Lecture 6:
Two Fish on the Rijndael

The algorithm might look haphazard, but we did everything for a reason. Nothing is in Twofish by chance. Anything in the algorithm that we couldn't justify, we removed. The result is a lean, mean algorithm that is strong and conceptually simple.

Bruce Schneier

CS588: Security and Privacy
University of Virginia
Computer Science

David Evans
http://www.cs.virginia.edu/~evans

Menu

- Clipper
- AES Program
- RC6
- Blowfish
- AES Winner - Rijndael

Breaking Grades File

- Not in my office or any UVA computer
  - **Do not try to break into any UVA computer**
- Home PC: C:\cs588\grades.txt (encrypted)
  - If you obtain that file, it tells you what to do next
- Adelphia Cable Modem
- My browser is set to disallow ActiveX, allow Java and JavaScript

Clipper

- 1993 – AT&T markets secure telephony device
- Law enforcement: US courts can authorize wire taps, must be able to decrypt
- NSA proposes Clipper Chip
  - Secret algorithm (Skipjack), only implemented in hardware

Key Escrow

- NSA has copy of special key, can get with a court order
- Sender transmits $E(M, k) \parallel LEAF$ ("law enforcement agents' field")
- Holder of special key can decrypt LEAF to find message key and decrypt message

LEAF

Known by FBI

$LEAF = E((E(k, u) \parallel n \parallel a), f)$

k = message key

u = 80-bit special key (unique to chip)

n = 30-bit identifier (unique to chip)

a = escrow authenticator

f = 80-bit key (same on all chips)

Wire Tap

- FBI investigating Alice, intercepts Clipper communication
- Uses f to decrypt LEAF:
  $\mathrm{D}\,(\mathrm{E}\,((\mathrm{E}\,(k, u) \,\|\, n \,\|\, a), f)) = \mathrm{E}\,(k, u) \,\|\, n \,\|\, a$
- Delivers n and court order to 2 escrow agencies, obtains u
- Decrypts $\mathrm{E}\,(k, u)$ to obtain message key and decrypt message

Two Escrow Agencies

- Proposal didn't specify who (one probably NSA)
- Divide u so neither one can decrypt messages on their own (even if they obtain f)

One gets $u \oplus \mathrm{X}$, other gets X

Clipper Security

- How do you prevent criminals from transmitting wrong LEAF?
  - NSA solution: put it in hardware, inspect all Clipper devices
    - Still vulnerable to out-of-the box device

Clipper Politics

- Not widely adopted, administration backed down
  - Secret algorithm
  - Public relations disaster
    - Didn't involve academic cryptographers early
    - Proposal was rushed, in particular hadn't figured out who would be escrow agencies
- See http://www.eff.org/pub/Privacy/Key_escrow/Clipper/
- Future?: Senators have called for new Clipper-like restrictions on cryptography
- Lessons learned well for AES process

AES

- 1996: NIST initiates program to choose Advanced Encryption Standard to replace DES
- Requests algorithm submissions: 15
- Requirements:
  - Secure for next 50-100 years
  - Performance: faster than 3DES
  - Support 128, 192 and 256 bit keys
    - Brute force search of 2^{128} keys at 1 Trillion keys/second would take 10^{19} years (10^{9} * age of universe)
  - Must be a block cipher

AES Process

- Open Design
  - DES: design criteria for S-boxes kept secret
- Many good choices
  - DES: only one acceptable algorithm
- Public cryptanalysis efforts before choice
  - Heavy involvements of academic community, leading public cryptographers
- Conservative (but quick): 4 year+ process

AES Round 1

- 15 submissions accepted
- Weak ciphers quickly eliminated
  - Magenta broken at conference!
- 5 finalists selected: MARS (IBM), RC6 (Rivest, et. al.), Rijndael (top Belgium cryptographers), Serpent (Anderson, Biham, Knudsen), Twofish (Schneier, et. al.)
  - Security v. Performance is main tradeoff
    - How do you measure security?
  - Simplicity v. Complexity
    - Need complexity for confusion
    - Need simplicity to be able to analyze and implement efficiently

