HYBRID SECURE SOCKET LAYER PROTOCOL

Gabriela MOGOŞ*, **Gheorghe RADU****

*Escuela Superior Politecnica de Chimborazo, Riobamba, Ecuador,
** "Henri Coandă" Air Force Academy, Braşov, Romania

Abstract: *Randomness is a crucial resource for cryptography, and random number generators are therefore critical building blocks of almost all cryptographic systems. Weak random values may result in an adversary ability to break the system, as was demonstrated by breaking the Netscape implementation of Secure Socket Layer. This paper explores alternatives to traditional ciphers and involves a theoretical study with available implementations described by the quantum literature as well as classical algorithms. This paper provides an original implementation of Secure Socket Layer that includes together Quantum Random Number Generator (QRNG) and Elliptical Curve Cryptography (ECC) in a hybrid cryptographic protocol.*

Keywords: *asymmetric cryptosystems, quantum random numbers, Secure Socket Layer (SSL).*
***MSC2010*:** 81P45, 94A15.

1. INTRODUCTION

In the last ten years, the Internet has enjoyed tremendous success connecting a large number of households and businesses with each other. This has created enormous economic possibilities. However, this economic potential can only be fully realized if the need for secure transmission of information over the inherently insecure and open Internet can be satisfied. Cryptography addresses this need.

One of the fundamentally important research areas involved in quantum information science is quantum communications, which deals with the exchange of information encoded in quantum states of matter or quantum bits (known as qubits) between both nearby and distant quantum systems.

Our paper is related to quantum communication and performs core research on the use of optical qubits – the quantum states of photons, with particular attention to application to future information technologies.

Our paper presents a theoretical demonstration of how to improve the security of Secure Socket Layer protocol by replacing Random Number Generator with Quantum Random Number Generator and the RSA cryptographic algorithm, currently used, with ECC algorithm.

For the beginning, the paper will present *Random Number Generator* (RNG) and *Quantum Random Number Generator* (QRNG), and, advantages of using QRNG in generating a sequence of random bits that are impossible to "guess" by the attackers.

Seeing that, the Quantum Random Number Generator (QRNG) is ideal for applications requiring very high rates of true random numbers, our paper present the importance of replacing the classic Random Number Generator (RNG) used by Secure Socket Layer (SSL) protocols, with quantum one (QRNG).

The reason for replacing the *RSA cryptosystem* (existing in SSL 3.0) with *ECC cryptosystem* will be presented in the second part of the paper through a comparative analysis of those cryptosystems. The comparative analysis is to demonstrate that we can improve the security using less resources, and a short time encryption / decryption.

2. HYBRID SECURE SOCKET LAYER PROTOCOL

2.1 Random Number Generators vs. Quantum Random Number Generators.

Any security application must employ both hardware and software and not rely solely upon software.

Software is just an application, and therefore it can be broken.

The specialists in the computer security field have been migrating to hardware solutions in order to supply the secure systems that protect their information and reduce risks due to loss, theft, or hacking. These solutions introduce a variety of approaches that implement the advantages offered by hardware-based security.

Random Numbers are a cryptographic primitive and cornerstone to nearly all cryptographic systems. They are used in almost all areas of cryptography, from key agreement and transport to session keys for encryption.

Imperfections in Random Number Generators (RNG) [4, 6, 2, 7] can introduce patterns undetected by statistical tests but known to an adversary.

Random number generators based on classical physics are fundamentally deterministic – as is classical physics – even if the complexity of the system can hide the determinism.

A powerful alternative to the Random Number Generator is a hardware-based random-number generator. This is a device residing on a local machine or a stand-alone unit that includes electronics that produce random numbers.

Random number generators based on quantum physics are true random number generators as quantum physical phenomena are intrinsically random.

With this last type of generators, called *Quantum Random Number Generators* (QRNGs), it provides a source of random numbers suitable for applications with the most stringent security standard.

A team of experimentalists from the Joint Quantum Institute (JQI), in partnership with European quantum information scientists, has demonstrated a method of producing a certifiably random string of numbers based on fundamental principles of quantum mechanics.

