

Advanced Encryption Standard (AES): Rijndael Algorithm

Abstract

After an introduction to general aspects of DES and on its history, the AES Rijndael algorithm is presented. Some notes on security aspects of Rijndael and known attacks conclude this section.

Outline

- **A bit of history and the basics**
 - The Rijndael algorithm: encryption
 - The Rijndael algorithm: decryption
 - Security of Rijndael

A Bit of History (1)

- **1997:** The National Institute of Standards and Technology (NIST) put out a call for new algorithms to replace DES
 - ◆ to be *"an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century"*
 - ◆ the algorithms were all to be block ciphers, supporting a block size of 128 bits and key sizes of 128, 192, and 256 bits
- **1998:** 15 different designs were submitted from several countries
 - ◆ CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, and Twofish
- **1998:** The cryptography community was asked to comment about the 15 candidate algorithms received from the public call
 - ◆ NIST held two public conferences open to all researchers (AES1, AES2)
- **1999:** Five finalists were selected with a transparent process
 - ◆ **MARS** (IBM), **RC6** (RSA Labs.), **Rijndael**, **Serpent**, **Twofish**

A Bit of History (2)

- **2000**: final NIST AES3 conference, where a representative of each of the final five teams made a presentation defending their design
- **2000**: NIST announced that **Rijndael** (by Vincent Rijmen and Joan Daemen, Belgium) had been selected as the proposed AES
- **2001**: NIST announced that the new Advanced Encryption Standard (AES) was approved (US FIPS PUB 197)
- **2002**: AES effective as an US federal government standard
- **AES is the first (and only) publicly accessible cipher approved by the National Security Agency (NSA) for top secret information** (when used in an NSA approved cryptographic module)

AES Basics

- AES is a subset of the Rijndael cipher by V. Rijmen and J. Daemen
 - ◆ Rijndael is a family of ciphers with different key and block sizes (any multiple of 32 bits in range 128-256 bits)
 - ◆ for AES, NIST selected three members of the Rijndael family with fixed block size 128 bits, but three different key lengths: 128, 192 and 256 bits
- The correct pronunciation of *Rijndael* is Dutch (Flemish accent)
- Rijndael is based on a **substitution-permutation network** (not a Feistel network)
- Rijndael is fast both in software and hardware
- The Rijndael algorithm is based on repetitions of *transformation rounds* (cycles) applied on the **128-bit block** to encrypt
 - ◆ **10, 12, 14 rounds** respectively for **128-bit, 192-bit, 256-bit keys**
- Rijndael can be used in any mode (ECB, CBC, CFB, OFB, CTR)

Outline

- A bit of history and the basics
- **The Rijndael algorithm: encryption**
- The Rijndael algorithm: decryption
- Security of Rijndael

The Rijndael Algorithm

- We focus on the simplest case: 10 rounds, 128-bit key, 128-bit blocks
- Each round i
 - ◆ uses a round key $K(i)$, derived from the master key K
 - all keys (128 bit) are expressed as 4×4 matrix of bytes
 - ◆ processes a 128-bit block and outputs a 128-bit block
 - expressed as 4×4 matrix of bytes (*state*)
- All calculations are done in a finite Galois field $GF(2^8)$
 - ◆ elements of $GF(p^n)$ may be effectively represented as
 - polynomials $a(x)$ of degree $< n$ and coefficients $\in \mathbb{Z}_p$
 - $a(x) \pmod{R(x)}$, where $R(x)$ is an irreducible polynomial of degree n and coefficients $\in \mathbb{Z}_p$
 - ◆ in Rijndael $GF(2^8)$: $R(x) = x^8 + x^4 + x^3 + x + 1$
 - ◆ elements of $GF(2^8)$ can be expressed as 8 bits (1 byte)
 - ◆ in $GF(2^8)$: addition, multiplication, inversion of elements are defined

The Rijndael Algorithm: Encryption

Structure of Rounds



- Rounds are based on four steps (*layers*)
 - Sub Bytes (SB)
 - a non-linear substitution, where each byte is replaced with another according to a lookup table
 - designed for resistance to differential and linear analysis
 - Shift Rows (SR)
 - a transposition mixing step where the last three rows of the state are shifted cyclically a certain number of steps
 - causes diffusion over multiple rounds
 - Mix Column (MC)
 - another mixing operation which operates on the columns of the state, combining the four bytes in each column
 - causes diffusion over multiple rounds
 - Add Round Key (ARK)
 - the round key $K(i)$ is bitwise XORed with the current state

