The Advanced Encryption Standard (AES)

Lecture 13

Josh Brake
Harvey Mudd College

Outline

- Introduction to Finite (Galois) Fields
- AES Overview

Learning Objectives

By the end of this lecture you will…

- Have a basic understanding of cryptography
- Have an operational understanding of the fundamental mathematics used in AES
- Understand the basic process of AES

Why Is Encryption Important?

Imagine you want to send a message to your friend, but you don’t have a secure channel to do it.

Q: How can you send a message?

A: Use a code, or cipher, to encode and decode the information.

Eliminating the need for a secure channel

To be able to decode our encrypted messages, we need a secure channel to transmit the key. Otherwise, everyone will have access to the key and be able to decrypt our messages.

What if we don’t have a private channel to begin with?

Public Key Cryptography

Enter public key cryptography. The basic process is as follows:

1. Generate two keys: public and private with the property that messages encrypted with the private key can only be decrypted with the public key and vice versa.
2. Publicly share your public key along with your name.
3. Encrypt messages with your private key and send.
4. Recipient uses your public key to decrypt.

This confirms that messages are sent from you.

Can be made even more powerful when you encrypt with your private key **and** your recipient's public key. This confirms the sender's identity and ensures that only the recipient can decrypt the message.

Key Terms in Cryptography

- Symmetric vs. Asymmetric encryption
  - Symmetric: Same key used to encrypt and decrypt [ex. Rivest–Shamir–Adleman (RSA)]
  - Asymmetric: Different keys (key pair) used to encrypt and decrypt [ex. Advanced Encryption Standard (AES)]
- Plaintext: Unencrypted information
- Key: Code used to encrypt/decrypt the information
- Ciphertext: Encrypted information

Typical applications on the Internet use a combination of RSA and AES. Use RSA to securely exchange AES keys, then use AES to encrypt all the data that's sent. Maximizes advantages of each algorithm (speed of AES, public key encryption of RSA).

Finite Fields

A finite field is a set F with two operators (+ and $\cdot$) satisfying the following:

- **Closure**: if $a, b \in F$, then $a + b \in F$ and $a \cdot b \in F$.
- **Identity**: 0 and 1 elements exist such that $a + 0 = a$ and $a \cdot 1 = a$ for any $a \in F$.
- **Inverse**: For any a, there exists x and y such that $a + x = 0$ and $a \cdot y = 1$. Except there is no multiplicative inverse of 0.
- **Associative**, **commutative**, and **distributive** properties hold.
- The two most common fields are $GF(p)$ and $GF(2^n)$.

History

Galois Fields are named after French mathematician Évariste Galois. He died in a duel at age 20 after being expelled from college for political protest after the French revolution. He spent six months in prison developing his mathematical ideas.

$\mathrm{GF}(p)$: Arithmetic modulo p

Example: $\mathrm{GF}(13)$: contains 0–12, math is all done mod 13.

- 0 and 1 are the identity elements
- Additive inverse of 4 is 9 because $4 + 9 = 0$.
- Multiplicative inverse of 4 is 10 because $4 \cdot 10 \mod 13 = 1$.

$GF(2^n)$

- Elements are polynomials in some dummy variable x with coefficients of 0 or 1.
- Hence, each coefficient operates mod 2.
- Addition, subtraction, and exclusive or are all the same.
- This makes the hardware very easy because addition simplifies to bitwise XOR.
- Operations are done modulo some characteristic irreducible (prime) polynomial. For AES, this polynomial is $m(x) = x^8 + x^4 + x^3 + x + 1$.

Advantages of Galois Arithmetic for Digital Hardware

Example: $\mathrm{GF}(2^8)$ with characteristic polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.

- Each element is an 8-bit number (easy hardware)
- Addition is 8-bit XOR
- Multiplication is shifting

$GF(2^8)$ Hexadecimal Representation

- Express $a = x^5 + x^3 + x^2 + 1$ in hexadecimal.

$a = 0b00101101 = 0x2D$

- Express $b = x^7 + x^6 + x^2 + x + 1$ in hexadecimal

$b = 0b11000111 = 0xC7$

$GF(2^8)$ Addition

Compute $c = a + b$ where $a = x^5 + x^3 + x^2 + 1$ and $b = x^7 + x^6 + x^2 + x + 1$.

$a = 0b00101101 = 0x2D$

$b = 0b11000111 = 0xC7$

$$\begin{array}{r} 0b00101101 \\ \oplus 0b11000111 \\ \hline 0b11101010 \end{array}$$

$\mathrm{GF}(2^8)$ Addition

Given $a = x^5 + x^3 + x^2 + 1$ and $b = x^7 + x^6 + x^2 + x + 1$.

What is the additive identity of a (i.e., what is the zero element)?

$a =$ `0b00101101` $=$ `0x2D`

0

What is the additive inverse of a (i.e., $-a$)?