They report their results in the 15 April 2010 issue of *Nature* [12].

"Classical physics [11] simply does not permit genuine randomness in the strict sense," says JQI Fellow Chris Monroe, who led the experimental team. "That is, the outcome of any classical physical process can ultimately be determined with enough information about initial conditions. Only quantum processes can be truly random - and even then, we must trust that the device is indeed quantum and has no remnant of classical physics in it."

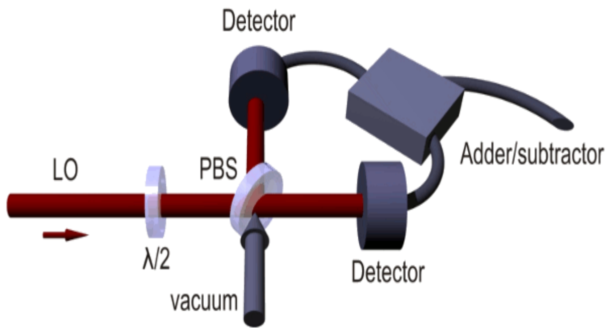
Quantum cryptography, presents a recent addition to the family of device-based random-number generators.

Some U.S. and European companies offer second or third generation quantum cryptography products, and many of the top academic, government, and commercial research institutions have development efforts underway.

In the opinion of top researchers at the National Security Agency [16], quantum cryptography appears to be unbreakable. The use of techniques such as polarized or entangled photons provides assurance against eavesdropping that is unmatched by other technologies.

Due to the apparent fundamental strength of quantum communication and cryptography, future attacks on the technology will likely be limited to brute force attacks by powerful computers (such as quantum computers) or perhaps physical threats to authorized parties.

Based on these considerations, a simple design for a Quantum Random Number Generator is based on a construction with a photon source, a 50/50 beam-splitter and two identical single photon detectors.



This construction ensures a random and balanced probability of detection of the photon at each detector. However, a drawback is that it can only produce one raw bit at a time. In order to increase the bit rate of the Quantum Random Number Generator, a new design has been considered. Although random numbers are required in many applications, their generation is often overlooked. Being deterministic, computers are not capable of producing random numbers. A physical source of randomness is necessary. Quantum physics being intrinsically random, it is natural to exploit a quantum process for such a source.

Quantum random number generators (QRNG) have the advantage over conventional randomness sources of being invulnerable to environmental perturbations and of allowing live status verification.

Quantum random number generators (QRNG) can generate truly random numbers from the fundamentally probabilistic nature of quantum processes.

Quantum Random Number Generator is

a fast, nondeterministic and novel random number generator whose randomness relies on intrinsic randomness of the quantum physical process of photonic emission in semiconductors and subsequent detection by the photo-electric effect.

The statistical tests [15] applied to random numbers sequences longer than 1 Gb produced with this quantum random number generator presents results which demonstrate the high quality of randomness resulting in bias less than 10^{-4}, autocorrelation consistent with zero, near maximal binary entropy and measured min-entropy near theoretical maximum.

2.2 Elliptical Curve Cryptography (ECC) vs. Rivest, Shamir, Adleman (RSA) cryptosystem. Every Secure Socket Layer connection begins with a handshake, during which the two parties communicate their capabilities to the other side, perform authentication, and agree on their session keys. The session keys are then used to encrypt the rest of the conversation, possibly spanning multiple connections. They are deleted afterwards. The goal of the key exchange phase is to enable the two parties to negotiate the keys securely, to prevent anyone else from learning these keys.

Several key exchange mechanisms exist, but, at the moment, by far the most commonly used one is based on RSA, where the server's private key is used to protect the session keys.

Building on the Diffie and Hellman "Public-key Cryptosystem," Ron Rivest, Adi Shamir, and Leonard Adleman created the *RSA* cipher in 1978 [9]. Today, the RSA cipher is the most common form of public-key cryptology in use.

Elliptic Curve Cryptography relies on the difficulty of solving the discrete logarithm problem as the basis of its security [5].

Elliptic Curve Cryptography is a quite newer version of Public-Key Cryptology and can provide the same level of security as RSA, but with smaller key sizes. With all variables being equal, Elliptic Curve Cryptography can run more transactions per second than RSA.

