DTTF/NB479: Dszquphsbqiz Day 17

- Announcements:
  - DES due Thursday.
  - Careful with putting it off since Ch 3 test Friday too.
- Today:
  - Finish GF(2^8)
  - Rijndael
- Questions?

AES (Rijndael)

- The S-boxes, round keys, and MixColumn functions require the use of GF(2^8), so

Fields (T&W, 3.11)

- A *field* is a **set of numbers** with the following properties:
  - Addition, with identity: a + 0 = a and inverse a+(-a)=0
  - Multiplication with identity: a*1=a, and inverse (a * a^{-1} = 1 for all a != 0)
  - Subtraction and division (using inverses)
  - Commutative, associative, and distributive properties
  - Closure over all four operations
- Examples:
  - Real numbers
  - GF(4) = {0, 1, ω, ω^2} with these additional laws: x + x = 0 for all x and $\omega + 1 = \omega^2$.
  - GF(p^n) for prime p is called a Galois Field.

A Galois field is a finite field with p^n elements for a prime p

- There is **only one** finite field with p^n elements for every power of n and prime p.
- $GF(p^n) = Z_p[X] \pmod{P(X)}$ is a field with p^n elements.
- Wasn't $Z^2[X] \pmod{X^2 + X + 1} = GF(4)$?
- Consider $GF(2^n)$ with $P(X) = X^8 + X^4 + X^3 + X + 1$

  Rijndael uses this!

Finish quiz.

Back to Rijndael/AES

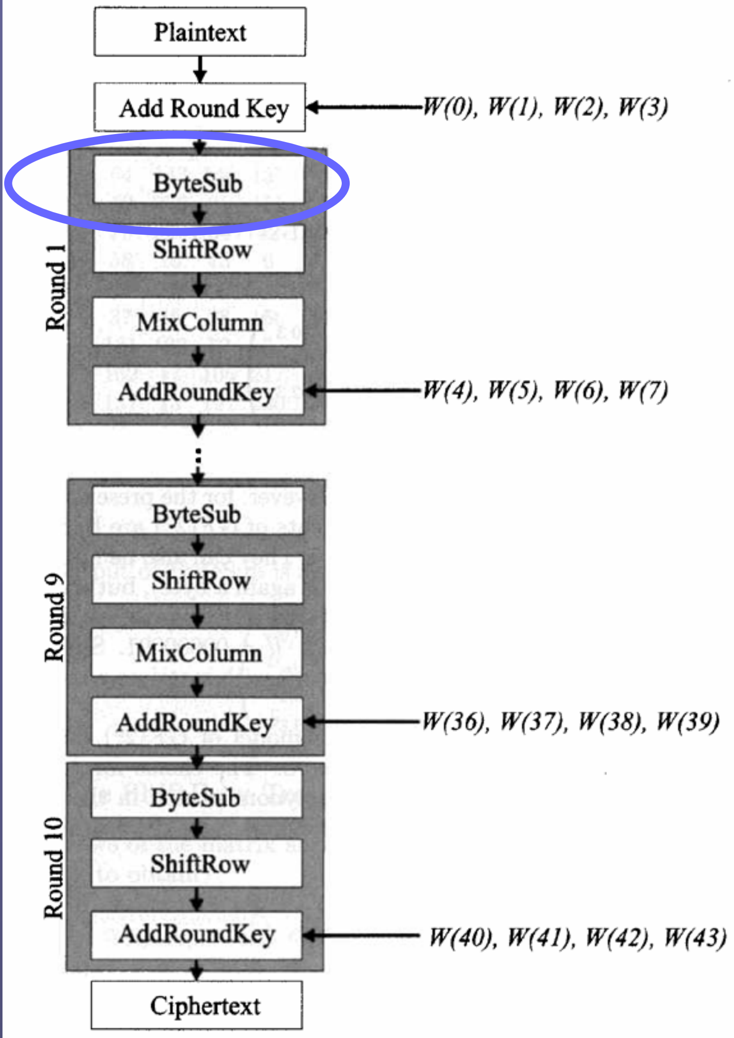

Figure 5.1: The AES-Rijndael Algorithm

- Parallels with DES?
  - Multiple rounds
    - (7 is enough to require brute force)
  - Diffusion
  - XOR with round keys
  - No MixColumn in last round
- Major differences
  - Not a Feistel system
  - Much quicker diffusion of bits (2 rounds)
  - Much stronger against linear, diffy. crypt., interpolation attacks

