International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Comparative Analysis of Some Encryption Algorithms and Secured Remote Protocol

Ojekudo Nathaniel Akpofure (Ph.D)[1], Nwankwe Dimie (M.Sc)[2]

[1]Ignatius Ajuru University of Education

[2]National Open University of Nigeria

Abstract: The need for information assurance and security all over the world cannot be overemphasized. Attacks on security of a computer or network are generally characterized as interruption, interception, modification and fabrication. The four main security issues related to these are confidentiality, authentication, integrity and non-repudiation. Cryptography is necessary and must be applied on many network layers. The Research work intends to compare some encryption algorithms and secured remote protocol based on the four main security issues and also developed a prototype attached in the appendix.

Keywords: Cryptography, Encryption, Algorithm

1. Background of the Study

The vulnerabilities of IT systems have become a regular phenomenon in the press, so much that no day passes without news items or articles about some security bug or exploit. Hackers' break-in and compromise personal computers. Industrial espionage is no new economic threat (Nowell Security Enforcement, 2003) and it has costed organizations millions of dollars.
People continue to transmit private messages over unsecured telephone lines via e-mail in ASCII text, which is the least common denominator for electronic text that rely heavily on passwords, cards, personal identification numbers, and keys to access restricted information or confidential files. But these forms of identification can be forged, stolen, given away or even lost. Many systems rely on IP address verification that limits access to users with a specific domain name or Internet address. Basically, this procedure identifies an individual by the machine he or she uses. Anybody using a particular computer can impersonate the rightful owner.

Corporate information officers in all sectors must take proactive and appropriate measures to protect their networks and create better security architecture, especially now that we have entered a period of cyber security uncertainty. With no assurance regarding the security qualities or even the origin of software and systems, system owners have few components from which to construct sound security architectures. It is essential to protect the communication channels and the interfaces of any system that handles information that could be the subject of attacks. Secure Socket Layer (Allan, Philip and Paul, 1996) and Wireless Transport Layer Security (WAP Forum, 2000) is one of the few examples amongst various security protocol tools that have been proposed to address this issue. Security protocols are carefully designed to guard against loopholes. To this end, a practical Secure Socket Layer (SSL) protocol has been adopted for protection of data in transit that encompasses all network services that use TCP/IP. This of course supports typical application tasks of communication between servers and clients.

2. Related Literature

Cryptography is the study of secret writing, **Crypto (Secret) and graphy (writing).** It is the art and science encompassing the principles and methods of transforming an intelligence message into one that is unintelligible and then transforming that message back to its original form.

Cryptographic algorithms are used in encrypting an original plaintext message into a cipher text at the sender side and to decrypt the cipher text back to the original message at the receiver side. The encryption and decryption processes depend on a secret key being shared between the sender and the receiver (Marek, and Urszula, 2009).

There are three types of cryptographic algorithms: **symmetric-key algorithms, asymmetric-key algorithms, and hashing functions.**

2.1 Symmetric Key Algorithms

In Symmetric key algorithms or private key algorithms, both the sender and the receiver make use of the same key for both encryption and decryption. In a two-party communication, both parties must have access to the same key before transmission and measures must be taken to protect the secrecy of the key. The key distribution becomes increasingly more difficult when the network grows since each pair of users must exchange keys. The total number of keys in an n-person network is n (n- 1)[12.] Though, symmetric key algorithms provide strong security, it suffers from this key distribution problem. The widely adopted symmetric key algorithms include Data Encryption Standard (DES); Triple DES (3DES) (NIST FIPSPUB, 1999) and Advanced Encryption Standard (AES) (NIST FIPSPUB, 2001).

2.2 Asymmetric Key Algorithms

In Asymmetric key algorithm, each party have their own private key, which is shared with no-one, and a public key that is known to all other communicating parties. This is also called public key algorithms. When sending a

Licensed Under Creative Commons Attribution CC BY

message to a particular receiver, the receiver's public key is used to encrypt the message. After receiving the message, it is decrypted using the receivers own private key. Compared to symmetric key algorithms, asymmetric key algorithms eliminate the need to secretly distribute a key, and therefore solve the key distribution problem.

2.2.1 Hashing Functions

Unlike the other two types of cryptographic algorithms mentioned above, hashing functions do not involve the use of keys. They take a variable length string as input and convert it to a fixed length output. An example is MD5 and SHA- I (NIST FIPS-180-2, 2002).

