Countering AES Static S-Box Attack

Behnam Rahnama
Department of Computer Engineering
T.C. Okan University
Istanbul, Turkey
brahnama@acm.org

Yunus Kıran
Dept. of Computer Info. System
European University of Lefke
Gemikonagi,North Cyprus
yunus@ieee.org

Raz Dara
Department of Computer Engineering
European University of Lefke
Gemikonagi, North Cyprus
raz.mohammadamin@gmail.com

ABSTRACT
The purpose of this research is to update AES with a dynamic s-box in order to counter the timing vulnerabilities that exist in AES. Static s-box allows data analyses attack and the eventual captures of sub-keys using inverse SubByte knowing inverse s-box. We propose a novel yet simple design and implementation with a very low overhead cost to existing AES in order to update s-box to a dynamic s-box depending on a variable sub-key at each round. This way, static data analysis of the program flow and data injection cannot affect the AES security.

Categories and Subject Descriptors.
E.3 Data Encryption – Private key cryptosystems.
D.4.6 Security and Protection (K.6.5) – Authentication, cryptographic controls, verification.

General Terms.
Algorithms, design, security, reliability.

Keywords.
Advanced Encryption Standard, AES Vulnerability, Dynamic S-Box

1. INTRODUCTION
Incessant expansions of the advancements in technology pertaining to multimedia fields, of which medical and the internet are encompassed, have facilitated an ease with which digital information can be adversely manipulated. The possibility of misuse, copying and distribution of digital information and documents in a detrimental manner undoubtedly causes great concern. Certain documents may contain highly sensitive and significant information which is susceptible to an assortment of threats and issues deriving from illegitimate data access, attacks or an interruption of security and privacy, therefore they necessitate rigorous protection. It is for this reason cryptography science has been researching techniques in consideration of safeguarding such important information by converting the data into ciphertext, rendering it unreadable by attackers or illegitimate users while being stored and transmitted over any kind of medium.

SIN '13, November 26 - 28 2013, Aksaray, Turkey
Copyright 2013 ACM 978-1-4503-2498-4/13/11…$15.00.
http://dx.doi.org/10.1145/2523514.2523544

Due to the technology development in various multimedia fields like medical, internet and etc., it has become very easy to play with digital information by copying and distributing digital documents. Those documents may contain very significant and sensitive information and would be in danger because of different threats and those issues that are caused by illegitimate data access, attacks or interruption to the security and privacy of documents that must be protected. For this reason the information security and privacy has become vital part of dealing with sensitive information in general. The cryptography science has been examining techniques used in order to protect such important information by turning the data into ciphertext unreadable by attackers or illegitimate users while it's stored and transmitted over any kind of medium.

One of the latest cryptographic algorithms still being used is the Advanced Encryption Standard (AES). It was invented by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, replacing the old Data Encryption Standard (DES). The algorithm was announced in 2001 by NIST and it was approved as a federal standard in May 2002 [1,2].

The algorithm uses a combination of primitives like, shift rows, Sub Bytes over many rounds (10 for a 128 bit key) and a Mix Column and then pre-compute of the combination of these primitives will appear in series of table lookups and XOR operations. The index of these tables is the XOR of a plaintext byte and a key byte so it must remain private but still the attacking process running on the same system can notice the variable timing of the AES encryption because of the cash behavior and narrowing down the possible values for the key.

Rijindael has published a large collection of block cipher and after a while the three block ciphers of AES (AES-128, AES-192 and AES -256) has been adopted from this collection. Each one of these cipher has 128-bit block size, with key size of 128, 192 and 256 bits respectively [*2*].

The Rijndael algorithm has been chosen as new Advanced Encryption Standard (AES) for several reasons. The purpose was to create an algorithm which is resistant against known attacks, easy and quick to code. The strength of AES depend on the inverse of the addition operation was itself making much of the algorithm easy to be implemented. In fact, every operation is invertible in design. In addition, the block size and key size can vary making the algorithm versatile. AES was originally designed for non-classified U.S. government information, but, due to its success, AES-256 has been utilized for top secret governmental information, and though AES is widely used and it is a standard of current data encryption but some weak points have been found out, especially time attack and s-box analysis attack that challenge its security.

2. COMPARITIVE STUDING OF AES AND OTHER SECURITY ALGORITHMS

A comparative studying between AES and other algorithms is presented in general to illustrate their gaps, advantages and disadvantages of each of them. Any question about the encryption security should be determined in the terms of how long, and what the cost will be to make an attack successful.