We used in our study these results [1] from the tables below:

Table1. Comparison of RSA (512 bits) and ECC(106 bits)

	Key Generation Time	Memory Requirement	**Encrypt/ Decrypt Time**
ECC (106 bits)	**57 ms**	108 bytes	**11 ms**
RSA (512 bits)	**383 ms**	157bytes	**77 ms**

Key Generation Time (RSA/ECC) = 6,72
Encrypt/Decrypt Time (RSA/ ECC) = 7
ECC (106 bits)/RSA (512 bits) = 4,83

Table 2. Comparison of RSA(768 bits) and ECC(132 bits)

	Key Generation Time	Memory Requirement	**Encrypt/ Decrypt Time**
ECC (132 bits)	**98 ms**	117 bytes	**17 ms**
RSA (768 bits)	**889 ms**	236 bytes	**160 ms**

Key Generation Time (RSA/ECC) = 9,07
Encrypt/Decrypt Time (RSA/ ECC) = 9,41
ECC (132 bits)/RSA (768 bits) = 5,82

Table 3. Comparison of RSA (1024 bits) and ECC(160 bits)

	Key Generation Time	Memory Requirement	**Encrypt/ Decrypt Time**
ECC (160	**108 ms**	125 bytes	**16 ms**
RSA (1024bits)	**2609 ms**	313 bytes	**388 ms**

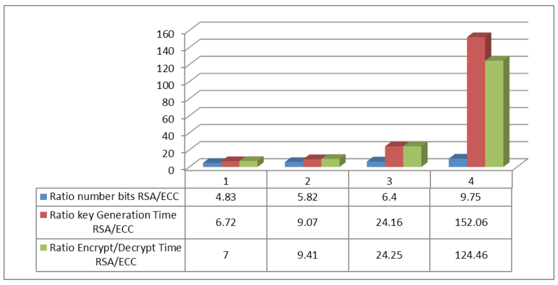
Key Generation Time (RSA/ECC) = 24,16
Encrypt/Decrypt Time (RSA/ ECC) = 24,25
ECC (160 bits)/RSA (1024 bits) = 6,4

Table 4. Comparison of RSA(2048 bits) and ECC(210 bits)

	Key Generation Time	Memory Requirement	**Encrypt/ Decrypt Time**
ECC (210bits)	**121 ms**	140 bytes	**15 ms**
RSA (2048bits)	**18399 ms**	621 bytes	**1867 ms**

Key Generation Time (RSA/ECC) = 152,06
Encrypt/Decrypt Time (RSA/ ECC) = 124,46
ECC (210 bits)/RSA (2048 bits) = 9,75

Making a comparative analysis of the two cryptographic algorithms, are three different characteristics [1] namely performance, security and space requirements.



	1	2	3	4
🟦 Ratio number bits RSA/ECC	4.83	5.82	6.4	9.75
🟥 Ratio key Generation Time RSA/ECC	6.72	9.07	24.16	152.06
🟩 Ratio Encrypt/Decrypt Time RSA/ECC	7	9.41	24.25	124.46

Figure 2. Comparative analysis RSA/ECC

Analysis conclusions:

➢ *In RSA case*, a doubling of the number of bits causes an increase of six times of *Key generation time*.

➢ *In RSA case*, a doubling of the number of bits causes an increase of 24 times of *Encrypt/Decrypt time*.

➢ *In ECC case*, the *Encryption/Decryption time* remains relatively constant even if the key size increases.

➢ *In ECC case*, a doubling of the *Number of bits* of the key causes a doubling of Key generation time.

2.3 Secure Socket Layer a hybrid protocol. The Secure Sockets Layer [13] is a standard protocol for encrypting Internet traffic. It is very mature and has been widely implemented and tested for vulnerabilities. As long as no one figures out how to factor large prime numbers in a hurry, the Secure Sockets Layer appears to be in good shape to provide security.

The Secure Sockets Layer protocol defines two different roles for the communicating parties. One system is always a client, while the other is a server.