2.3 Basic Vulnerabilities and Network Threats

It is usually difficult to say which threat is important and most dangerous from security point of view. This depends on a lot of factors, for example Denial-of-Service attack for web page of small company which does not utilize Internet as basic information carrier is not so important compared to attacks for Internet –based company. There are some common, well known basic security attacks against well known vulnerabilities or properties of network protocols. Many of them are mentioned here such as Physical security, Sniffing and Spoofing, Spanning Tree Protocol vulnerabilities, Attack against IP protocols just to mention a few.

3. Methodology

Cryptography is a very important tool in network security. It is one of the most critical and necessary element of every network infrastructure and communication, Confidentiality and data integrity are ensured by cryptographic algorithms. One of the most important groups of algorithms in cryptography is encryption algorithms. The Augusta Kerckhoff's principle (Marek R. Ogiela, 2002) state that, "every encryption algorithm shall be publicly known and only cryptographic key shall be hidden" all algorithms utilized for security purpose is in compliant with this axiom. Algorithms with symmetric keys are very popular and used in almost every encrypted connection. Such algorithms utilize only one private key shared between peers. The problem associated with this is in how to send secret key using unsecured channel? This section describes some of the most popular symmetric algorithms.

3.1 Data Encryption Standard

Created by IBM (International Business Machines Corporation) in 1976 (Gregory et al; 2009), in response to a request for proposals for a standard cryptographic algorithm from the National Bureau of Standards (NBS), now the National Institute of Standards and Technology (NIST). The original Prototype of DES algorithm was called Lucifer (Marcin K, 2012) and uses 128-bit keys. There are few legends that DES algorithm has a backdoor introduced by National Security Agency. There are few known attacks like linear or differential cryptanalysis but both may be applied to all S-box based algorithms, not only to DES.

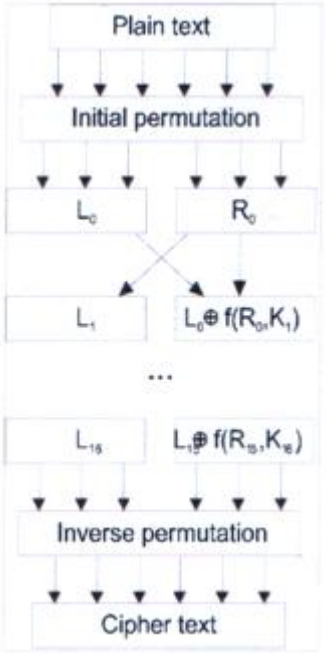


Figure 1.3: Encryption of Single Block in DES Algorithm

DES is a block cipher based on Substitution boxes (S-boxes) theory. Plain text is split into 64-bit blocks. The last block is padded to 64 bits. The 56-bit key is elongated to 64- bit with the last bit of every byte being a parity bit. Every of 64-bit blocks are encrypted as a separate data and takes eighteen actions (Figure 1.3). The initial action is an initial permutation. The block is then divided into left and right halves. Every part contains 32 bits. There are sixteen rounds, called Feistel permutations. Every round produces left and right part. K_i represents 48-bit round key derived from original key given as encryption key for algorithm. The f function has few steps described below:

i. 32-bit right part (R_i) of round data is expanded to 48 bits
ii. R is XORed with round key K
iii. XOR result is splitted into 8 parts, every part has 6 bits
iv. Every part are shortened to 4 bits after S-box substitution
v. Final permutation is performed

Round key is computed according to the following rules:

- Initial key is splitted into two parts
- Every part is shifted to the left about specified number of bits (depends on iteration number)
- Both parts are concatenated and 48 bits are selected according to p-box (permutation box)

Last operation is inverse permutation to produce cipher text. The result of DES is 64-bit block with cipher text.

$L_i = R_{i-1}$
$R_i = L_{i-1}$ XOR $f(R_{i-1}, K_i)$ 1.3

Licensed Under Creative Commons Attribution CC BY

DES algorithm has been a cryptographic standard over the years. However there are four weak keys (see Table 1.3) in DES algorithm. These keys produce the same round keys in all sixteen rounds and give cipher text the same as plain text. Moreover there are twelve semi-weak keys (Table 1.4). Semi-weak key reduces number of effective rounds and gives much better opportunity to effective cryptanalysis. There are also 48 keys that produce only four distinct round keys (instead of 16) - these are called possibly weak keys.