According to a research paper published by H. Alanazi et al [3] it is shown that AES is better and securer than DES and DES III is in nine factors. The first factor is the differences in the key length for each one of them, for example the key length of DES is 56bits, for DES III is ((for k1, k2 and k3) 168 bits, (k1 and k2 is same) 112 bits), and for AES it is (128,192 and 256 bits). The second factor is the cryptographic type; all of them have the same cipher type which is (symmetric block cipher). The third factor is the block size, the DES block size is 64 bits long and for AES 128 bits, 192 bits and 256 bits). The forth factor shows the Cryptanalysis resistance for example, DES is vulnerable to linear and differential to cryptanalysis. It also has weak substitution tables; DES III is also vulnerable to differential and brute force attacks by analyzing plaintext using differential cryptanalysis, but AES is a strong truncated differential, against differential, interpolation, linear, and square attacks. The fifth factor concerns the security; the DES security is proven inadequate, the DES III is also proven inadequate. Like DES while the AES is considered secure. The sixth factor is possible keys, the possible keys for DES is 2^{56}, for DES III are 2^{112} and 2^{168}, and for AES are 2^{128}, 2^{192} and 2^{256}. The seventh factor is the possibility of ASCII printable character keys for DES is 95^{7}, for DES III is 95^{14} and 95^{21} for AES is 95^{16},95^{24} and 95^{32}.the eighth factor is time check all possible keys required at 50 billion keys per second for a 56 bit key of DES it requires 400 days , for a 112- bit key of DES III it requires 800, and for a 128 bit key of AES :$5X10^{21}$ years. The ninth factor is the developed year of each one of them; DES was developed in 1977, DES III was developed in 1978, AES was developed in 2000 [3]

Notice that, the breaking time considered earlier concerns a sequential code on a work station. Current Intel CPUs to reach about 100 gigaflops and GPGPUs' performance reach up to 1.3 teraflops per card. It makes the brute force attack much faster than expected.

On the other hand, the presented work in [4] based on a mathematical study of AES and RSA with a short description of some cryptographic primitives proved that though AES gives us less implementation complexity and better security. It's considered as one of the most efficient algorithms and the strongest in existence today, but just like other symmetric encryption algorithms; the secret key regulation is considered as a critical issue for it. Asymmetric cryptosystem like RSA solves this problem but suffers from greater computational overhead because of its large key size so the best solution is to use the hybrid encryption system in which AES is used to encrypt large data block and digital signature application and RSA is used for the key management. Yet, RSA itself suffers from mathematical attacks as it relies on the theory of relatively prime numbers and therefore, it's easy to break them using personal supercomputers today.

The RC4 is fast and energy efficient for encryption and decryptions and it's also less computational intensive than AES according to a comparative analysis of both of them, which compares the AES algorithm with different modes of operation (block cipher) and RC4 algorithm (stream cipher) in terms of CPU time, encryption time, memory utilization and productivity at different settings like variable key size and variable data packet size [5].

3. AES SECURITY

Cryptographic attacks are invented to destroy the security of cryptographic algorithms where attackers try to decrypt data without former access to a key as cryptanalysis art of deciphering encrypted data. One of the latest cryptographic algorithms is the AES algorithm. AES algorithm has some vulnerability that leads to different attacks on its algorithm for instance: Cache-timing attacks, power analysis of key scheduling of AES etc.

According to a research published by Rahnama et al [6] AES security strength against brute force attack rated as an infeasible attack without supercomputing power because the AES cipher key is the smallest length of 128 bits which supplies 2^{128} possible keys and executing a whole search in this massive key space is rated as non-applicable and therefore a mathematical attack against the AES algorithm is more promising, while s-boxes contain static value for all rounds and for all entries. On the other hand AES is not totally resistant against different types of Side Channel Attacks including timing attack, cache collision timing attacks, electromagnetic radiation and power consumption attack. The timing attack is an application level attack that attacks cipher text only and could happen to any application that doesn't work in a steady time, rather than to rely on the input so the high speed AES software application is in danger against this kind of attack because it executes a series of S-box lookups relying on variable time among rounds, as the last round in AES encryption does not include the Mix Column stage. They have implemented an updated AES without initial Add Round Key and with an extra Mix Column stage to avoid variable timing detection. They also applied a play fair stage in every single round so it will use dynamic exchanging byte depending on input data, not depending on static replacement and exchanging tables [6]. However, it causes an additional overhead due to the extra stage embedded in AES rounds.

