AES: side-channel attacks for the masses. (rev 0.2)

Victor Muñoz
vmunoz@ingenieria-inversa.cl

October 2007

Abstract.

AES (Rijndael) has been proven very secure and resistant to cryptanalysis, there are not known weakness on Rijndael algorithm up to day. But there are some practical ways to break weak security systems that rely on AES.

Introduction.

AES has been subject to exhaustive cryptanalysis efforts, but none of them could break the cipher.

The newest attacks can break only short-cut versions of AES, with a reduced number of rounds (up to 9 rounds on AES-192), the most fruitfully techniques used were Collision Attack, Square Attack, Impossible Differential, Truncated Differential and Related Key, you could see a summary of the cipher breaking level of such techniques in [1], and see a briefly description of some of them in [2].

The most practical attacks on AES are side-channel attacks, that don't intend to attack the algorithm itself, but look to reconstruct the key from secret leakage through the physical implementation of the algorithm; such leak of information could be –among others- Power Consumption, Time, Electromagnetic Radiation, and etcetera.

In AES breaking quest Simple Power Analysis and Differential Power Analysis were used roughly on attacks to smart-cards as stated in [x]. Also Cache Timing Attacks are well known, but seem a little hard to use it in real world situations, also they may need clock cycle level accuracy in the timing measurements, and big amounts of sampling, those Cache Timing Attacks do not seems feasible for other scenarios than process-to-process attacks (ie: remote key retrieval).

Suppose you are in a dealing with a process-to-process situation, that means that your offensive process has some access to the overall system, then why to bother to use a complex attack when you could use some other meaning to spot AES keys in no time?.

In this document we will see 2 methods for attack AES that should work with no problem in real world situations and are

not exclusively for neither laboratory experiments nor concept proofs.

Those attacks are intended to retrieve an AES key when you have physical access to the machine you want to attack, one method require you have full access to the system meaning you could install a debugger or exception handler, and full access to the process you want to attack.

The second method is simpler to implement and you only need to have reading access to memory of the victim process, extending this method you could gain access to AES key directly from the RAM IC modules assuming the RAM is not encrypted, the AES implementation is software based, and of course all the key processing is not fit just in the internal CPU data cache.

Why could you be interested to attack machines that you own and not a third party victim? Simple, there exists lot of boxes that come locked (and limited) only to run the software singed for the box vendor, machines like videogame consoles, set-top boxes, cell phones, routers, etc.

Such key retrieving activity has been very useful –for example- in the efforts to circumvent DRM schemes like AACS, that rely strongly on AES, your AACS licensed player software hides you the keys needed for decode a movie, and that simply prevent you to make your own media player or see your movies in any free operating system, moreover you could not see a HD movie at full resolution in a non HDCP licensed (and yet expensive) monitor.

Easy AES key retrieval History.

Let's begin with a little of history, *muslix* (the former hacker of AACS system) [4], has got the keys needed to consider AACS cracked back in December 2006 without the need for tracing or debugging any bit of code, the method he used was simply guess the decrypted header of a video stream block and run a key finder in a memory dump of the process of the AACS enabled player software trying every 16 continuous bit as keys, and that lead him –just in seconds- to a VUK (Volume Unique Key) needed to decrypt the whole movie, and see it in any player, setup or OS that you want.

We are going to refer here to the above attack as known-plaintext/key within process memory (in rigor was guessed-plaintext and not known-plaintext).

This attack was recognized by the same AACS LA on January 24, 2007 [5], and from that moment AACS scheme was in fact full compromised.

Some months after the original attack, more attacks come to

the AACS scheme, all those attacks have something in common: AES key spotting with a little of effort in comparison with the state of art side-channel attacks on AES.

Reference

[1] http://www.iaik.tu-graz.ac.at/research/krypto/AES/ - IAIK Krypto Group - AES Lounge

[2] http://www.iaik.tugraz.at/aboutus/people/oswald/papers/aes_report.pdf - AES - The State of the Art of Rijndael's Security

[x]

[4] http://forum.doom9.org/showthread.php?t=119871

[5] http://www.aacsla.com/press/ January 24, 2007