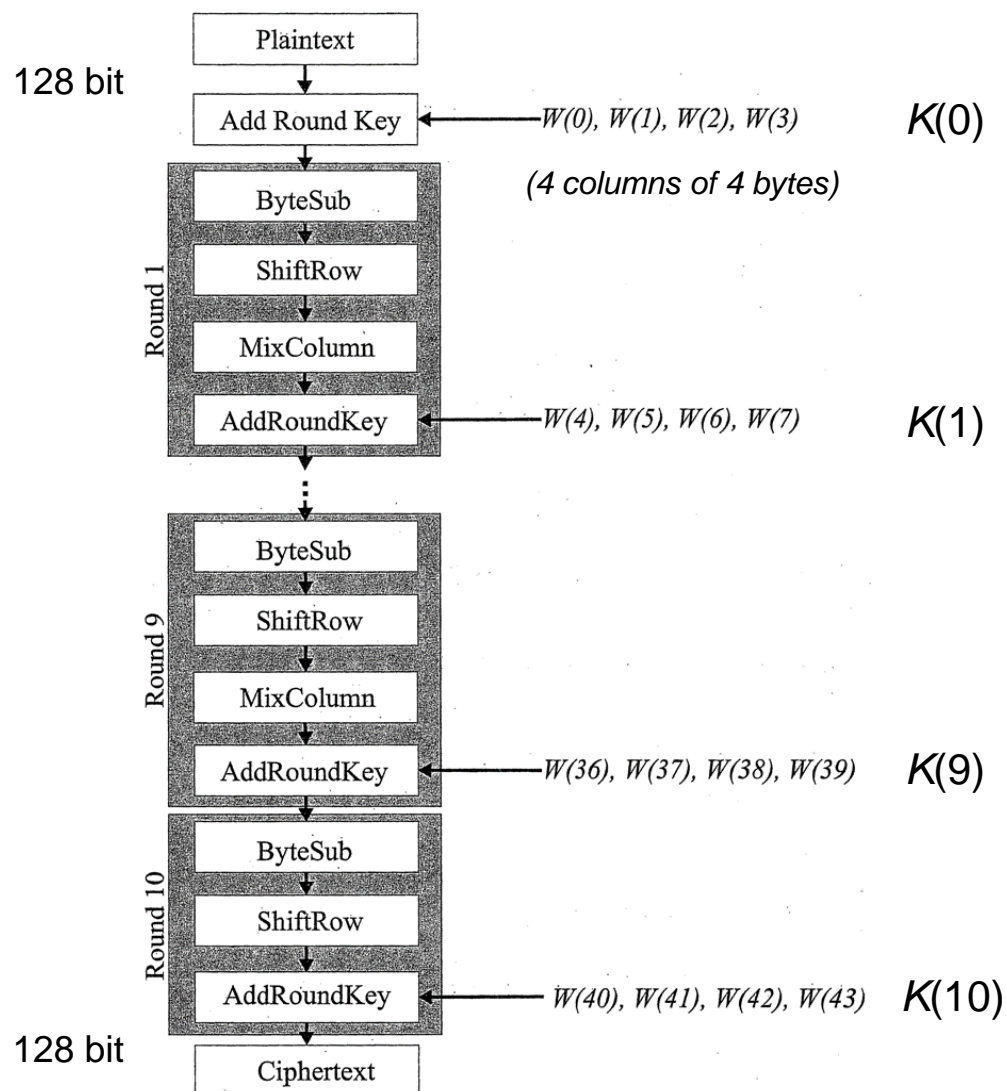
The Rijndael Algorithm: Encryption

Sequence of Rounds and Layers



- The input of each round is a block of 128 bits expressed as a 4×4 matrix of bytes

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



The Rijndael Algorithm: Encryption

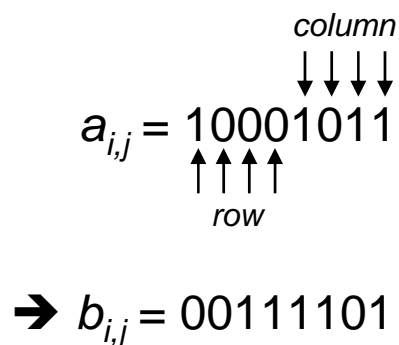
The Sub Bytes Transformation



Non-linear substitution

- each byte in the state matrix $a_{i,j}$ is substituted with another $b_{i,j}$ using an 8-bit lookup table (Rijndael S-Box) to yield