Cache-timing attacks on AES can also be achieved by taking the AES key from recognized-plaintext timings of a network from another computer, and according to the design of AES itself, caused this problem not the AES library which has been used by the server [7]. The obvious way to avoid leaking information is to make sure that s-box will be loaded to level-1 cache (because as we know all CPUs today have at least two or more cache levels and usually information traverses from level 2 to level 1 and the level 1 latency is shorter than that of level 2). There are four types of problems for achieving this :- 1st the (AES-box lines) may be pushed out of level 1 cache in order to make a space for other computations not only AES , 2nd the (AES-box lines) may be pushed out of level 1 cache in order to make a space for AES computations itself, 3rd the (s-box lines) may be pushed out of level 1 cache by other operations which stops AES computations , 4th saving all (s-box lines) in level 1 doesn't ensure a fixed time for s-box lookups [7]

Another Access-Driven Cache attack (which is a class of cache-based side channel analysis) presented by [8] observes and checks the cache changes in detail before assessing all of the runtime. Access-driven attacks influence the capability to reveal whether a cache line has been evicted or not, as the main technique for increasing an attack.

However, the aforementioned attack needs prior knowledge of timing behavior of the cipher algorithm, using the same recognized key on a similar platform. According to [9] an efficient attack on AES was done by Osvik et al when they explained and simulated several different techniques to do the local cache attacks. They utilized a local array, then they took advantage of the encounters between AES table lookups and accessed the array, they then retrieved the complete 128-bit AES key through two rounds attack for about 300 samples on AMD Athlon 64.

Though Neve et al illustrated a final round 128-bit AES cache attack in their method and it was the similar to Osvik et al's Prime plus Probe attack, it was not practical because they presumed that the key extension works just one time and it arises with numerous encryptions, as a matter of fact, AES key extension requires access 40 times *T*4 lookup table, and this may annoy the 16 times *T*4 lookups of AES last round encryption. In addition, none of these researches provided a recovery of the complete AES key at first round attack proficiently.

Francois Dassance et al [10] offered a combined fault and side channel attack on the AES key table depending on another work done by Roche et al. The major obstacles for the genuine attack are the high demand for recurrence errors, high feasible samples of these errors and high intricacy of key restoration algorithm.

Other known attacks unlike the origin time attack deals with the execution of the suggested attacks rather than effecting AES constants. This can improve the complexity of attack iterations from 2^{28} to 2^{12} that is equal to a classical differential side channel attack analysis key recuperation, but it still has two weak points, first the suggested check on the origin attacks defeat most of their suggestions and second the path attack on RCon9 needs a specific safety check [10]

Eli Biham et al [11] provided a new method of achieving power analysis attacks based on Kochar power analysis targeting smart card applications and extract their keys by analyzing the power consumption graph .They improved a new alternative to this method that analyzes the power consumption through the key scheduling section of the block cipher. The lower power consumption boundaries limits the AES to be the key function and not the data function so it can be used for fielded smart card systems while data is hidden from the attackers. This method is very efficient against AES and DES block cipher but less efficient against Skipjack block cipher.

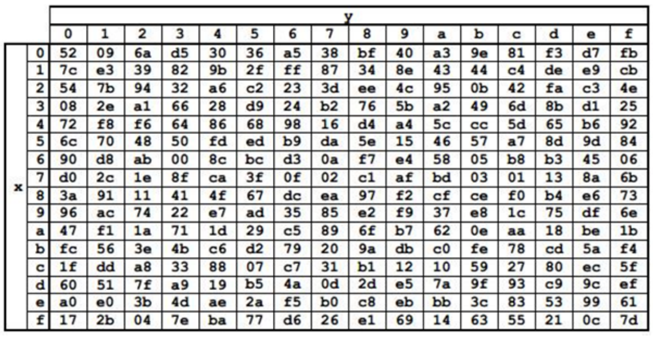
Finally, in [12] They made use of the order points of AES 256 key to analyze the modification properties of AES 256 and discovered a new technique to apply a collision attack. They analyzed the 72 bytes of the primary key of AES 256, with intricacy of 2^{128} and according to them "the possibility of its existence is 2^{-96}". They also provided new rectangle circuit sectionalizers for attached keys, and performed analysis for AES 256 security based on these circuit sectionalizers.