Table 1.3: DES four weak keys

0x	0000	0000	0000	0000
0x	0000	0000	FFFF	FFFF
0x	0EOE	0E0E	F1F1	F1F1
0x	F1F1	F1F1	0EOE	0E0E

Table 1.4: DES Half weak keys

0x	011F	011F	010E	010E	0x	1F01	1F01	0E01	0E01
0x	010E	010E	01F1	01F1	0x	E001	E001	F101	F101
0x	01FE	01FE	01FE	01FE	0x	FE01	FE01	FE01	FE01
0x	1FE0	1FE0	0EF1	0EF1	0x	E01F	E01F	F10E	F10E
0x	1FFE	1FFE	0EFE	0EFE	0x	FE1F	FE1F	FE0E	FE0E
0x	E0FE	E0FE	FIFE	FIFE	0x	FEE0	FEE0	FEF1	FEF1

3.2 Triple DES

Triple DES (or 3DES) utilises DES algorithm. It encrypts plain text three times with three different keys. The purpose of 3DES creation is to shorten the key in DES. Depending on the specific variant, 3DES uses two or three keys instead of single 56-bit key. Two keys variant is in Figure 1.4. The Plain text is encrypted two times and decrypted once in two keys variant 3DES. Decryption with a second key is performed between two encryption processes with the first key.

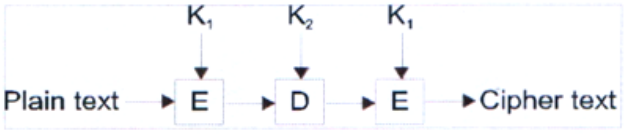


Figure 1.4: Encryption in two keys (K1 and 1(2) variant of Triple DES

This algorithm increases the number of attempts needed to retrieve the secret key which is concatenation of all used 56-bit keys. It is a significant enhancement of security. The most important issue for Triple DES is that all weak, semi weak and possible weak keys exist also in 3DES algorithm.

3.3 Substitution Box Theory

DES and 3DES uses Substitution-boxes and Feistel permutations. There is a common theory which describes S-boxes. According to Shannon theory (C.E. Shannon, 1948), the most important things are: confusion and diffusion. Moreover there is an assumption that good S-boxes shall have the following properties (we assume that x {O, 1} and a {O, 1}):

1. Strict avalanche criterion (SAC) — single input bit change should affect at least half of output bits (at least half of them should be changed). This property is applicable also to P-boxes
2. Completeness — every output bit should be a complex function of every input bit. This property is also applicable to P-boxes
3. Well balanced — binary function fr—GF (2) is well balanced if the truth table contains $2n - 1$ zeros (and ones), where GF (2) means binary Galois field.
4. Non-linearity — the minimum distance between the function and the set of all affine functions.
5. Propagation criterion of degree k — the function f holds this criterion if:
   $\forall x \in \Sigma^n, a \in \Sigma^n, a \neq 0^n$ f(x)XORf(xX0Ra): $1 \leq W(a) \leq k$
6. Good XOR profile — should not contain entries with big numbers, big numbers requires more rounds. XOR profile shows differences between S-box input and output data. Proofs and Properties of S-boxes can be found in (Pieprzyk, T. Hardjono et al; 2003), Marcin K, 2012).

3.4 Advanced Encryption Standard

Advanced Encryption Standard (AES) algorithm is much more secure than DES (Gregory white, WM; Arthur Conklin et al; 2009). The main reason is the key length (128, 192 or 256 bits). The Original algorithm supported more keys: 128-, 160-, 192, 224- and 256 bits.

The first step in AES algorithm is to divide the input, plain text to 128-bit length blocks. As shown in table 1.5, the number of rounds depends on the key's length. Data are stored in byte matrix. The height of every matrix is 4 bytes, so the key and block matrix sizes may be computed as total bits divided by 32 (4 8 bits = 32 bits).

Table 1.5: Number of round, Key and Matrix sizes as a function of key's length

	Key Matrix size [B]	Block Matrix size [B]	Number of rounds
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

There are four functions used for encryption in AES algorithm: SubBytes, ShiftRows, MixColumns and Add Round Key. Matrix with initial plain text is called the state. Previously a mentioned function operates on these states and transforms bytes inside matrix. Pseudo-code of AES algorithm is listed below:

Licensed Under Creative Commons Attribution CC BY

```
AES(input, key)
        state := input;
        AddRoundKey(state, currentKey);
        for round:=0 to N-1 begin
                SubBytes(state);
                ShiftRows(state);
                MixColumns(state);
                AddRoundKey(state, currentKey);
        end
        SubBytes(state);
        ShiftRows(state);
        AddRoundKey(state, currentKey);
        return state;
end
```

The SubBytes function operates independently on every byte given state doing nonlinear transformation according to S-box table, Table 1.6.