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



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

The Rijndael Algorithm: Encryption

The Shift Rows Transformation



- The four rows of the state matrix are shifted cyclically to the left by offsets 0, 1, 2, 3, to obtain

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

- To avoid the columns being encrypted independently (AES would degenerate into four independent block ciphers)

The Rijndael Algorithm: Encryption

The Mix Columns Transformation



- Regarding each byte as an element of $\text{GF}(2^8)$, the state is multiplied by another matrix, to obtain

$\mathbf{D} = \mathbf{MC}$

$$\begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} \cdot \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix}$$

- The four bytes of each column of the state are combined using an invertible linear transformation

The Rijndael Algorithm: Encryption

Add Round Key



- The i -th round key $K(i)$, derived from the master key K according to a *key schedule* algorithm, is bitwise XORed with the state matrix

$$\mathbf{E} = \mathbf{D} \oplus \mathbf{K}(i)$$

$$\mathbf{E} = \begin{pmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{pmatrix} = \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \oplus \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}$$

The Rijndael Algorithm: Encryption

Key Schedule



- 11 round keys $\mathbf{K}(i)$ ($i=0, \dots, 10$) are needed
 - 4+40 columns of 4 bytes $\mathbf{w}(i)$
- The 44 columns $\mathbf{w}(i)$ are obtained as
 - $\mathbf{K} = \mathbf{K}(0) = [\mathbf{w}(0), \mathbf{w}(1), \mathbf{w}(2), \mathbf{w}(3)]$
 - $\mathbf{w}(i) = \mathbf{w}(i-4) \oplus \mathbf{w}(i-1)$ if i is not a multiple of 4
 - $\mathbf{w}(i) = \mathbf{w}(i-4) \oplus T[\mathbf{w}(i-1)]$ if i is a multiple of 4
- The transformation $T[(a,b,c,d)]$ is defined as

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \rightarrow \begin{pmatrix} b \\ c \\ d \\ a \end{pmatrix} \xrightarrow{\text{SubBytes}} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix} \rightarrow \begin{pmatrix} e \oplus r(i) \\ f \\ g \\ h \end{pmatrix}$$

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

- The i -th round key is $\mathbf{K}(i) = [\mathbf{w}(4i), \mathbf{w}(4i+1), \mathbf{w}(4i+2), \mathbf{w}(4i+3)]$

The Rijndael Algorithm: Encryption

Construction of the S-Box



- The S-Box consists of a lookup table 16×16 to substitute $a_{i,j} \rightarrow b_{i,j}$
 - ◆ $a_{i,j} = x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$ (8-bit word)
 - ◆ $b_{i,j}$ is the 8-bit word at row $x_7 x_6 x_5 x_4$ and column $x_3 x_2 x_1 x_0$
- It is derived by a simple mathematical procedure
 - ◆ input: $a_{i,j} = x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$
 - ◆ compute $a_{i,j}^{-1}$ in $\text{GF}(2^8) = y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0$ (let $00000000^{-1} = 00000000$)
 - ◆ represent $y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0$ as a vertical vector $(y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7)^T$
 - ◆ compute $b_{i,j} = z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0$ as vector $(z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7)^T$
- To resist attacks
 - ◆ the inverse function is combined with an invertible affine transformation
 - ◆ the S-box does not have any fixed points neither any opposite fixed points

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

The Rijndael Algorithm: Encryption Design Remarks



- Rijndael is not based on a Feistel scheme
 - ◆ in Feistel, half of the block is swapped but not modified in each round, while in Rijndael all the block is scrambled uniformly
 - ➔ in Rijndael, two rounds are sufficient to obtain **full diffusion**
- The S-Box is built in an explicit algebraic way (**no secret trapdoor**)
- The S-Box is **highly non-linear** due to $x \rightarrow x^{-1}$ in $GF(2^8)$
 - ◆ excellent at resisting to differential, linear and interpolation analysis
- Shift Rows was added to resist to two more attacks (truncated differentials and Square attack)
- Mix Columns causes **diffusion** among all the bytes
- Key Schedule causes nonlinear mixing (S-Box) of key bits: **confusion**
- 10 rounds, because no attacks are known which are better than brute force for 7 or more rounds

