CPSC 467: Cryptography and Security

Michael J. Fischer

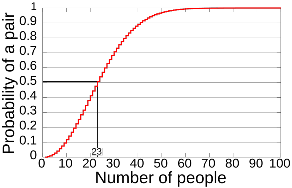
Lecture 9
September 29, 2020

Birthday Attack

Birthday Problem

The *birthday problem* is to find the probability that two people in a set of randomly chosen people have the same birthday.

This probability is greater than 50% in any set of at least 23 randomly chosen people.



From Wikipedia, "Birthday problem".

23 is far less than the 253 people that are needed for the probability to exceed 50% that at least one of them was born on a specific day, say January 1.[1]

[1] $1 - (364/365)^{252} \approx 0.49910 < 0.5$, but $1 - (364/365)^{253} \approx 0.50048 > 0.5$.

Birthday attack on a cryptosystem

A *birthday attack* is a known plaintext attack on a cryptosystem that reduces the number of keys that must be tried to roughly the square root of what a brute force attack needs.

If for example the original key length was 56 (as is the case with DES), then only about $\sqrt{2^{56}} = 2^{28}$ keys need to be tried.

Any cryptosystem with the group property is subject to a birthday attack.

How a birthday attack works

Assume (m, c) is a known plaintext-ciphertext pair, so $E_{k_0}(m) = c$ for Alice's secret key k_0.

- Choose 2^{28} random keys k_1 and encrypt m using each.
- Choose another 2^{28} random keys k_2 and decrypt c using each.
- Look for a common element u in these two sets.
- Suppose one is found for k_1 and k_2, so $E_{k_1}(m) = u = D_{k_2}(c)$. It follows that $E_{k_2}(E_{k_1}(m)) = c$, so we have succeeded in finding a key pair (k_2, k_1) that works for the pair (m, c).

By the group property, there is a key k such that $E_k = E_{k_2} \circ E_{k_1}$, so $E_k(m) = c$.

How a birthday attack works (cont.)

Alice's key k_0 also has $E_{k_0}(m) = c$. If it happens that $E_k = E_{k_0}$, then we have broken the cryptosystem.

We do not need to find k itself since we can compute E_k from E_{k_1} and E_{k_2} and D_k from D_{k_1} and D_{k_2}.

There are unlikely to be many distinct keys k such that $E_k(m) = c$, so with significant probability we have cracked the system. (For Caesar, there is only one such k.)

Using additional plaintext-ciphertext pairs, we can increase our confidence that we have found the correct key pair. Repeat this process if we have not yet succeeded.

I've glossed over many assumptions and details, but that's the basic idea.

Weakness of the birthday attack

The drawback to the birthday attack (from the attacker's perspective) is that it requires a lot of storage in order to find a matching element.

Fortunately, DES is not a group. If it were, this attack could be carried out in about a gigabyte of storage, easily within the storage capacity of modern workstations.

Advanced Encryption Standard

New Standard

Rijndael was the winner of NIST's competition for a new symmetric key block cipher to replace DES. An open call for algorithms was made in 1997 and in 2001 NIST announced that AES was approved as FIPS PUB 197.

Minimum requirements:

- Block size of 128-bits
- Key sizes of 128-, 192-, and 256-bits
- Strength at the level of triple DES
- Better performance than triple DES
- Available royalty-free worldwide

Five AES finalists: MARS, RC6, Rijndael, Serpent, and Twofish.

Details

Rijndael was developed by two Belgian cryptographers Vincent Rijmen and Joan Daemen.

Rijndael is pronounced like *Reign Dahl*, *Rain Doll* or *Rhine Dahl*.

Name confusion

- AES is the name of the standard.
- Rijndael is the name of the cipher.
- AES is a restricted version of Rijndael which was designed to handle additional block sizes and key lengths.

More details

AES was a replacement for DES.