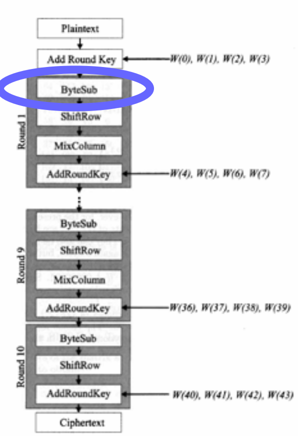

Figure 5.1: The AES-Rijndael Algorithm

ByteSub (BS)

S-Box

99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Table 5.1: S-Box for Rijndael

1. Write 128-bit input *a* as matrix with 16 byte entries (column major ordering):

$$a = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

2. For each byte, abcdefgh, replace with byte in location (abcd, efgh)

   Example: 00011111 → ___

   Example: 11001011 → ___

3. Output is a matrix called b

Why were these numbers chosen?

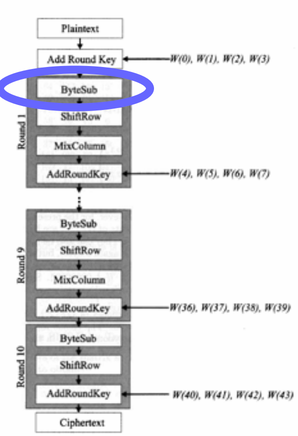

Figure 5.1: The AES-Rijndael Algorithm

S-box Derivation

The S-box maps byte x to byte z via the function $z = Ax^{-1}+b$:

Input byte *x*: $x_7x_6x_5x_4x_3x_2x_1x_0$

Compute the inverse in GF(2^8): $y_7y_6y_5y_4y_3y_2y_1y_0$ *(non-linear, vs. attacks)*
(use 0 as inverse of 0)

Compute this linear function *z in GF(2^8)*: (to complicate attacks)
(A is simple to implement)
b chosen so

$$z \neq x \text{ and } z \neq \overline{x}$$

$$\begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{pmatrix}$$

ShiftRow (SR)

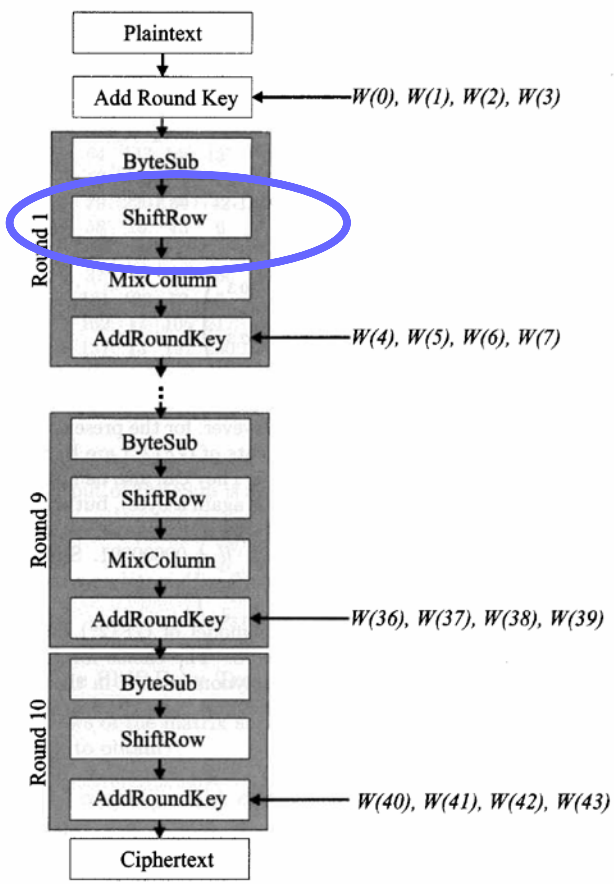

Figure 5.1: The AES-Rijndael Algorithm

Shifts the entries of each row by increasing offset:

$$c = \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{pmatrix}$$

Gives resistance to newer attacks (truncated differentials, Square attack)