Table 1.6: SubBytes S-box in AES. All values are in hexadecimal system

x \ 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
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

S-box transformation is a substitution of input bytes for bytes from S-box table. The first four bits (most-significant bits) are x value and then next four bits (least-significant bit) are y value in S-box table. The Shift Rows function shifts last three row of every state in left side. The first row is not changed. The second row is one byte shifted, third row is two bytes shifted and forth row is three bytes shifted in the right. The MixColumns function works with columns of state. Four values in every column are third degree polynomial coefficients. This polynomial is multiplied by: $(03)_{16}x^3.(01)_{16}x^2.(01)_{16}x.(02)_{16}x$ mod x^4+1. The AddRoundKey transformation utilizes only XOR function. The current state is XORed with current round's key. Round's key is generated by KeyExpansion function described in [http://csrc.nist.gov/publications).

Decryption process is quite similar to encryption. Almost all helper function in this process is inverse functions to previously described helper functions. Pseudo-code of inverse AES is listed below:

```
InverseAES(input, key)
        state := input;
        AddRoundKey(state, currentKey);
        for round:=0 to N-1 begin
                SubBytes(state);
                ShiftRows(state);
                MixColumns(state);
                AddRoundKey(state, currentKey);
        end
        SubBytes(state);
        ShiftRows(state);
        AddRoundKey(state, currentKey);
        return state;
end
```

3.5 Blowfish Algorithm

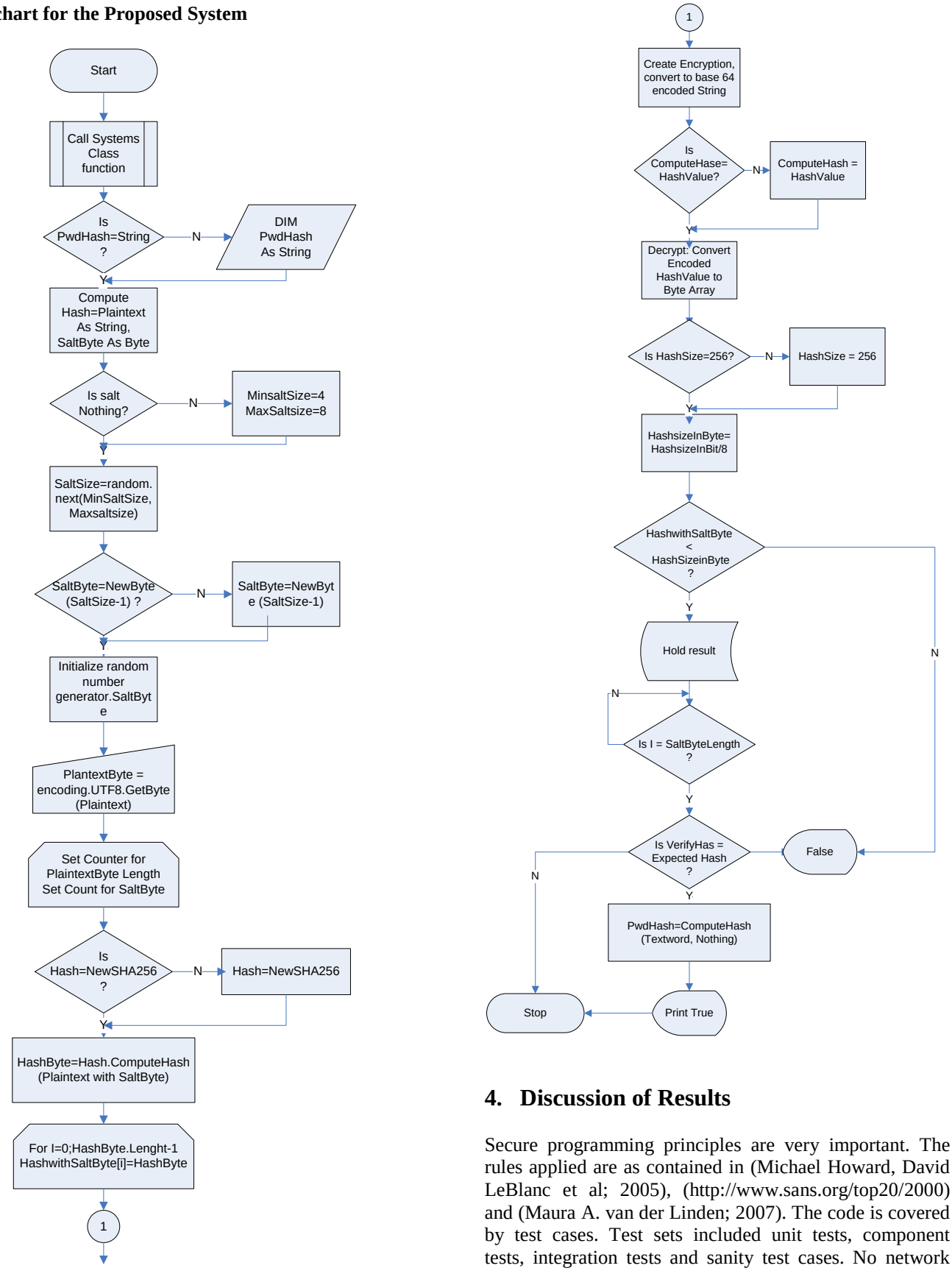
The Blowfish is a symmetric block cipher based on Feistel permutations, S-boxes and P-arrays. Designed by Bruce Schneier in 1996, it provides a good encryption rate and no effective cryptanalysis of it has been found to date. The only successful cryptanalysis against this algorithm was against variants that uses reduced number of rounds. Blowfish cipher splits input plain text into 64-bit blocks and the key length is variable from 32 to 448 bits. There are sixteen rounds in this cipher. Encryption is performed by separating 64-bit input block into two 32-bit blocks and the function is executed every round:

```
for i:=1 to 16 begin
        XLeft := XLeft XOR Pi;          // Pi is ith element from Permutation array
        XRight := F(XLeft) XOR XRight; // F() is a Feistel function

                swap(XLeft and XRight);
        end
        XRight := XRight XOR P17;
        XLeft := XLeft XOR P18;
        return XLeft || XRight;                 // concatenation
```