- Like DES, AES is an iterated block cipher.
- Unlike DES, AES is not a Feistel cipher.
- Unlike DES, AES can be parameterized.

AES supports key lengths of 128-, 192- and 256-bits.

The algorithm consists of 10 to 14 rounds.

- Number of rounds depends on the key length.
- 10 rounds for 128-bit key, 12 for 192, 14 for 256.

How does AES actually work?

We present an overview of the structure of AES. For more details, as well as a much more extensive overview of its resistance to known attacks, see Wikipedia: Advanced Encryption Standard.

3 Big Ideas:

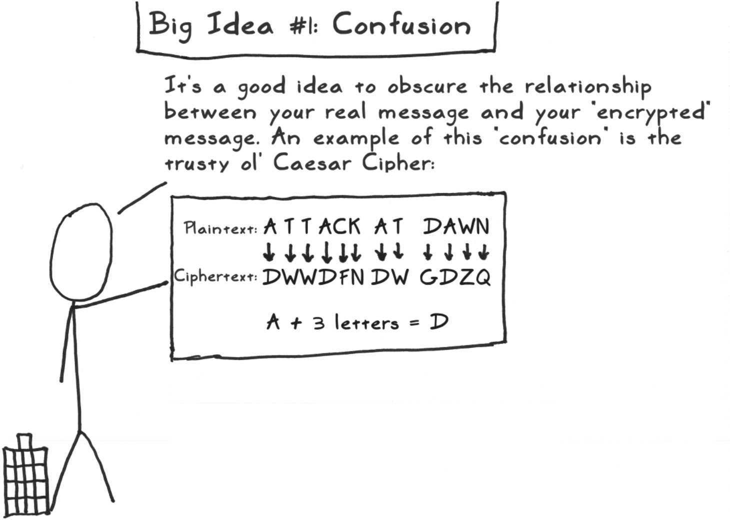
- Big Idea #1: Confusion
- Big Idea #2: Diffusion
- Big Idea #3: Key secrecy

Confusion & Diffusion

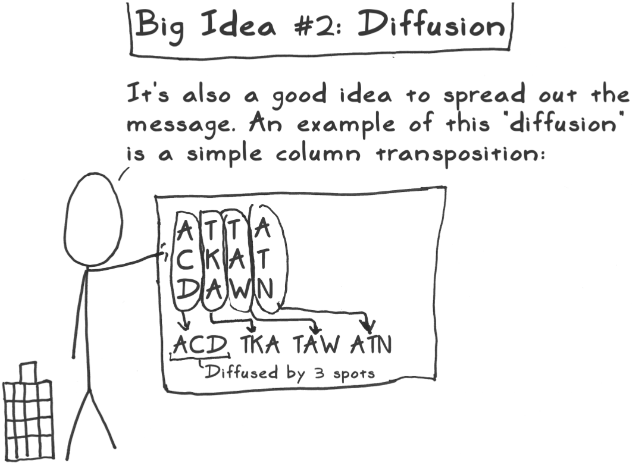
Confusion and *diffusion* are two properties of the operation of a secure cipher which were identified by Claude Shannon in his paper *Communication Theory of Secrecy Systems*[2].

DES, AES and many block ciphers are designed using Shannon's idea of confusion and diffusion.

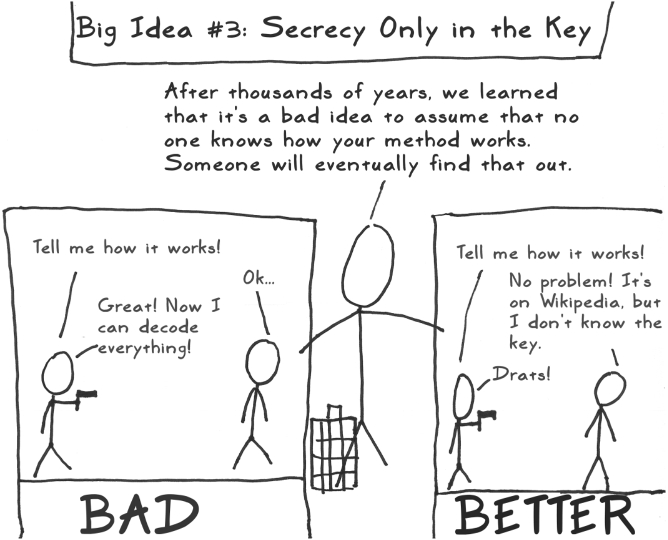
[2] `http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf`