MixColumn (MC)

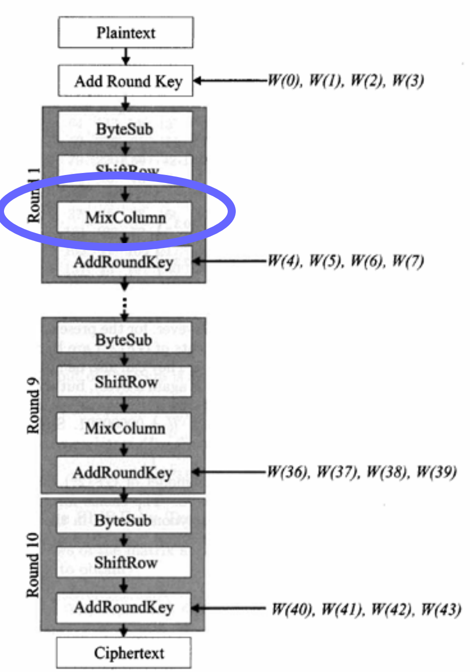

Figure 5.1: The AES-Rijndael Algorithm

Multiply – via GF(2^8) – with the fixed matrix shown.

$$d = \begin{pmatrix} 00000010 & 0..011 & 0..01 & 0..01 \\ 00000001 & 0..010 & 0..011 & 0..01 \\ 00000001 & 0..01 & 0..010 & 0..011 \\ 00000011 & 0..01 & 0..01 & 0..010 \end{pmatrix} \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix}$$

Speed?

64 multiplications, each involving at most 2 shifts + XORs

Gives quick diffusion of bits

AddRoundKey (*ARK*)

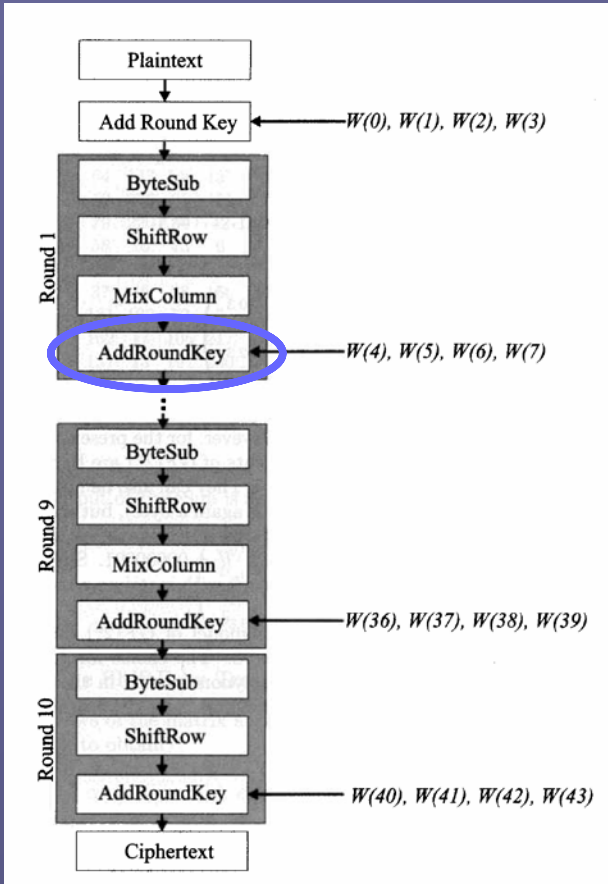

Figure 5.1: The AES-Rijndael Algorithm

XOR the round key with matrix d.

