is the same as multiplying a polynomial by \(x\)
      o think about what is the binary representation of the polynomial \(x\)?
    • Figure out when the mod operation should occur.

Multiplication by higher powers can be accomplished through repeated applications of xtime()
Efficient Finite Field Multiply

Example: \{57\} \cdot \{13\}

\{57\} \cdot \{02\} = \text{xtime}({57}) = \{ae\}
\{57\} \cdot \{04\} = \text{xtime}({ae}) = \{47\}
\{57\} \cdot \{08\} = \text{xtime}({47}) = \{8e\}
\{57\} \cdot \{10\} = \text{xtime}({8e}) = \{07\}

\{57\} \cdot \{13\} = \{57\} \cdot (\{01\} \oplus \{02\} \oplus \{10\})
= \{57\} \cdot (\{01\} \oplus \{02\} \oplus \{10\})
= (\{57\} \cdot \{01\}) \oplus (\{57\} \cdot \{02\}) \oplus (\{57\} \cdot \{10\})
= \{57\} \oplus \{ae\} \oplus \{07\}
= \{fe\}
Efficient Finite Field Multiply

Example: \{57\} \cdot \{13\}

\[
\begin{align*}
\{57\} \cdot \{02\} &= \text{xtime}(\{57\}) = \{ae\} \\
\{57\} \cdot \{04\} &= \text{xtime}(\{ae\}) = \{47\} \\
\{57\} \cdot \{08\} &= \text{xtime}(\{47\}) = \{8e\} \\
\{57\} \cdot \{10\} &= \text{xtime}(\{8e\}) = \{07\}
\end{align*}
\]

\[
\begin{align*}
\{57\} \cdot \{13\} &= \{57\} \cdot (\{01\} \oplus \{02\} \oplus \{10\}) \\
&= \{57\} \cdot (\{01\} \oplus \{02\} \oplus \{10\}) \\
&= (\{57\} \cdot \{01\}) \oplus (\{57\} \cdot \{02\}) \oplus (\{57\} \cdot \{10\}) \\
&= \{57\} \oplus \{ae\} \oplus \{07\} \\
&= \{fe\}
\end{align*}
\]

{10} in hex is 16, not decimal 10!

These are hexadecimal numbers \{xx\}
See detailed multiplication example on the Lectures web page