www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Avalanche Effect and Evaluation Criteria[3]

Strict avalanche criterion (SAC) states that when a single input bit i is inverted, each output bit j changes with probability $\frac{1}{2}$, for all i and j.

Bit independence criterion (BIC) states that output bits j and k should change independently when any single input bit i is inverted, for all i, j and k.

[3] Wikipedia: Avalanche Effect

Transformations

Each AES round consists of 4 transformations:

- SubBytes(State)
- ShiftRows(State)
- MixColumns(State)
- AddRoundKey(State, Key)

Each round works on a state array.

A round key is derived from the primary key using a key schedule algorithm.

All four transformations are invertible. *Q: Is it a good thing?*

Roles of the four transformations

SubBytes() replaces bytes using a fixed S-box to achieve non-linearity.

ShiftRow() and *MixColumns()* are intended to mix up bits to achieve a wider distribution of plaintext in the whole message space.

AddRoundKey() provides the necessary secret randomness.

Q: How do these transformations relate to the Big Ideas?

Roles of the four transformations

Big Idea #1 *SubBytes()* replaces bytes using a fixed S-box to achieve non-linearity. *Q: Why non-linearity?*

Big Idea #2 *ShiftRow()* and *MixColumns()* are intended to mix up bits to achieve a wider distribution of plaintext in the whole message space.

Big Idea #3 *AddRoundKey()* provides the necessary secret randomness.

Q: How do these transformations relate to the Big Ideas?

Preliminaries

We will consider the minimum case of 128-bit key.

- The input and output arrays consist of sequences of 128 bits represented by a 4 x 4 matrix of 8-bit bytes over $\mathrm{GF}(2^8)$.
- The intermediate state is referred to as the state array.
- Columns and rows are also referred to as words which consist of 4 bytes.

input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

$\rightarrow$

State array

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$\rightarrow$

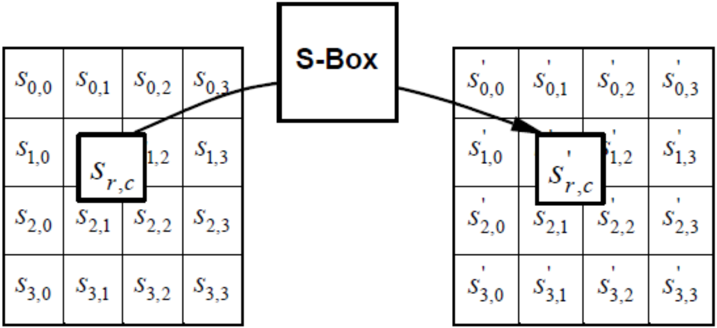
output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

All AES images come from FIPS Pub 197 available at
`http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

SubBytes()

Non-linear byte substitution that operates independently on each byte of the state array using a substitution table (S-box).



SubBytes() S-box

Example: SubBytes(45) = 6e

- Rows: First 4 bits of the input byte
- Columns: Last 4 bits of input

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
x	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

SubBytes()

Each byte x is substituted by y, where $y = Ax^{-1} + b$ if $x \neq 0$ and $y = b$ if $x = 0$.

$$A = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&0 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \qquad b = \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix}$$

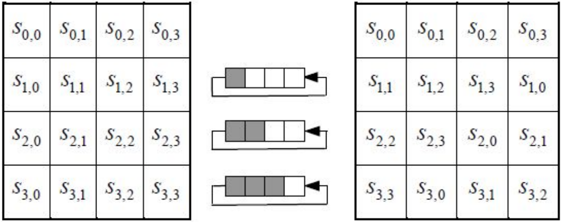
The S-box on the previous slide is just a pre-computed table. It eliminates the possibility of a timing analysis attack:

- Observing the time difference may give out whether an operation is performed on a zero or a non-zero byte.