The most basic function that a Secure Sockets Layer client and server can perform is establishing a channel for encrypted communications.

The cryptographic parameters of the session state are produced by the *Secure Socket Layer Handshake Protocol*. When a Secure Sockets Layer client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets.

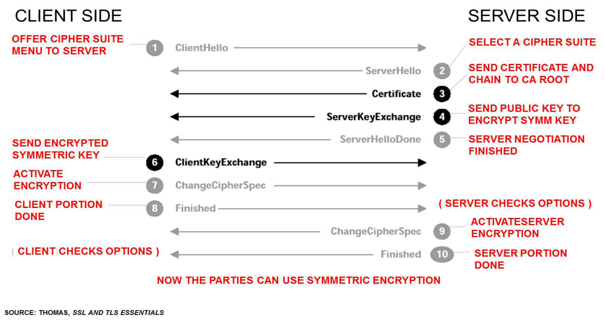

CLIENT SIDE
SERVER SIDE
OFFER CIPHER SUITE MENU TO SERVER
1 ClientHello
SELECT A CIPHER SUITE
ServerHello 2
SEND CERTIFICATE AND CHAIN TO CA ROOT
Certificate 3
ServerKeyExchange 4
SEND PUBLIC KEY TO ENCRYPT SYMM KEY
ServerHelloDone 5
SERVER NEGOTIATION FINISHED
SEND ENCRYPTED SYMMETRIC KEY
6 ClientKeyExchange
ACTIVATE ENCRYPTION
7 ChangeCipherSpec
CLIENT PORTION DONE
8 Finished
(SERVER CHECKS OPTIONS)
ChangeCipherSpec 9
ACTIVATESERVER ENCRYPTION
(CLIENT CHECKS OPTIONS)
Finished 10
SERVER PORTION DONE
NOW THE PARTIES CAN USE SYMMETRIC ENCRYPTION
SOURCE: THOMAS, SSL AND TLS ESSENTIALS

Figure 3. SSL Messages

Two random values are generated and exchanged: *ClientHello.random* and *ServerHello.random*.

These *random numbers* will have a very important role in the Secure Socket Layer protocol. They will be used to obtain:

- Master secret -generated by both parties from premaster secret and random values generated by both client and server.

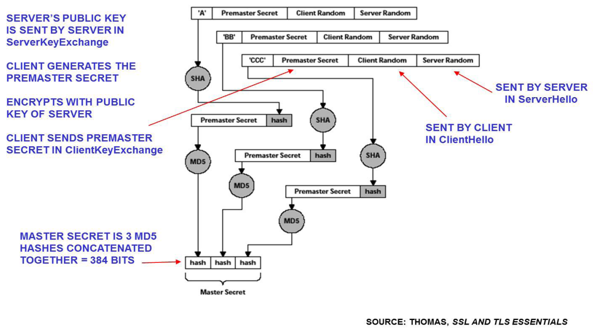

SERVER'S PUBLIC KEY IS SENT BY SERVER IN ServerKeyExchange
CLIENT GENERATES THE PREMASTER SECRET
ENCRYPTS WITH PUBLIC KEY OF SERVER
CLIENT SENDS PREMASTER SECRET IN ClientKeyExchange
'A' Premaster Secret Client Random Server Random
'BB' Premaster Secret Client Random Server Random
'CCC' Premaster Secret Client Random Server Random
SENT BY SERVER IN ServerHello
SENT BY CLIENT IN ClientHello
SHA
MD5
Premaster Secret hash
MASTER SECRET IS 3 MD5 HASHES CONCATENATED TOGETHER = 384 BITS
hash hash hash
Master Secret
SOURCE: THOMAS, SSL AND TLS ESSENTIALS

Figure 4. Generating the Master Secret

- Key material - generated from the master secret and shared random values, and, finally, the encryption keys.