Licensed Under Creative Commons Attribution CC BY

Flowchart for the Proposed System



4. Discussion of Results

Secure programming principles are very important. The rules applied are as contained in (Michael Howard, David LeBlanc et al; 2005), (http://www.sans.org/top20/2000) and (Maura A. van der Linden; 2007). The code is covered by test cases. Test sets included unit tests, component tests, integration tests and sanity test cases. No network scanners were in use due to lack of attack vectors to SRP. It would be good to test such implementation by penetration tests. Such tests were not performed because all known bugs were fixed during implementation. There were also lack of pen-testers who may implement such test scenarios. This is an issue to current SRP implementation.

Licensed Under Creative Commons Attribution CC BY

Attached in the Appendix is a new SRP with the features as described below.

4.1 Security implementation guidelines for remote protocols

Systems are said to be most secure at its weakest part. Many systems have its own communication protocols to exchange messages between hosts. Cryptography is very important to make the protocol secure, however, this is not enough because there are many other dangers. It is advised that the designer of any protocol should implement secure patterns described here.

4.2 Cryptography-Related Dynamics

Clear-text protocols are not secure because everybody can listen to transmitted packets, it is better to exchange them or create new, fake packets to trigger some events in the system. Therefore cryptography is necessary. The first thing cryptographic algorithms provide is confidentiality. Messages transmitted over the network sometimes are classified and sensitive. Moreover personal data are protected in many countries and as required by law. Such data needs to be protected by encryption algorithms. Symmetric and asymmetric encryption algorithms were described earlier. Data transmitted over the network needs to be consistent. Remote protocols are used in distributed systems for instance, shopping, and financial operation or even in Supervisory Control And Data Acquisition (SCADA systems).

We cannot allow the exchange of sensitive data in clear text. Data integrity mechanism needs to be implemented by one-way hash functions, described also in third chapter. Cryptographic mechanisms shall also protect against replay attacks and such mechanisms should be build-in into the protocols. Such mechanisms may be implemented by additional numbers sent in packages and checked by destination hosts. Replay attacks are also quite dangerous as it poses a lot of danger (Shu-Wai Chow; 2007) including bringing danger in mission critical systems. Random numbers used for securing protocols should be as random as possible. Applications that use weak random number generators are more vulnerable to attack.