Breaking a Cipher

- Real World Standard
  - Attacker can decrypt secret messages
  - Reasonable amount of work, actual amount of ciphertext
- "Academic" Standard
  - Attacker can determine something about the message
  - Given unlimited number of chosen plaintext - ciphertext pairs
  - Can perform a very large number of computations, up to, but not including, 2^n, where n is the key size in bits (i.e. assume that the attacker can't mount a brute force attack, but can get close)

AES Evaluation Criteria

1. Security

   Most important, but hardest to measure

   Resistance to cryptanalysis, randomness of output
2. Cost and Implementation Characteristics

   Licensing, Computational, Memory

   Flexibility (different key/block sizes), hardware implementation

From RC5 to RC6 in seven easy steps

From Rivest's RC6 talk, http://www.rsasecurity.com/rsalabs/aes/

Description of RC6

- RC6-w/r/b parameters:
  - *Word size* in bits: w (32) (lg(w) = 5)
  - Number of *rounds*: r (20)
  - Number of *key bytes*: b (16, 24, or 32)
- Key Expansion:
  - Produces array S[0, … 2r + 3] of w-bit *round keys.*
- Encryption and Decryption:
  - Input/Output in 32-bit registers A,B,C,D

Design Philosophy

- Leverage experience with RC5: use *data-dependent rotations* to achieve a high level of security.
- Adapt RC5 to meet AES requirements
- Take advantage of a new primitive for increased security and efficiency: *32x32 multiplication*, which executes quickly on modern processors, to compute rotation amounts.

Data-Dependent Rotations

a	b	c	d	e	f	g	h	$\ll 3$
d	e	f	g	h	a	b	c	

$X \oplus X' = \Delta X$

$X_1 = X \ll f(X, k)$ $X_1' = X' \ll f(X', k)$

Can we say anything about $\Delta X_1 = X_1 \oplus X_1'$?

Same number of bits are still different, but can't tell which ones.

$\lll n$ means rotate left by amount in low order $\log_2 w$ bits of n (word size $w = 32$, 5 bits)

(1) Start with RC5

RC5 encryption inner loop:

```
for i = 1 to r do
    A = ((A ⊕ B) <<< B) + S [i]
    (A, B) = (B, A)
```

<<< only depends on 5 bits of B

Can RC5 be strengthened by having rotation amounts depend on *all* the bits of B?

Better rotation amounts?

- *Modulo* function?
  Use low-order bits of (B mod d)
  Too slow!
- *Linear* function?
  Use high-order bits of $(c \times B)$
  Hard to pick c well
- *Quadratic* function?
  Use high-order bits of $(B \times (2B+1))$

Properties $B \times (2B+1)$ should have:

1. One-to-one (can invert for decryption)
2. Good distribution – if B is well distributed, so is $B \times (2B + 1)$
3. High order bits depend on all bits of B (diffusion)
4. Easy to calculate efficiently (if your hardware has 32-bit multiplies)

$B \times (2B+1)$ is *one-to-one* mod 2^w

Proof: By contradiction: Assume $B \neq C$
and $B \times (2B + 1) = C \times (2C + 1) \bmod 2^w$
then

$B \times (2B + 1) - C \times (2C + 1) = 0 \bmod 2^w$

$2B^2 + B - (2C^2 + C) = 0 \bmod 2^w$

$(B - C) \times (2B+2C+1) = 0 \bmod 2^w$

But (B-C) is nonzero and (2B+2C+1) is odd; their product can't be zero! □

Corollary:
B uniform → $B \times (2B+1)$ uniform
(and high-order bits are uniform too!)

3. High-order bits of $B \times (2B+1)$ depend on all bits of B (diffusion)

$B = B_{31}B_{30}B_{29} \ldots B_1B_0$ in binary,

$\times$ ~~T = 2B+1 = $B_{30}B_{29}B_{28} \ldots B_0 1$~~

$$
\begin{array}{r}
B_{31}B_{30}B_{29} \ldots B_1B_0 \\
B_0 * B_{31}B_{30}B_{29} \ldots B_1B_0 \quad \\
B_1 * B_{31}B_{30}B_{29} \ldots B_1B_0 \qquad \\
\hline
+ \qquad \ldots \qquad\qquad
\end{array}
$$

$f(B) = F_{31}F_{30}F_{29} \ldots F_1F_0$

$F_i = (1 \times B_i) + \sum_{j = 0..i-1} (B_j \times B_{i-j-1}) + C_{i-1} \bmod 2$

Diffusion, cont.