ShiftRows()

The bytes are cyclically shifted over by 0, 1, 2 and 3 bytes.

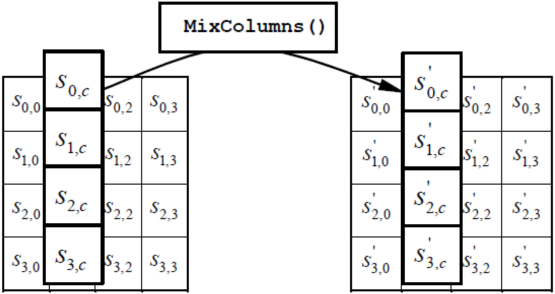
This operation works like a transposition cipher because only the positions of bytes are changed, not the bytes themselves.



MixColumns()

Operates on the state array column-by-column.

Each column is multiplied by a fixed array.



Matrix multiplication

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

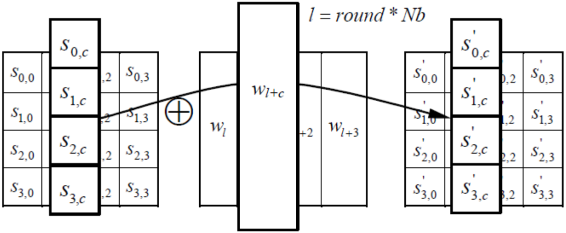
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

$\oplus$ exclusive OR operation, $\bullet$ finite field multiplication

AddRoundKey()

Each column of the state array is XORed with a word from the key schedule.

The round key is determined by the key schedule algorithm.



Nb - number of columns, here Nb = 4

Decryption

AES is not a Feistel cipher.

Q: How does it affect the decryption process?

Decryption

AES is not a Fiestel cipher so decryption works differently than encryption. Steps are done in reverse order.

All four transformations (SB, SR, MC, ARK) are invertible.

- InvSubBytes()
- InvShiftRows()
- InvMixColumns()
- AddRoundKey()

Decryption

- InvSubBytes() - the inverse S-box is applied to each byte of the state array.
- InvShiftRows() - bytes in the last three rows of the state array are cyclically shifted over to the right.
- InvMixColumns() - the state array is multiplied by the matrix inverse used in MixColumns().
- AddRoundKey() is its own inverse, since it is an XOR operation.

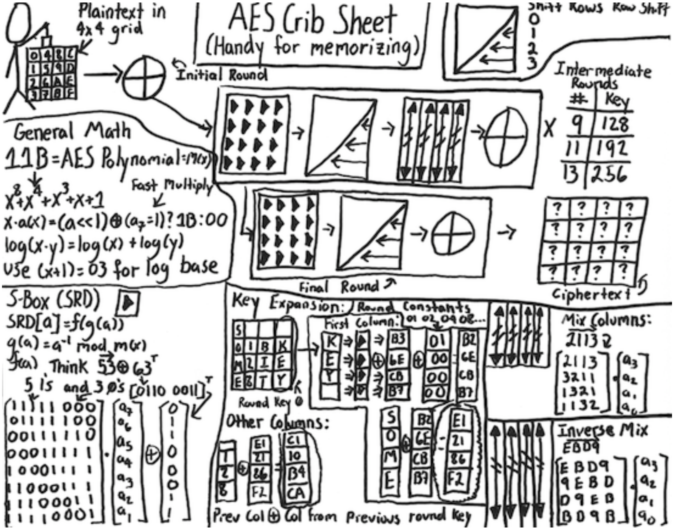
Encryption

- ARK
- SB, SR, MC, ARK
- . . .
- SB, SR, MC, ARK
- SB, SR, ARK

Decryption

- ARK, ISR, ISB
- ARK, IMC, ISR, ISB
- . . .
- ARK, IMC, ISR, ISB
- ARK

MixColumns() is not applied in the last round in order to make the encryption and decryption more similar in structure. This is similar to the absence of the swap operation in the last round of the DES.