Outline

- A bit of history and the basics
- The Rijndael algorithm: encryption
- **The Rijndael algorithm: decryption**
- Security of Rijndael

Inverting the Steps of the Rijndael Algorithm

- Each of the four layers is invertible
 - the inverse of Sub Bytes is another lookup table: [Inv Sub Bytes \(ISB\)](#)
 - the inverse of Shift Rows is obtained by shifting the rows to the right instead of to the left: [Inv Shift Rows \(ISR\)](#)
 - the inverse of Mix Column exists because the matrix **M** used in Mix Column is invertible (**M**⁻¹): [Inv Mix Column \(IMC\)](#)

$$\mathbf{D} = \mathbf{M}\mathbf{C}$$

$$\mathbf{C} = \mathbf{M}^{-1}\mathbf{D}$$

$$\begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix} = \begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix} \cdot \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix}$$

- Add Round Key (ARK) is its own inverse ([ARK](#))

Sequence of Steps in The Rijndael Algorithm

- The Rijndael encryption consists of these steps over 0+10 rounds
 - ◆ K_0 ARK
 - ◆ K_1 SB SR MC ARK
 - ◆
 - ◆ K_9 SB SR MC ARK
 - ◆ K_{10} SB SR ARK
- To decrypt, the inverse steps in reverse order must be run
 - ◆ K_{10} ARK ISR ISB
 - ◆ K_9 ARK IMC ISR ISB
 - ◆
 - ◆ K_1 ARK IMC ISR ISB
 - ◆ K_0 ARK

Running the Inverse Steps

- Applying SB then SR is the same as applying SR then SB
→ the order of ISR and ISB can reversed
- On encryption: $E = D \oplus K = MC \oplus K$
On decryption: $C = M^{-1}E \oplus M^{-1}K = M^{-1}E \oplus K'$ (with $K' = M^{-1}K$)
- Defining: ARK: $\oplus K$ IARK: $\oplus K'$
→ the inverse of (MC, ARK) is (IMC, IARK)



Decryption sequence

- | | | |
|-------------------------|----------------------------|--------------------------|
| ♦ K_{10} ARK ISR ISB | ♦ K_{10} ARK ↺ ISB ISR | ♦ K_{10} ARK |
| ♦ K_9 ARK IMC ISR ISB | ♦ K_9 IMC IARK ↺ ISB ISR | ♦ K_9 ISB ISR IMC IARK |
| ♦ | ♦ | ♦ |
| ♦ K_1 ARK IMC ISR ISB | ♦ K_1 IMC IARK ↺ ISB ISR | ♦ K_1 ISB ISR IMC IARK |
| ♦ K_0 ARK | ♦ K_0 ARK | ♦ K_0 ISB ISR ARK |

Outline

- A bit of history and the basics
- The Rijndael algorithm: encryption
- The Rijndael algorithm: decryption
- **Security of Rijndael**

Security of the Rijndael Algorithm

- Until 2009, the only successful published attacks against the full AES were attacks on some specific implementations
- NSA reviewed all the AES finalists and stated that all of them were secure enough for U.S. Government non-classified data
- In 2003, the US Government announced that
The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use.
- By 2006, the best known attacks were effective on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys

Known Attacks to the Rijndael Algorithm

(i.e., anything faster than a brute-force attack)



- 2002: XSL plaintext attack
 - ◆ theoretical, but unworkable in practice: 8000 quadratic simultaneous equations with 1600 variables to recover an AES 128-bit key
- 2009: new "related-key" attack with complexity $2^{99.5}$
 - ◆ related-key attacks are not of concern in practice
- 2009: first "known-key distinguishing" attack vs. an 8-round AES-128
- 2011: first key-recovery attacks on full AES (biclique attack)
 - ◆ faster than brute force by a factor 4
 - 2^{126} operations for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256
 - the current best results in key recovery attack against AES
 - in the best case, minimum 9 PB storage for AES-128
- Snowden documents: NSA is investigating "tau statistic" attacks
- At present, no practical attack is known that would allow someone to decrypt AES without knowing the key (when correctly implemented)