$F_i = B_i + \sum_{j=0..i-1}(B_j \times B_{i-j-1}) + C_{i-1} \bmod 2$

$C_i = B_i + \sum_{j=0..i-1}(B_j \times B_{i-j-1}) + C_{i-1} \text{ div } 2$

- Flipping bit B_i
  - Leaves bits $F_0 \ldots F_{i-1}$ of f(B) unchanged,
  - Flips bit F_i always
  - Flips bit F_j for $j > i$, with probability approximately ½
    - Different for different j's, but F_j depends on B_i for all $i > j$.
  - Is likely to change some high-order bits

(2) Quadratic Rotation Amounts

for i = 1 **to** r **do**
  t = (B × (2B + 1)) <<< 5
  A = ((A ⊕ B) <<< t) + S[i]
  (A, B) = (B, A)

But now much of the output of multiplication is being wasted (only 5 top bits used)...

(3) Use t, not B, as xor input

for i = 1 **to** r **do**
  t = (B × (2B + 1)) <<< 5
  A = ((A ⊕ t) <<< t) + S[i]
  (A, B) = (B, A)

RC5 used 64 bit blocks
AES requires 128-bit blocks

Double size of A and B?

  64-bit registers and operations are poorly supported by typical compilers and hardware

(4) Do two RC5's in parallel

$M = A_0B_0A_1B_1A_2B_2A_3B_3\ldots$
$M = A_0B_0C_0D_0A_1B_1C_1D_1\ldots$

for i = 1 **to** r **do**
  t = (B × (2B + 1)) <<< 5
  A = ((A ⊕ t) <<< t) + S[2i]
  ~~(A, B) = (B, A)~~
  u = (D × (2D + 1)) <<< 5
  C = ((C ⊕ u) <<< u) + S[2i + 1]
  ~~(C, D) = (D, C)~~

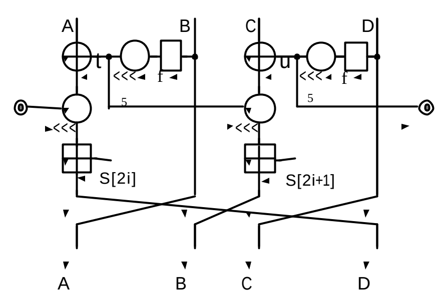
Same thing for next 64 bits

(5) Mix up data between copies

Switch rotation amounts between copies, and cyclically permute registers instead of swapping:

for i = 1 **to** r **do**
  t = (B x (2B + 1)) <<< 5
  u = (D x (2D + 1)) <<< 5
  A = ((A ⊕ t) <<< u) + S[2i]
  C = ((C ⊕ u) <<< t) + S[2i + 1]
  (A, B, C, D) = (B, C, D, A)

One Round of RC6



Key Expansion (Same as RC5's)

- Input: array L [0 … c-1] of input key words
- Output: array S [0 … 43] of round key words
- Procedure:

S [0] = 0xB7E15163 $= \text{Odd}[(e-2)2^{32}]$
for i = 1 **to** 43 **do** S[i] = S[i-1] + 0x9E3779B9 $= \text{Odd}[(\Phi-1)2^{32}]$
A = B = i = j = 0
for s = 1 **to** 132 **do**
  A = S[i] = (S[i] + A + B) <<< 3
  B = L[j] = (L[j] + A + B) <<< (A + B)
  i = (i + 1) mod 44
  j = (j + 1) mod c

What do $\pi/e/\Phi$ have to do with cryptography?

- Used by RC5, RC6, Blowfish, etc. in magic constants
- Mathematical constants have good pseudorandom distribution
- Since they are public and well-known, no fear that choice is a trap door

(6) Add Pre- and Post-Whitening

B = B + S[0]
D = D + S[1]
for i = 1 **to** r **do**
  t = (B x (2B + 1)) <<< 5
  u = (D x (2D + 1)) <<< 5
  A = ((A ⊕ t) <<< u) + S[2i]
  C = ((C ⊕ u) <<< t) + S[2i + 1]
  (A, B, C, D) = (B, C, D, A)
A = A + S[2r + 2]
C = C + S[2r + 3]