www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Additional Resources

A Stick Figure Guide to AES by Jeff Moser Highly recommended!

`http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html`

AES Inspector by Enrique Zabala

`http://www.formaestudio.com/rijndaelinspector/archivos/inspector.html`

AES Animation by Enrique Zabala

`http://www.formaestudio.com/rijndaelinspector/archivos/rijndaelanimation.html`

AES Example by instructors at Massey U., New Zealand

`http://www.box.net/shared/static/uqrq0hmnb9.pdf`

AES Real-World Issues

Hardware Support for AES

Both Intel and AMD provide a set of instructions for AES promising 3x to 10x acceleration versus pure software implementation.

- `AESENC/AESDEC` - one round of encryption / decryption
- `AESENCLAST/AESDECLAST` - last round of encryption / decryption
- `AESKEYGENASSIST` - key expansion

Breaking AES

The ability to recover a key from known or chosen ciphertext(s) with a reasonable time and memory requirements.

Frequently, reported attacks are attacks on the implementation, not the actual cipher:

- Buggy implementation of the cipher (e.g., memory leakage)
- Side channel attacks (e.g., time and power consumption analysis, electromagnetic leaks)
- Weak key generation (e.g., bad PRBGs, attacks on master passwords)
- Key leakage (e.g., a key saved to a hard drive)

AES Security: Keys

Certain ciphers (e.g., DES, IDEA, Blowfish) suffer from weak keys.

A *weak key*[4] is a key that makes a cipher behave in some undesirable way. A cipher with no weak keys is said to have a *flat*, or *linear*, key space.

DES, unlike AES, suffers from weak keys (alternating 0's and 1's, F's and E's, E's and 0's, 1's and F's).

DES weak keys produce 16 identical subkeys.

Q: Why are DES weak keys a problem?

[4] Wikipedia: Weak Keys

AES Security

Attacks have been published that are computationally faster than a full brute force attack.

Q: What is the complexity of a brute force attack on AES-128?

AES Security

All known attacks are computationally infeasible.

Interesting results:

- Best key recovery attack: AES-128 with computational complexity $2^{126.1}$; AES-192, $2^{189.7}$; and AES-256, $2^{254.4}$.

  A. Bogdanov, D. Khovratovich and C. Rechberger, *Biclique Cryptanalysis of the Full AES*, ASIACRYPT 2011

- Related-key attack on AES-256 with complexity 2^{99} given 2^{99} plaintext/ciphertext pairs encrypted with four related keys.

  A. Biryuko and D. Khovratovich, *Related-key Cryptanalysis of the Full AES-192 and AES-256*, ASIACRYPT 2009

AES Security

Allegedly, NSA is actively looking for ways to break AES.

"Prying Eyes: Inside the NSA's War on Internet Security", Spiegiel, 12/2014

(TS//SI//REL) **TUNDRA** -- Electronic codebooks, such as the Advanced Encryption Standard, are both widely used and difficult to attack cryptanalytically. NSA has only a handful of in-house techniques. The TUNDRA project investigated a potentially new technique -- the Tau statistic -- to determine its usefulness in codebook analysis. This project was supported by [redacted] of R21.

