International Journal of Innovative Research in Information Security (IJIRIS) ISSN: 2349-7017(O)
Issue 3, Volume 2 (March 2015) ISSN: 2349-7009(P)
www.ijiris.com

Data Cyber Security Network Encryption Impact on Government Application Performance

Ravishankar Belkunde
Senior Database Administrator & Researcher
Boston, Massachusetts, 02108

Abstract- Government applications are easy target of Cyber threats. Government agencies have multiple small applications it is difficult to maintain same cyber security for all Information systems. Protecting data from cyber attack with system performance upkeep becomes challenging. The encryption algorithms used for encryption should have minimum impact on application performance. Here we will discuss the performance results of application with different encryption algorithms combined with different checksum types and see the data cyber security network encryption algorithm and checksum impact on the government application performance.

Keywords: Encryption, Checksum, Cyber Security, Data encryption, Algorithms, Performance analysis.

I.INTRODUCTION

The encryption algorithm use for the security against Cyber attacks should not come at the cost of system performance. Practically system performance, implementation cost and application response time is as important as security. Here we will be comparing the encryption algorithms combined with different checksum impact on application. We will be discussing Data Encryption Standard 56 bit encryption method(DES), triple Data Encryption Standard with three-key(3 DES168), Advanced Encryption Standard with 128-bit key(AES128), Advanced Encryption Standard with 256-bit key(AES256), Rivest Cipher 4 with 128-bit key(RC4_128) and Rivest Cipher 4 with 256-bit key(RC4_256) algorithms combined with MD5 and SHA-1 checksum against Oracle 10.2 database. In this paper, the security issues have been completely ignored, and only the performance of the leading secret key algorithms has been compared on different platforms, using input data files of varying sizes and formats.

II.ENCRYPTION ALGORITHMS

The following encryption algorithms/protocols are used.

- DES
- TripleDES
- AES128(Rijndael)
- AES256
- RC4_128
- RC4_256

A. Data Encryption Standard (DES)

DES (Data Encryption Standard) is currently the most widely used block cipher in the world. In May 1973, NIST (then NBS) called for possible encryption algorithms for use in unclassified adopted encryption algorithm and is in many standards around the world (e.g. Australian Standard AS2805.5-l985). One of the largest users of the DES is the banking industry. It is for this use that the DES was primarily standardized, with ANSI reconfirming its use for 5 year periods - in future it will be replaced with AES. Although the DES standard is public, the design criteria used are classified. There has been considerable controversy over the design, particularly in the choice of a 56-bit key.

B. Triple DES (3DES)

The triple DES (3DES) algorithm was needed as a replacement for DES due to advances in key searching. 3DES is a proposal based on the existing DES, and was standardized in ANSI X9.l 7 & ISO 8732 and in PEM for key management. It was also proposed for general EFT standard by ANSI X9 [I]. It is backwards compatible with existing single DES (when Kl=K2=K3). The 3DES algorithm uses either two or three 56-bit keys. Thus the effective key length is up to 168 bits. 3DES is defined by the following function:

C = DESK3 {DES-1 K2{DESKt(P)}
where P = Plaintext C = Ciphertext

© 2014-15, IJIRIS- All Rights Reserved

DESK= DES encryption using key K
DES-1
K = DES decryption using key K

C. Advanced Encryption Standard (AES) 128/256

In September 1997, US NIST announced a call for candidate ciphers for its new Advanced Encryption Standard (AES), because clearly a replacement for DES was needed at that time [2]. The candidate ciphers were to be submitted by June 1998, and a finalist was selected in October 2000. In total 15 candidates were accepted in June 98 (6 were rejected as incomplete), and 5 were short-listed in August 99. Finally, Rijndael was selected as the AES finalist in October 2000.