4.3 Authentication, Authorization and Accounting

Verification of the identity of a user or device wishing to access the application is called the authentication process. This process establishes a trust relation between the user or other service and the application. There are many kinds of authentications. If the client wants to trust the server, the server authentication is required. On the other site almost every server authenticates users who want to perform some operations. The best way of authentication is the **mutual authentication.** Secure Remote Protocol provides such mechanism. Authentication mechanisms may be divided into few types.

a) PNCODE This is something you know: passwords are something like users' identities.
b) Something you have: this includes: magnetic cards, smart cards, software and hardware tokens and many others.
c) Identity recognition: Here Identity is confirmed by something you looks like for example fingerprint or iris in the eye.
d) Identity Confirmation: checking identity examples are: handwritten signature or tone of the voice.

These methods are quite similar to third group. Both groups include biometric methods. The difference is that last group is behavior-related and third group contains static properties. Many guidelines advise to use at least two groups of authentication. It is much more difficult for the intruder, to give correct credentials if we use two or more types of authentication.

Authentication process is not enough (Michael Cross, Norris L. Johnson '2003). Every legitimate user has its own right in the system. We have to check user's right before performing any requested operation. This action is called authorization. Trusted users and devices may have permissive rights than guest users. Appropriate configuration and user management is very important to secure system. **An excessive privilege in the system allows** too many users for unnecessary actions. Access Control is of the following types:

a) First type is Mandatory Access Control (MAC). This is based on the sensitivity of the resource.
b) Discretionary Access Control (DAC). An Access Contract that assumes that the owner of the resource has the ability to change the permissions for rest of users/devices (it is the most common access Control).
c) Rule-Based Access Control (RBAC): Here access is granted or denied based on the set of predefined rules.
d) Role-Based Access Control (RBAC). The acronym is the same, but the idea is quite different. Role-Based Access Control is related to user's roles in the organization.

Every user/device has at least one role and access is granted only if the role allows for such access to specified resource. Detailed information about described access control methods is in (Gregory White, Wm. Arthur Conklin, Dwayne Williams et al; 2009).

4.4 Denial-of-Service

Prevention against Denial-of-Service (DoS) attacks is very difficult task. Localized DoS attacks may be discovered because of huge number of similar requests. In this case source of the attack may be blocked and the server may be freed from unnecessary operations. However such DoS attack is very simple and not performed very often. Almost every Intrusion Detection System (IDS) may discover and protect against such attacks. Distributed DoS is a more complicated kind of attack. There is no single source of such attack. It is very difficult to protect the system against such attack and it is almost impossible to protect the system without very good IDS and additional network connection.

Secure protocol should have limits for specified operations and such limits should be managed by the administrator. Sources of attack should be also blocked. It is very hard to

Licensed Under Creative Commons Attribution CC BY

implement such things and some part of the risk may be mitigated to appropriate firewall and IDS.

4.5 Input Validation

Most of threats listed on the top ten lists (http://www.sans.org/top20/2000) are related to insufficient input validation. The secure implementation of any protocol should validate input specified in the protocol description. Also programmers should validate the input even if the implementation of the utilized protocol is secure. The protocol's specification allows for quite big set of values, for example for any 32-bit integer value, but only small subset may be allowed. In this case it is impossible to block incorrect messages on the protocol's layer.

Wrong values in the network messages may cause information leaks, performing of unprivileged operations or crashing the program. All these threats shall be addressed in the protocol's implementation by input validation. Secure protocol cannot be vulnerable on any of the above mentioned threats. SRP implementation in Java properly addresses input validation issues and contains utils exception packet with Exceptions thrown in case of any errors on the input.

Others, such as Race conditions, where Servers which have many client connections implement many threads in single process or runs many processes with inter-process communication mechanisms (Shu-Wai Chow; 2007). **Third party software and unsafe functions where only** well-checked cryptographic modules should be used as we can't be so sure that third party software used in protocol's implementation does not contain bugs and vulnerabilities. **Security-related tests** where every project, including new protocol designing shall have special testers for security purpose. These testers shall perform both: black-box and white-box penetration tests.

5. Conclusion and Recommendations

The basis of this paper is to show that it is possible to create a new, universal network protocol designed for remote procedures, function or methods calls with the following properties:

- The protocols will have built-in security mechanisms.
- Security mechanisms will be configurable in that way that the security level will be aligned with current requirements for the system using the protocol (including clients and legal requirements).
- The protocol shall be platform and programming language-independent. In line with the following objectives:
- Propose of a new protocol specification meets the characteristics listed above.
- Sample implementation of the designed protocol.
- The analysis of the proposed protocol.
- The analysis of the mechanisms and risks in the existing network protocols for remote function and procedure calls and
- An attempt to describe the security implementation guidelines that should be used when designing new network protocols.
- Identification of further developments in the field of remote network protocols security.
- These goals were met and sample implemented. A new SRP protocol has build-in security mechanism not limited only to cryptographic algorithms. It is a flexible and can easily be configured by the administrator as recompilation is not needed.