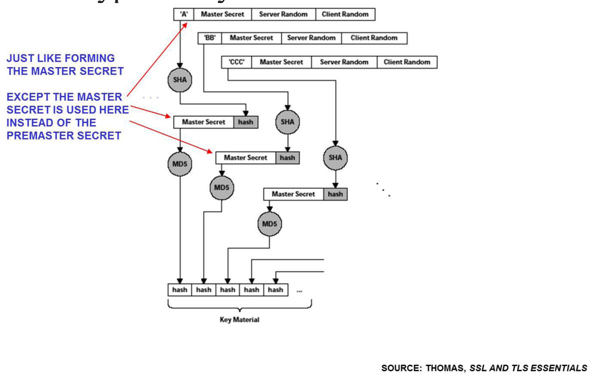

'A' Master Secret Server Random Client Random
'BB' Master Secret Server Random Client Random
'CCC' Master Secret Server Random Client Random
JUST LIKE FORMING THE MASTER SECRET
EXCEPT THE MASTER SECRET IS USED HERE INSTEAD OF THE PREMASTER SECRET
SHA
MD5
Master Secret hash
hash hash hash hash hash ...
Key Material
SOURCE: THOMAS, SSL AND TLS ESSENTIALS

Figure 5. Generation of Key Material

Random numbers are critical to the operation of the Secure Sockets Layer protocol. The random numbers exchanged in *ClientHello* and *ServerHello* messages ultimately determine the encryption key for the session. Random numbers, however, present an interesting challenge to computer systems; software cannot do anything truly randomly.

Instead, software implementations typically rely on algorithms known as *pseudorandom number generators*. These algorithms simulate true randomness with complex mathematical calculations.

Press, Teukolsky, Vetterling, and Flannery [8] report on one widely used pseudorandom number generator that, in an extreme case, effectively generated only 11 distinct random values**.**

If you know the parameters of the algorithm and one specific value, it is easy to predict all future values that the algorithm will generate.

Predictable random numbers are a serious problem for any security protocol, as they allow attackers to plan and prepare well into the future, waiting, perhaps, for a single, compromised value to appear.

Starting from the above findings (theoretical and practical), the proposed hybrid Secure Socket Layer protocol hybrid will use a Quantum Random Generator and an ECC algorithm.

Why to replace Random Number Generator with Quantum Random Number Generator?

Replacing the Random Number Generator (RNG) with quantum one (QRNG), we can get an increase of the number of bits that should be a “cryptographically secure” random number. Therefore, increasing the number of bits used in the encryption determines a high security of Secure Sockets Layer protocol. In the following, we will show how to connect Quantum Random Number Generator to a computer.

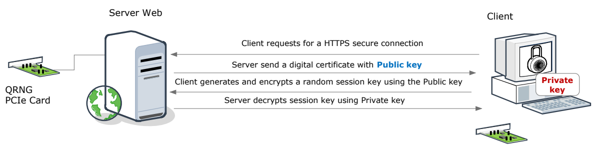

Server Web
Client
QRNG
PCIe Card
Client requests for a HTTPS secure connection
Server send a digital certificate with Public key
Client generates and encrypts a random session key using the Public key
Server decrypts session key using Private key
Private key

Figure 7. Connect a Quantum Random Number Generator to a computer

The Quantum Random Number Generator device connects to a computer via PCI or USB interface. This enables connecting several Quantum Random Number Generator devices and achieving random numbers acquisition speeds much higher than those of a single device. Quantum Random Number Generator hardware access is implemented as a dynamic link library that communicates directly with the Quantum Random Number Generator device driver.

Thereby, replacing the classic method with quantum one, the obtained key is very safe and we can choose as its dimension to be much higher than that obtained by the classical method.

Since the data will be encrypted using asymmetric cryptographic algorithms, key size is essential in ensuring the security and integrity of encrypted information.

Security is not something we ordinarily associate with randomness, but it is important in this case.

Most computer programs use a technique known as pseudorandom number generation to create random numbers. When used correctly, this approach does yield numbers that have the appearance of randomness.

Generating large numbers using Quantum Random Number Generator, lead to a "*chain reaction*" of improving the security protocol Secure Socket Layer.

We conclude these:

The requirement that a Quantum Random Number Generator be physically manufactured offers a variety of opportunities for the designers to design security features and self-consistency checks directly into the device.

The use of these techniques can virtually eliminate any security concerns regarding theft, spoofing, or even counterfeiting.