We have seen that the greatest risk attacks to AES algorithm is the side channel attacks because they have the ability to take back the secret key of AES encryption being executed on small devices like smart cards easily. Other names for side channel attacks are differential attacks covering power analysis attacks, timing attacks and so on. Side channel attack can be countered by applying security software applications or hardware protection algorithms. The other weak point of the AES algorithm is the cache memory processer leakage which causes the differential timing for memory access that attackers take advantage of and uses it to obtain information concerning the secret key.

In the next section, we propose our novel approach to fix the AES imperfection against s-box analysis attack. This also prevents the cache analysis attack as the content of s-box is updated at each round and therefore there is no single reference for all rounds.

4. DYNAMIC S-BOX APPROACH

The AES algorithm uses s-box to implement byte by byte substitution of the block. Basically s-box is 16 ×16 matrix and it works as a lookup table, and this matrix is a static table. In order to make the algorithm more secure against a timing attack and make its information kept safe and difficult to be discovered by attackers, we will change this static table in to dynamic table depending on the key that we have. For instance we have this s-box table as below.



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

Fig. 1. Sample of S-Box table

As shown in Fig. 1. the s-box consists of a 16 by 16 matrix of bytes. This can be represented as a 256 bytes array. The SubByte function extracts the value out of the static s-box matrix. In order to make the system dynamically updated at each round, we simply take the mirrored position sub-key of 16 bytes. The sub-key is filled at row 0 of the Dynamic s-box (Dbox) then it's circularly left shifted per each row by BuildLCS function.

```
void BuildLCS(byte* [] Data, int Times){
   for (i = 0; i < Times; i++){
      byte temp = Data[0];
      for(int j = 0; j < 15; Data[j] = Data[j+1],
j++);
      Data[15] = temp;
   }
}
```

A combination of 16 lines of circularly shifted rows creates the DBox. This matrix is changed per each row as the basic ingredient of it as the sub-key is changed per round. Then it is updated by applying the byte-wise XOR with s-box. The generated AddByte is dynamically changed per round. It is applied to the s-box table and as a result it generates a different box per each round. The inverse box is generated accordingly for decryption phase.

```
void DBoxGen(byte [,] statebox, int round){
   byte [] Temp = new byte[16];
   BuildLCS(ref Temp, round);
   byte AddByte = 0; t = 0;
   for(byte k = 0;k < 16; k+2){
      AddByte << = 1;
      t = (byte)(Temp[k] & 0x01);
      AddByte | = t;
   }
   for(byte i = 0; i < 16; i++){
      for(byte r = 0; r < 4; r++)
         for(byte c = 0; c < 4; Temp[r*4+c] = w[((Nr-
         round)*4)+r,c], ++c);
      for(byte j = 0; j < 16; j++){
         byte row,col,y=0;
         Dbox[i,j]=(byte)((int)statebox[i,j]^
         (int)AddByte);
         row=(byte)(Dbox[i,j]>>4);
         col=(byte)(Dbox[i,j]&0x0f);
         iDbox[row,col]=(byte)((int)(i<<4)|(int)j);
      }
   }
}
```

The first transformation is used in the sub bytes encryption where the SubBytes operation involves 16 independent byte-to-byte transformations and in order to substitute a byte, we interpret the byte as two hexadecimal digits.

D-box contains source values for dynamic mode while s-box includes source values for the conventional AES routines. A Boolean Mode variable can be taking true or false values enumerated as Standard and Dynamic terms.

SubBytes function acts as before in case Standard is selected for Mode. Upon calling SubByte with Dynamic Mode value it generates D-box and then it applies XOR of 4 by 4 matrices with d-box rather than s-box. This simple trick avoids attackers analyzing and breaking the payload when it accesses static s-box.

The following implementation is based on an open source AES project available at [13]

```
void SubBytes(int round)
{  int i;
   if (Mode)
      DBoxGen (Sbox, round) ;
   for(r = 0; r < 4; r++) {
      for(c = 0; c < 4; c++) {
      State[r,c] = Dbox[( State[r,c])] >> 4),
(State[r,c] & 0x0f) ];
}
}
```

Similar to SubByte, the InvSubByte works with a Boolean variable whether to work in conventional mode or in the proposed dynamic mode.

```
void InvSubBytes(int round)
{
   If (mode)
      BuildDbox(Sbox,round);
   for(int r = 0; r < 4; ++r){
      for(int c = 0; c < 4; ++c){
         if (mode)
            State[r,c] = iDbox[(State[r,c] >> 4),
(State[r,c] & 0x0f)];
      }
   }
}
```