References

[1] Allan, O.F., Philip K. & Paul C.K. (1996). The SSL Protocol Version 3.0, Internet Draft
[2] Andrew, S.T. (2003). Computer Networks (4th Edition) - Prentice Hall, New Jersey
[3] Bruce, S., (1996). Applied Cryptography, John Wiley and Sons,
[4] Gregory, W.W., Arthur C., Dwayne, W., Roger, D., & Chuck C., (2009). CompTIA Security+ Exam Guide - McGraw Hill
[5] Marcin, K. (2012). Rainbow tables as brute-force algorithm optimization (Vol 28) Krakow, Elektrotechnika in Elektronika, wyd. AGH, (pages 7-13)
[6] Marek, R.O., Urszula, O (2009). Linguistic Cryptographic Threshold Schemes,
[7] IJFGCN - International Journal of Future Generation Communication and Networking, Vol. 2, No. 1, pp. 33-40
[8] Maura, A.V. (2007). Testing Code Security - Auerbach Publications
[9] National Institute for Standard and Technology (2001). Advanced Encryption Standard
[10] (AES) FIPS PUB 197, U.S Department of Commerce, Retrieved from http://csrc.nist.gov/publication/fips/fips197/fips-197.pdf
[11] Novel Security Enhancement (2003). Preventing Industrial Espionage, Retrieved from http://www.novellgroup.com/index.php?page-espionage
[12] Ogiela, R. (2002) Advanced Image Understanding and pattern alaysis methods in Medical Imaging. Proceedings of the Fourth IASTED International Conference SIGNAL and IMAGE
[13] Shu-Wai C. (2007) - PHP Web 2.0 Mashup Projects - Packet Publishing
[14] Shannon, C. E. (1948). Communication theory of secret systems, Bell Systems Technical Journal, 27, 379, 423, 656
[15] Stanford University (2006). Network security resources and reporting problems,
[16] School of Earth Science, October 2006, Retrieved from http://pangea.standaford.edu/computerinfor/resources/network/security/
[17] Thomas, H. C, Charles E. L., & Ronald L (2001). Rivest, Clifford Stein – Introduction to Algorithms, (2nd Ed), MIT Press and McGraw-Hill
[18] Federal Information Processing Standards (2007). Announcing the Advanced Encryption Standard

Licensed Under Creative Commons Attribution CC BY

(AES) Publication 197, Retrieved from http://csrc.nist.gov/publications/fips/fips 197/fips-197.pdf
[19] NIST (2008). Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher Special Publication Retrieved from http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf - -, May 2008
[20] MIT Kerberos Consortium (2008). The Role of Kerberos in Modern Information Systems, Retrieved from http://www.kerberos.org/software/ro1ekerberos.pdf - MIT Kerberos Consortium
[21] http://www.w3.org/TR/soap/ - SOAP specifications at W3C

Appendix 1 – Source Code

```
Imports System
Imports System.Text
Imports System.Security.Cryptography

Module modHash

Public SaltedHash As String


Public Function ComputeHash(ByVal plainText As String, _
ByVal saltBytes() As Byte) _
As String
'*************************************************************************'REMOTE    SECURED
PROTOCOL (RPC) '*************************************************************************

' If salt is not specified, generate it on the fly.
If (saltBytes Is Nothing) Then

' Define min and max salt sizes.
Dim minSaltSize As Integer
Dim maxSaltSize As Integer

minSaltSize = 4
maxSaltSize = 8

' Generate a random number for the size of the salt.
Dim random As Random
random = New Random()

Dim saltSize As Integer
saltSize = random.Next(minSaltSize, maxSaltSize)

' Allocate a byte array, which will hold the salt.
saltBytes = New Byte(saltSize - 1) {}

' Initialize a random number generator.
Dim rng As RNGCryptoServiceProvider
rng = New RNGCryptoServiceProvider()

 ' Fill the salt with cryptographically strong byte values.
 rng.GetNonZeroBytes(saltBytes)
 End If

 ' Convert plain text into a byte array.
 Dim plainTextBytes As Byte()
 plainTextBytes = Encoding.UTF8.GetBytes(plainText)

 ' Allocate array, which will hold plain text and salt.
 Dim plainTextWithSaltBytes() As Byte = _
 New Byte(plainTextBytes.Length + saltBytes.Length - 1) {}

 ' Copy plain text bytes into resulting array.
```