$$e = d \oplus k_i$$

Key schedule on next slide

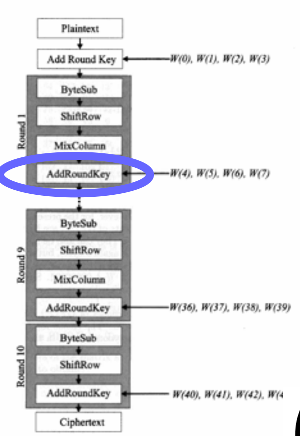

Figure 5.1: The AES-Rijndael Algorithm

Key Schedule

Write original key as 4x4matrix with 4 columns: W(0), W(1), W(2), W(3).
Key for round i is (W(4i), W(4i+1), W(4i+2), W(4i+3))

$$\underbrace{(W(0)\quad W(1)\quad W(2)\quad W(3)}_{K_0}\quad \underbrace{W(4)\quad \ldots W(7)}_{K_1}\quad \ldots\quad \underbrace{W(43))}_{K_{10}}$$

Other columns defined recursively:

$$W(i) = W(i-4) \oplus \begin{cases} T(W(i-1)) & if\ 4\,|\,i \\ W(i-1) & otherwise \end{cases}$$

$$W(i) = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \xrightarrow{Shift\ and\ Sbox} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix} \oplus \begin{pmatrix} r(i) \\ 0 \\ 0 \\ 0 \end{pmatrix} = T(W(i))$$

$$r(i) = (00000010)^{(i-4)/4}\ in\ GF(2^8)$$

Highly non-linear. Resists attacks at finding whole key when part is known

192-, 256-bit versions similar

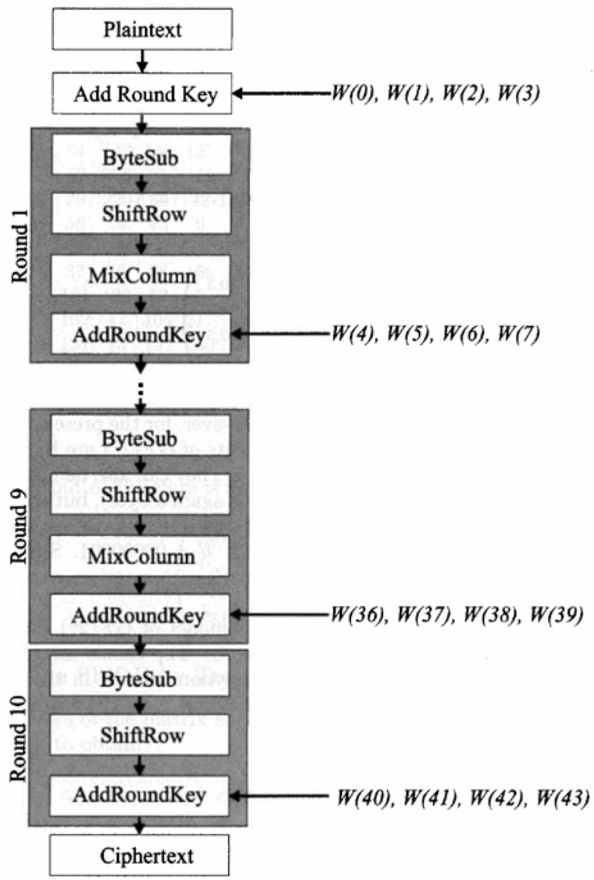

Figure 5.1: The AES-Rijndael Algorithm

Decryption

E(k) is:

(ARK_0, BS, SR, MC, ARK_1, … BS, SR, MC, ARK_9, BS, SR, ARK_{10})

Each function is invertible:

ARK; IBS; ISR; IMC

$$\begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix}$$

So D(k) is:

ARK_{10}, ISR, IBS, ARK_9, IMC, ISR, IBS, … ARK_1, IMC, ISR, IBS, ARK_0)

Half-round structure:

- Write E(k) = ARK, (BS, SR), (MC, ARK), … (BS, SR), (MC, ARK), (BS, SR), ARK
  (Note that last MC wouldn't fit)
- D(k) = ARK, (ISR, IBS), (ARK, IMC), (ISR, IBS), … (ARK, IMC), (ISR, IBS), ARK

Can write:

D(k) = ARK, (IBS, ISR), (IMC, IARK), … (IBS, ISR), (IMC, IARK), (IBS, ISR), ARK

Wrap-up

- Wikipedia's entry has some nice visuals
- But this site has even nicer animations*

* Thanks to Adam Shiemke, 2009 for the link