The encryption function is updated in order to call the updated subbytes with an extra parameter either as (Standard or Dynamic). These two words are predefined literals to force the algorithm working in the old fashion or with Dynamic S-Box updates. The other steps including AddRoundKey, ShiftRows and MixColumns are the same. Our proposed approach only updates the Subbytes routine.

```
void Cipher(byte [] input, byte [] output)
{
   State = new byte[4, Nb];
   for(int i = 0; i < (4*Nb); ++i)
      State[i%4, i/4] = input[i];
   AddRoundKey(0);
   int round;
   for(round = 1; round <= (Nr-1); ++round)
   {
      SubBytes(round);
      ShiftRows();
      MixColoums();
      AddRoundKey(round);
   }
   SubBytes(Nr);
   ShiftRows();
   AddRoundKey(Nr);
   for (int i = 0; i < (4*Nb); ++i)
   {
      output[i] = State[i%4, i/4];
   }
}
```

Iterations and operations are involved in the encryption and the decryption process depending on the cipher key size where each round accomplishes four known stages. The 128-bit cipher key requires 10 rounds to accomplish either the encryption or decryption phases. In AES 128-bit, 11 sub-keys are deduced from the master key. But the key must be expanded prior to decryption. Encryption is thus faster than Decryption. The decryption is updated with just a similar amendment to the encryption function.

```
void InvCipher(byte [] input, byte [] output)
{
   State = new byte[4,Nb];
   for(int i = 0; i < (4*Nb); ++i)
      State[i%4, i/4] = input[i];
   AddRoundKey(Nr);
   for(int round = Nr-1; round >= 1; --round)
```

```
    {
       InvShiftRows();
       InvSubBytes(round+1);
       AddRoundKey(round);
       InvMixColoums();
    }
    InvShiftRows();
    InvSubBytes(1);
    AddRoundKey(0);
    for (int i = 0; i < (4*Nb); ++i)
       output[i] = State[i%4,i/4];
}
```

5. CONCLUSION

In this paper we firstly covered a concise study on various AES attacks. In addition we provided a simple implementation of our novel approach to update the s-box table per each round based on a mirrored position sub-key to protect AES from s-box analysis and similar side channel attacks. The implementation is provided in C language in detail suitable for both computers and embedded targets.

6. REFERENCES

[1] Svante Seleborg. (2007, August) Yudu. [Online]. http://content.yudu.com/Library/A1mykn/AboutAdvancedEncrypt/resources/index.htm

[2] Douglas Selent, "Advanced Encryption Standard," *Rivier Academic Journal*, vol. 6, 2010.

[3] Hamdan.O Alanazi, B.B Zaidan, Hamid A Jalab, M Shabbir, and Y. Al-Nabhani, "New Comparative Study Between DES,3DES and AES within Nine Factors," *Journal Of Computing*, vol. 2, no. 3, March 2010.

[4] Abdullah Al Hasib and Abul Ahsan Md. Mahmudul Haque, "A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography," in *Third International Conference on Convergence and Hybrid Information Technology*, 2008.

[5] Singhal and J.P.S. Raina, "Comparative Analysis of AES and RC4 Algorithms for Better Utilization," *International Journal of Computer Trends and Technology*, no. 2011.

[6] Behnam Rahnama, Atilla Elci, and Ibukun Eweoya, "Fine Tuning the Advanced Encryption Standard (AES)," in *Fifth International Conference on Security of Information and Networks*, 2012.

[7] Daniel J. Bernstein. (2004) yp. [Online]. http://cr.yp.to/antiforgery

[8] Michael Neve and Jean-Pierre Seifert, "Advances on Access-Driven Cache Attacks on AES," in *13th international Conference on Selected areas in cryptography*, 2006, pp. 147-162.

[9] Zhao Xinjie, Wang Tao, Mi Dong, Zheng Yuanyuan, and Lun Zhaoyang, "Robust First Two Rounds Access Driven Cache Timing Atta ck on AES," in *International Conference on Computer Science and Software Engineering*, 2008.

[10] Francois Dassance and Alexandre Venelli, "Combined Fault and Side-Channel Attacks on the AES Key Schedule," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2012.

[11] Eli Biham and Adi Shamir. cs. [Online]. http://www.cs.technion.ac.il

[12] Hu Zhihua and Qin Zhongping, "Related-Key Collision Attack of AES_256," in *International Symposium on Intelligence Information Processing and Trusted Computing*, 2012.

[13] x-n2o Blog. [Online]. http://x-n2o.com/aes-explained