The initial condition issues that dominate the software-based Random Number Generator number generators are more manageable for hardware-based Quantum Random Number Generators, since such number generators cannot be forced into a compromising state. Quantum Random Number Generators eliminate most of the security concerns relating to improper handling of the device after receipt, since, unlike Random Number Generators, the hardware generators are not susceptible to software and operating-system attacks.

Why to use Elliptic Curve Cryptography (ECC) on Secure Socket Layer?

- System *Secure Socket Layer* has been updated to support 20 new ECC cipher suites.

- ECC is an emerging public-key crypto-system that *offers equivalent security with smaller keys sizes*.

- Augments end to end encryption for data in flight by helping to maintain data privacy and prevent data leakage of sensitive information particularly when providing the next generation of security level requirements.

- We can *reduce the transmission cost during handshake*.

Replacing RSA cryptographic algorithm with ECC algorithm, we can get the same time for encryption/decryption regardless of cryptographic key size, and, the number of bits of the encryption/decryption key determining a directly proportional increase of the key generation time.

CONCLUSIONS & ACKNOWLEDGMENT

Hardware solutions offer advantages that are unavailable to software-only implementations of security. Software based number generation suffers from the predictability inherent in algorithmically based Random Number Generators. The Quantum Random Number Generator offers the best option for strong security in the long term, especially if the technologies are based on the use of algorithms.

Photons and entangled photons of the sort used in quantum cryptography and communication have a form of tamper evidence and protection built into the technology at the level of quantum-mechanical processes. At the moment, it is widely believed that this technology cannot be defeated.

This theoretical study is part of a larger project that we be will developed by *Facultad de Informatica y Electronica, Riobamba, Ecuador*. The entire theoretical study will be sustained with an experimental hybrid protocol.

This work was financed from Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) of Ecuador, through Prometeo project.

BIBLIOGRAPHY

1. Kancheti S., *Comparative Study of Elliptic Curve Cryptography and RSA in Constrained Environment*, (2010).
2. Lofberg J., *Yalmip: A toolbox for modeling and optimization in MATLAB.*
3. http://www.idquantique.com .
4. Navascues M., Pironio S., and Acin A.. *Bounding the set of quantum correlations*, Physical Review Letters, 98:010401, (2007).
5. NIST, Recommendation for Key Management – Part 1: General (Revised). *National Institute of Standards and Technology.* NIST Special Publication 800-57, (2007).
6. Pironio S., Navascues M., Acin A., *Convergent relaxations of polynomial optimization problems with non-commuting variables*, arXiv:0903.4368, (2009).
7. Pironio S., Acin A., Brunner N., Gisin M., Massar S., and Scarani V., *Device-independent quantum key distribution secure against collective attacks*, New Journal of Physics, 11:045021, (2009).
8. In the second edition of *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press, 1992), p. 277. Chapter 7 includes a thorough (and sobering) discussion of random number generation.
9. Rivest R.L., Shamir A., & Adleman, L., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystem*, Retrieved from http://people.csail.mit.edu/rivest/Rsapaper.pdf (1978).
10. Sturm J.F., SeDuMi, *a MATLAB toolbox for optimization over symmetric cones*, http://sedumi.mcmaster.ca .
11. http://jqi.umd.edu/sites/default/files/newsletters/may_2010_newsletter.pdf .
12. Pironio S., Acín A., Massar S., Boyer A. de la Giroday, Matsukevich D. N., Maunz P., Olmschenk S., Hayes D., Luo L., Manning T. A. & Monroe C., *Random numbers certified by Bell's theorem*, Nature, (2010); 464 (7291): 1021 DOI: 10.1038/nature09008.
13. Thomas S., *Networking – SSL and TLS Essentials*, Wiley, (2001).
14. http://cms.unige.ch/gap/optics/wiki/research:quantum_cryptography:quantum_random_number_generators
15.Stipcevic, M., *Quantum Random Bit Generator (QRBG)*, http://qrbg.irb.hr .
16. Chapter 25 http://www.cse.iitk.ac.in/users/anuag/crypto.pdf.