Licensed Under Creative Commons Attribution CC BY

```
Dim I As Integer
For I = 0 To plainTextBytes.Length - 1
plainTextWithSaltBytes(I) = plainTextBytes(I)
Next I

' Append salt bytes to the resulting array.
For I = 0 To saltBytes.Length - 1
plainTextWithSaltBytes(plainTextBytes.Length + I) = saltBytes(I)
Next I

'specify hash algorithm SHA256
Dim hash As HashAlgorithm

hash = New SHA256Managed()

' Compute hash value of our plain text with appended salt.
Dim hashBytes As Byte()
hashBytes = hash.ComputeHash(plainTextWithSaltBytes)

' Create array which will hold hash and original salt bytes.
Dim hashWithSaltBytes() As Byte = _
New Byte(hashBytes.Length + _
saltBytes.Length - 1) {}

' Copy hash bytes into resulting array.
For I = 0 To hashBytes.Length - 1
hashWithSaltBytes(I) = hashBytes(I)
Next I

' Append salt bytes to the result.
For I = 0 To saltBytes.Length - 1
hashWithSaltBytes(hashBytes.Length + I) = saltBytes(I)
Next I

' Convert result into a base64-encoded string.
Dim hashValue As String
hashValue = Convert.ToBase64String(hashWithSaltBytes)

' Return the result.
ComputeHash = hashValue
End Function

Public Function VerifyHash(ByVal plainText As String, _
ByVal hashValue As String) _
As Boolean

' Convert base64-encoded hash value into a byte array.
Dim hashWithSaltBytes As Byte()
hashWithSaltBytes = Convert.FromBase64String(hashValue)

' We must know size of hash (without salt).
Dim hashSizeInBits As Integer
Dim hashSizeInBytes As Integer

' Size of hash is based on the specified algorithm.
'in this case it SHA256

hashSizeInBits = 256

' Convert size of hash from bits to bytes.
hashSizeInBytes = hashSizeInBits / 8

' Make sure that the specified hash value is long enough.
```


Licensed Under Creative Commons Attribution CC BY

```
If (hashWithSaltBytes.Length < hashSizeInBytes) Then
VerifyHash = False
End If

' Allocate array to hold original salt bytes retrieved from hash.
Dim saltBytes() As Byte = New Byte(hashWithSaltBytes.Length - _
hashSizeInBytes - 1) {}

' Copy salt from the end of the hash to the new array.
Dim I As Integer
For I = 0 To saltBytes.Length - 1
saltBytes(I) = hashWithSaltBytes(hashSizeInBytes + I)
Next I

' Compute a new hash string.
Dim expectedHashString As String
expectedHashString = ComputeHash(plainText, saltBytes)

' If the computed hash matches the specified hash,
' the plain text value must be correct.
VerifyHash = (hashValue = expectedHashString)
End Function
End Module

Public Class frmMenu

Private Sub btnHash_Click(sender As System.Object, e As System.EventArgs) Handles btnHash.Click
Dim hash As New frmHasher
hash.ShowDialog()
End Sub

Private Sub btnConfirm_Click(sender As System.Object, e As System.EventArgs) Handles btnConfirm.Click
Dim confirm As New frmConfirm
confirm.ShowDialog()
End Sub

Private Sub frmMenu_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub
End Class

Imports System
Imports System.Text
Imports System.Security.Cryptography
Public Class frmHasher

Dim pwdHash As String

Private Sub btnHash_Click(sender As System.Object, e As System.EventArgs) Handles btnHash.Click
'Create hash using SHA256 algorithm.

pwdHash = ComputeHash(txtWord.Text, Nothing)
lblHashed.Text = pwdHash
SaltedHash = pwdHash

End Sub


Private Sub frmHasher_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub
End Class
```


Licensed Under Creative Commons Attribution CC BY

```
Public Class frmConfirm

Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
 'very the hash return true or false
 'based on the SHA256 algorithm.
 Dim plainword As String
 plainword = VerifyHash(txtWord2.Text, SaltedHash)
 lblPlain.Text = plainword
 End Sub

 Private Sub frmConfirm_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

 End Sub
End Class
```


Licensed Under Creative Commons Attribution CC BY