Bruce Schneier on AES security[5]

"I don't think there's any danger of a **practical** attack against AES for a long time now. Which is why the community should start thinking about migrating now" (2011)

"Cryptography is all about **safety margins**. If you can break n round of a cipher, you design it with $2n$ or $3n$ rounds. At this point, I suggest AES-128 at 16 rounds, AES-192 at 20 rounds, and AES-256 at 28 rounds. Or maybe even more; we don't want to be revising the standard again and again" (2009)

[5] Bruce Schneier's Blog `http://www.schneier.com/blog/archives/2011/08/new_attack_on_a_1.html`

Alternative Private Key Block Ciphers

Other ciphers

There are many good block ciphers to choose from:

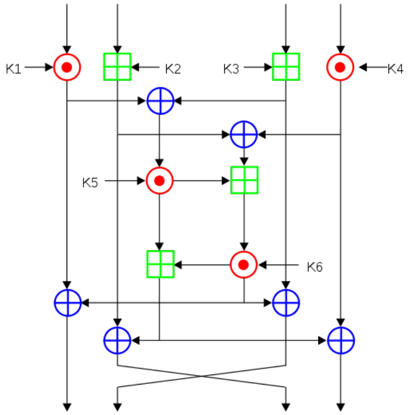
- Blowfish, Serpent, Twofish, Camellia, CAST-128, IDEA, RC2/RC5/RC6, SEED, Skipjack, TEA, XTEA

We will have a brief look at

- IDEA
- Blowfish
- RC6
- TEA

IDEA (International Data Encryption Algorithm)

- Invented by James Massey
- Supports 64-bit data block and 128-bit key
- 8 rounds
- Novelty: Uses mixed-mode arithmetic to produce non-linearity
  - Addition mod 2 combined with addition mod 2^{16}
  - Lai-Massey multiplication ˜multiplication mod 2^{16}
- No explicit S-boxes required



multiplication modulo $2^{16}+1$ bitwise XOR addition modulo 2^{16}

Image retrieved from `http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm`

Blowfish

- Invented by Bruce Schneier
- Supports 64-bit data block and a variable key length up to 448 bits
- 16 rounds
- Round function uses 4 S-boxes which map 8 bits to 32 bits
- Novelty: the S-boxes are key-dependent (determined each time by the key)

RC6

- Invented by Ron Rivest
- Variable block size, key length, and number of rounds
- Compliant with the AES competition requirements (AES finalist)
- Novelty: data dependent rotations
  - Very unusual to rely on data

TEA (Tiny Encryption Algorithm)

- Invented by David Wheeler and Roger Needham
- Supports 64-bit data block and 128-bit key
- Variable number of rounds (64 rounds suggested)
  - "Weak" round function, hence large number of rounds
- Novelty: extremely simple, efficient and easy to implement

TEA Encryption(32 rounds)

$(K[0], K[1], K[2], K[3]) =$ 128 bit key
$(L, R) =$ plaintext (64-bit block)
delta $=$ 0x9e3779b9
sum $= 0$
for $i = 1$ to 32
    sum $=$ sum $+$ delta
    $L = L + (((R \ll 4) + K[0]) \oplus (R + \text{sum}) \oplus ((R \gg 5) + K[1]))$
    $R = R + (((L \ll 4) + K[2]) \oplus (L + \text{sum}) \oplus ((L \gg 5) + K[3]))$
next i
ciphertext $= (L, R)$

TEA Decryption

$(K[0], K[1], K[2], K[3]) =$ 128 bit key
$(L, R) =$ ciphertext (64-bit block)
delta $=$ 0x9e3779b9
sum $=$ delta $\ll 5$
for $i = 1$ to 32
    $R = R - (((L \ll 4) + K[2]) \oplus (L + \text{sum}) \oplus ((L \gg 5) + K[3]))$
    $L = L - (((R \ll 4) + K[0]) \oplus (R + \text{sum}) \oplus ((R \gg 5) + K[1]))$
    sum $=$ sum $-$ delta
next i
plaintext $= (L, R)$

Figures retrieved from *Information Security Principles and Practice*, Mark Stamp, Wiley, 2006