NIST has released all submissions and unclassified analyses. The AES candidates are the latest generation of block ciphers, and have a significant increase in the block size - from the old standard of 64-bits up to 128-bits; and keys from 128 to 256-bits. In part this has been driven by the public demonstrations of exhaustive key searches of DES & RC-5 (at 64-bits). After exhaustive analysis and evaluation rounds Rijndael, designed by Rijmen&Daemen in Belgium [4], was selected as AES finalist [3]. It has the following attributes:

- 128-bit block size
- 128, 192, or 256 bit key size
- an iterative rather than a Feistel cipher (like IDEA)
- treats data as 4 groups of 4 bytes
- has 9, 11, or 13 rounds, where each round consists of:
- a byte substitution step (l S- box used
- on every byte)
- a shift rows step (shuffle the bytes
- between groups)
- a mix columns step (matrix
- multiplication of groups with each
- other)
- an add round key step
- all operations can be combined into xor and table lookups hence implementation can be very fast and efficient

C. RC4_128/256

RC4 was designed by Ron Rivest of RSA Security in 1987. While it is officially termed "Rivest Cipher 4", the RC acronym is alternatively understood to stand for "Ron's Code"[9]. RC4 was initially a trade secret, but in September 1994 a description of it was anonymously posted to the Cypherpunks mailing list.[10] It was soon posted on the sci.crypt newsgroup, where it was analyzed within days by Bob Jenkins.[11] From there it spread to many sites on the Internet. The leaked code was confirmed to be genuine as its output was found to match that of proprietary software using licensed RC4. Because the algorithm is known, it is no longer a trade secret. The name RC4 is trademarked, so RC4 is often referred to as ARCFOUR or ARC4 (meaning alleged RC4)[12] to avoid trademark problems. RSA Security has never officially released the algorithm;

RC4 generates a pseudorandom stream of bits (a keystream). As with any stream cipher, these can be used for encryption by combining it with the plaintext using bit-wise exclusive-or; decryption is performed the same way (since exclusive-or with given data is an involution).RC4 generates a pseudorandom stream of bits (a keystream). As with any stream cipher, these can be used for encryption by combining it with the plaintext using bit-wise exclusive-or; decryption is performed the same way (since exclusive-or with given data is an involution). This is similar to the one-time pad except that generated pseudorandom bits, rather than a prepared stream, are used.

To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:

1. A permutation of all 256 possible bytes (denoted "S" below).
2. Two 8-bit index-pointers (denoted "i" and "j").

The permutation is initialized with a variable length key, typically between 40 and 2048 bits, using the key-scheduling algorithm (KSA). Once this has been completed, the stream of bits is generated using the pseudo-random generation algorithm (PRGA)

© 2014-15, IJIRIS- All Rights Reserved

Key-scheduling algorithm (KSA): The key-scheduling algorithm is used to initialize the permutation in the array "S". "keylength" is defined as the number of bytes in the key and can be in the range 1 ≤ keylength ≤ 256, typically between 5 and 16, corresponding to a key length of 40 – 128 bits. First, the array "S" is initialized to the identity permutation. S is then processed for 256 iterations in a similar way to the main PRGA, but also mixes in bytes of the key at the same time.

```
fori from 0 to 255
   S[i] := i
endfor
j := 0
fori from 0 to 255
j := (j + S[i] + key[i mod keylength]) mod 256
swap values of S[i] and S[j]
endfor
```

Pseudo-random generation algorithm (PRGA)

Each element of S is swapped with another element at least once every 256 iterations.

```
i := 0
j := 0
whileGeneratingOutput:
i := (i + 1) mod 256
j := (j + S[i]) mod 256
swap values of S[i] and S[j]
K := S[(S[i] + S[j]) mod 256]
output K
endwhile
```

III. CHECKSUM ALGORITHMS

The following checksum are used.

- MD5
- SHA-1

A. MESSAGE DIGEST MD5

The MD5 message-digest algorithm is a widely used hash function producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption. It remains suitable for other non-cryptographic purposes, for example for determining the partition for a particular key in a partitioned database.

MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function MD4, and was specified in 1992 as RFC 1321. MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit words); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with 64 bits representing the length of the original message, modulo 264.

MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it. Most unix-based operating systems include MD5 sum utilities in their distribution packages; Windows users may use the included PowerShell function "Get-FileHash", install a Microsoft utility, or use third-party applications

B. SECURE HASH SH-1

SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard. SHA-1 was developed as part of the U.S. Government's Capstone project. The original specification of the algorithm was published in 1993 under the title Secure Hash Standard, FIPS PUB 180, by U.S. government standards agency NIST (National Institute of Standards and Technology.

© 2014-15, IJIRIS- All Rights Reserved

This version is now often named SHA-0. It was withdrawn by the NSA shortly after publication and was superseded by the revised version, published in 1995 in FIPS PUB 180-1 and commonly designated SHA-1. SHA-1 differs from SHA-0 only by a single bitwise rotation in the message schedule of its compression function. SHA-1 produces a message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD2, MD4 and MD5 message digest algorithms, but generates a larger hash value (160 bits vs. 128 bits).

I. Enabling CLIENT and SERVER side ENCRYPTION

To enable the encryption in Oracle sqlnet.ora is modified, appropriate value should be assigned to SQLNET.ENCRYPTION_% parameter to accept the encryption data on client side and on database server side.

The possible values for client and server side SQLNET.ENCRYPTION_% parameter

SQLNET.ENCRYPTION_CLIENT= [accepted | rejected | requested | required]
SQLNET.ENCRYPTION_SERVER= [accepted | rejected | requested | required]

accepted: The client or server will allow both encrypted and non-encrypted connections. This is the default if the parameter is not set.
rejected: the client /server rejects any connection requiring secure net traffic
requested: The client or server will request encrypted traffic if it is possible, but will accept non-encrypted traffic if encryption is not possible..the security service is activated if the other side requests or requires it
required: The client or server will only accept encrypted traffic. The client accepts the connection only if the other side accepts a secure traffic

	Rejected	Accepted	Requested	Required
Rejected	OFF	OFF	OFF	ORA-12660
Accepted	OFF	OFF	ON	ON
Requested	OFF	ON	ON	ON
Required	ORA-12660	ON	ON	ON

Figure 1. Encryption status according to client and server parameters.

II. Implementing ENCRYPTION ALGORITHM and CHECKSUM on CLIENT and SERVER side

The sqlnet.ora file of the oracle database is modified to use different encryption algorithms and checksum. Following parameters are modified.

SQLNET.ENCRYPTION_TYPES_SERVER=<encryption algorithm>
SQLNET.ENCRYPTION_TYPES_CLIENT=<encryption algorithm>
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = <crypto_checksum_algorithms>
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = <crypto_checksum_algorithms>

Different encryption algorithms and checksum algorithm are assigned to above parameter and testing is performed. Following algorithms are used.

DES: Data Encryption Standard (an old 56 bit encryption method)
3DES168: triple DES with a three-key (168 bit) option
AES128: Advanced Encryption Standard with 128-bit key
AES256: Advanced Encryption Standard with 256-bit key
RC4_128: RC4 with 128-bit key
RC4_256: RC4 with 256-bit key
MD5: Message-Digest Algorithm
SHA-1: Secure Hash Algorithm 1

© 2014-15, IJIRIS- All Rights Reserved

IV.PERFORMANCE RESULTS

The testing consists in querying 100 times the DBA_OBJECTS view (select * from dba_objects) using Oracle release 10.2. The overall elapsed time for about 220MB of data exchange on network is recorded. The tests are done on a loopback address to prevent counting real network times and on a totally idle server: tests are the only processes to run.

Each test is executed 3 times for each of the listed algorithms. The average elapsed response time from the client point of view is reported in the following table.