$a =$ `0b00101101` $=$ `0x2D`

a

$GF(2^8)$ Multiplication

What is $a(x) \cdot x$ where $a = x^5 + x^3 + x^2 + 1$?

$$a = x^5 + x^3 + x^2 + 1 = 0b00101101$$
$$a \cdot x = x^6 + x^4 + x^3 + x \rightarrow 0b01011010$$

$\mathrm{GF}(2^8)$ Multiplication

What is $b(x) \cdot x$ where $b = x^7 + x^6 + x^2 + x + 1$?

$$b = x^7 + x^6 + x^2 + x + 1 = 0b11000111$$

$$b \cdot x = x^8 + x^7 + x^3 + x^2 + x \rightarrow 0b110001110$$

$GF(2^8)$ Multiplication

$a = x^5 + x^3 + x^2 + 1$

What is the multiplicative **identity** of $a(x)$ (i.e., our 1 element?

$$a \cdot x = a \rightarrow x = 1$$

What is the multiplicative **inverse** of $a(x)$?

Tricky to find! Need to use Extended Euclidean Algorithm.

Galois Field Math Summary

- Addition is XOR.
- Multiplication by x is left shift.
  - If the coefficient of x^8 is 0, then you are done.
  - Otherwise, you need to reduce modulo an irreducible polynomial. For AES, this polynomial is $m(x) = x^8 + x^4 + x^3 + x + 1$.

Galois Fields Worksheet

Consider a Galois field GF(2^8) with characteristic polynomial $m(x) = x^8 + x^4 + ^3 + x + 1$

1. Express the characteristic polynomial as a hexadecimal number.

`0x{01}1B`.

2. Express $c = x^7 + x^6 + x^3 + x^2 + x + 1$ in hexadecimal.

`0b11001111=0xCF`

3. Let $d = x^7 + x^3 + x$. Compute $e = c + d$.

$d = 0b10001010$

$$\begin{array}{r} 0b11001111 \\ \oplus 0b10001010 \\ \hline 0b01000101 \end{array}$$

4. Compute $-d$.

$d = -d = 0b10001010$

5. Compute $c \cdot x$.

$\{01\}9E \oplus \{01\}1B$

$$\begin{array}{r} 0b1\ 1001\ 1110 \\ \oplus 0b1\ 0001\ 1011 \\ \hline 0b0\ 1000\ 0101 \end{array}$$

`0x85`

6. Compute $c \cdot x^2$.

$= (c \cdot x) \cdot x$

Can either approach by iteratively xoring with the $m(x)$ or doing long division.

$c \cdot x^2 = x^4 + 1 = 0x11$

The Advanced Encryption Standard

AES Overview

From the spec:

> The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext.

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

128-bit AES

- The cipher works by taking a plain text message and scrambling it into a cyphertext message in a way that is hard to reverse.
- Scrambling is done 10 times to make it hard. The key changes from round to round.

AES Data Organization

128-bit message is organized as a matrix of 16 bytes (4x4).

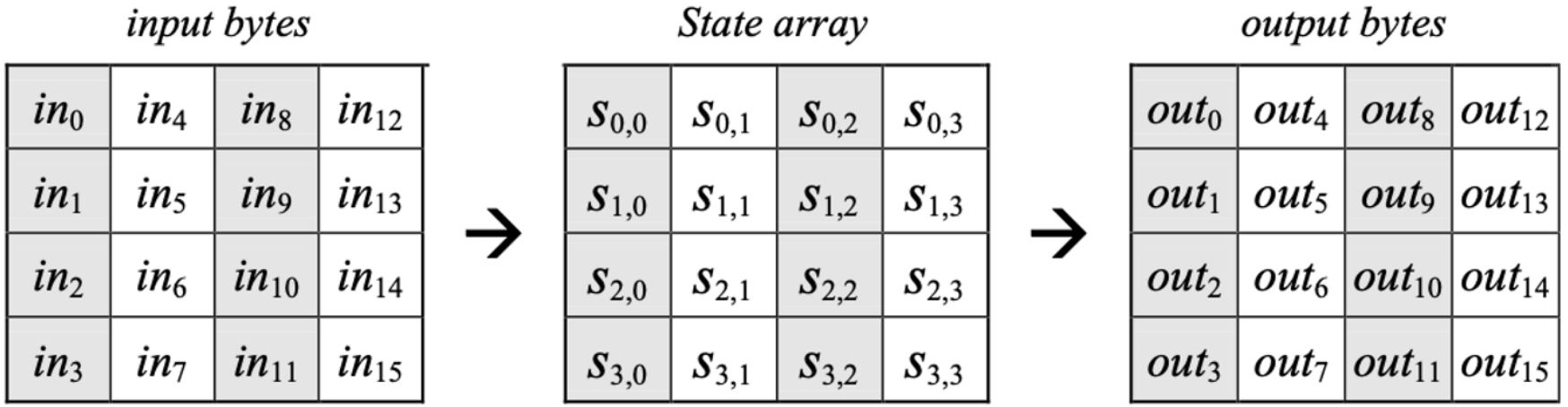

Figure 3. State array input and output.

AES Cipher Process

Each step of the cipher involves the following steps

1. `SubBytes`: take each byte and replace it with a different byte using a randomish lookup table.
2. `ShiftRows`: move bytes around in the rows
3. `MixColumns`: funky Galois multiplication on elements of the columns
4. `AddRoundKey`: XOR with the key for the current round.

AES Cipher Pseudocode

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Figure 5. Pseudo Code for the Cipher.[1]