(7) Set r = 20 for high security

(based on analysis)

B = B + S[0]
D = D + S[1]
for i = 1 **to** 20 **do**
  t = (B x (2B + 1)) <<< 5
  u = (D x (2D + 1)) <<< 5
  A = ((A ⊕ t) <<< u) + S[2i]
  C = ((C ⊕ u) <<< t) + S[2i + 1]
  (A, B, C, D) = (B, C, D, A)
A = A + S[42]
C = C + S[43]

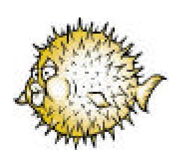
Final RC6

RC6 Decryption (for AES)

C = C – S [43]
A = A – S [42]
for i = 20 **downto** 1 **do**
  (A, B, C, D) = (D, A, B, C)
  u = (D x (2D + 1)) <<< 5
  t = (B x (2B + 1)) <<< 5
  C = ((C – S [2i + 1]) >>> t) ⊕ u
  A = ((A – S [2i]) >>> u) ⊕ t
D = D - S[1]
B = B - S[0]

Blowfish

- [Schneier93]
- 64-bit block cipher
- Much faster than DES
- Variable key length: 32-448 bits



- Many attempted crytanalyses, none successful yet
- Widely used: ssh, OpenBSD, PGPFone

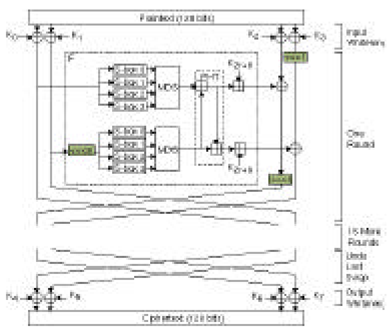
Key-Dependent S-Boxes

- Differential Cryptanalysis depends on analyzing S-box input/output different probabilities
- Change the S-boxes so you can’t do analysis

Blowfish → Twofish

- Blowfish: runs encryption 521 times to produce S-boxes
  - Too slow for AES, requires too much memory for smart cards
- Twofish
  - Provides options for how many key-dependant S-boxes (tradeoff security/time-space)
  - Also: increase block size (128 required by AES), change key schedule, etc.

Two Fish



From http://www.ddj.com/articles/1998/9812/9812b/9812bf1.htm

Choosing AES

(Table from Twofish Paper)

Cipher	Speed (32)	Speed (8)	Safety Factor	Simplicity (code size)
Serpent	62	69	3.56	341 KB
MARS	23	34	1.90	85 KB
RC6	15	43	1.18	48 KB
Rijndael	18	20	1.11	98 KB
Twofish	16	18	2.67	104 KB

(cycles/byte encrypt)

AES Winner: Rijndael

Invented by Joan Daemen and Vincent Rijmen

Rijndael. A variant of Square, the chief drawback to this cipher is the difficulty Americans have pronouncing it.

Bruce Schneier

Selected as AES, October 2000

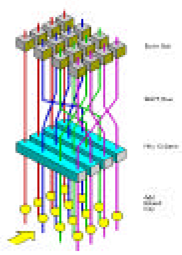
Rijndael Overview

- Key sizes: 128, 192, 256 bits
- Block sizes: 128, 192, 256 bits
- 10 rounds (including initial AddKey)
  - Academic break on 9 rounds, 256-bit key gives safety factor of 10/9 = 1.11
    - Requires 2^{224} work and 2^{85} chosen *related-key* plaintexts (why is this considered a break for 256-bit key but not 128-bit key?)

“Our results have no practical significance for anyone using the full Rijndael.”

Rijndael Round

1. Byte substitution using non-linear S-Box (independently on each byte)
2. Shift rows (square)
3. Mix columns – matrix multiplication by polynomial
4. XOR with round key



Rijndael Design

- Resistant to linear and differential cryptanalysis
- Differential trail
  - Probability that a given difference a' pattern at input produces an output difference of b'
  - Choose S-box and multiplication polynomial to minimize maximum difference probability

Charge

- Designing and picking a Cipher that will last 50 years is hard
  - Advances in computing power
  - Advances in cryptanalysis
  - Performance/security tradeoff keeps changing – need something that works today and in 2050
- This week: talk or email me about your project ideas
- Next time:
  - Key Distribution