Algorithm	None		MD5		SHA-1	
	Time	%None	Time	%None	Time	%None
None	79.6 s		80.5 s	101%	82.4 s	104%
DES	104.7 s	132%	107.1 s	135%	108.2 s	136%
3DES168	151.8 s	191%	153.9 s	193%	155.6 s	196%
AES128	88.8 s	112%	90.5 s	114%	92.1 s	116%
AES256	91.8 s	115%	93.5 s	117%	94.2 s	118%
RC4_128	81.6 s	103%	82.5 s	104%	85.0 s	107%
RC4_256	81.7 s	103%	82.8 s	104%	85.0 s	107%

Figure 2. Encryption and Checksum performance results.

V. CONCLUSION

The work presented here is primarily about performance but it does not seem sensible to consider this in complete isolation from security since there are inevitably trade-offs between these attributes in the real world scenario. One obvious example is the number of rounds employed by each algorithm - a large number of rounds make the algorithm slower but are supposed to provide greater security [6].

The performance of a block cipher varies with the block size and the key size. The larger the block size, the faster will be the algorithm, because with a larger block size, a large chunk of the data will be encrypted in a single execution cycle of the algorithm. With a smaller block size, the same size of input data would require more execution cycles of the algorithm, thus increasing the overall encryption time other things being equal. Likewise, a larger key results in a slower algorithm, because in general, all bits of the key are involved in an execution cycle of the algorithm. With a smaller key, a lower number of key bits are involved, thus reducing the time to complete one execution cycle.

Keeping performance of application primary and ignoring strong security will not serve our purpose there are inevitably trade-offs between these attributes in the real world scenario. One obvious example is the number of rounds employed by each algorithm - a large number of rounds make the algorithm slower but are supposed to provide greater security [6].Likewise, the quality of each round in achieving the mixing that the cipher is intended to provide is also a factor that affects performance of an algorithm. Thus, there is always a trade-off between security and performance in similar algorithms - such as symmetric key algorithms. Eli Biham [7] . The RC4_128 wihtout checksum takes least time and 3DES168 algorithm with SHA-1 checksum take more time and resources to provide cyber security.

REFERENCES

[l] National Bureau of Standards - Data Encryption Standard, FIPS Publication 46,1977.
[2] NIST, "Advanced Encryption Standard Call", NIST, 1997.http://www.nist.gov/aes/
[3] NIST Advanced Encryption Standard (AES) Development Effort website http://csrs.nist.gov/encryption/aes/aes-home.html
[4] Daemen, J., Rijmen, V.: "AES Proposal: Rijndael", Banksys/KatholiekeUniversiteit Leuven, Belgium, AES submission, Jun1998.
[5] Schneier, B.: "Description of a NewVariable-Length Key, 64-Bit Block Cipher (Blowfish)", Fast Software Encryption, Cambridge Security Workshop Proceedings (Dec. 1993), Lecture Notes in Computer Science (LNCS), Springer-Verlag, Vol. 809, pp. 191-204, 1993, ISBN3-540- 58108-1.

© 2014-15, IJIRIS- All Rights Reserved

[6] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N.: "Performance Comparison of the AES Submissions", Counterpane Systems, Dec 1998. http://www.counterpane.com/AESperformance.html
[7] Biham, E.: "Design Tradeoffs of the AES Candidates", in *Asiacrypt'98*, Lecture Notes in Computer Science, Springer-Verlag,1998.
[8] "SHA-1 Collision Search Graz". Archived from the original on 2009-02-25. Retrieved 2009-06-30
[9] [RS14] Spritz---A spongy RC4-like stream cipher and hash function. Ronald L. Rivest and Jacob C. N. Schuldt
[10] Bob Anderson". Cypherpunks (Mailing list). 9 September 1994.
[11]Bob Jenkins (15 September 1994). "Re: RC4 ?". Newsgroup: sci.crypt
[12]"Manual Pages: arc4random". 5 June 2013.

© 2014-15, IJIRIS- All Rights Reserved