BENCHMARKING FOR THE QP CRYPTOGRAPHIC SUITE

This work has been realized by the research team led by Prof. Giuseppe F. Italiano at the Dipartimento di Informatica, Sistemi e Produzione, Università di Roma “Tor Vergata”. The team is composed by Vittorio Ottaviani ,Antonio Grillo and Alessandro Lentini.

SUMMARY

INTRODUCTION

In everyday life of European, US, Australian, and part of Asian and African cities (see Figure 1) everybody holds a mobile device in his/her hands during the day for business or private reasons. The penetration rate is rapidly growing every year.

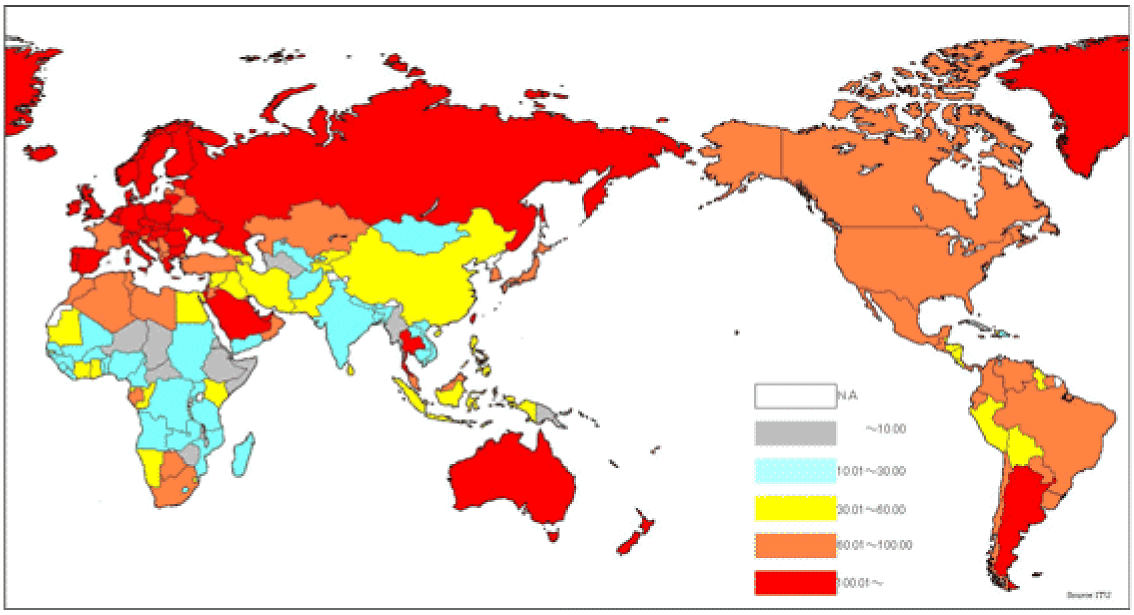

Figure 1: 2007 world mobile device penetration [7].

This growing interest in mobility suggests that there will be the need of new more powerful devices and, even more, technologies that allow people to use their mobile devices for working and personal purpose, allowing to access the internet, or other services granting him/her a good level of security, privacy and usability.

Offering security and privacy to users, at an acceptable level of usability, should allow a better diffusion of new mobile services such as: mobile banking, home automation (domotics), mobile internet, etc…

In order to ensure trust, security and privacy, there is a strong need for robust cryptographic techniques comparable to those used in a desktop environment. However, these techniques do not appear to be fully applicable in mobile devices, because cryptographic techniques are computationally expensive and the resources currently available on mobile devices are simply not enough to support sophisticated cryptographic algorithms.

The resource greediness of the currently available cryptographic algorithms mainly affects the usability of the systems developed basing the security on standard algorithm. This motivates the need of new algorithms capable of offering the same level of security as standard cryptographic algorithms, but with less usage of resources.

The QP cryptographic suite is composed of two algorithms that allow the user to perform a complete communication process made by a key exchange phase (over an unsecure channel,) and a communication over an encrypted channel.

The two algorithms are:

1. **QP-CONJ:** used for the key exchange phase of the communication;
2. **QP-DYN:** used for the encryption/decryption phase of the communication.

The security of QP-DYN's has been statistically tested and the results are available in the section named Statistically testing *QP - Dyn* and *RC4*. These results do not prove that QP-DYN is unbreakable; however they show that QP-DYN not only satisfies NIST requirements for classified information but also it passes tighter and more robust tests, such as Rabbit, Alphabit Pseudodiehard, FIPS-140-2 and Crush test batteries.

Since in a mobile environment the response time of an application using cryptographic algorithms is of the utmost important for usability requirements, we evaluated the performance of QP-DYN and QP-CONJ and compared it against standard algorithms performing similar tasks.

TESTING PERFORMANCES OF CRYPTOGRAPHIC ALGORITHM IN MOBILE ENVIRONMENT

We report here our performance tests of QP on a Nokia N70 platform. The NOKIA N70 is a multimedia smartphone launched in Q3 2005. In 2007, it was the second most popular cellular phone, with 8% of all sales at Rampal Cellular Stockmarket[1]. Our experiments show similar results with other mobile devices.

NOKIA N70 is equipped with:

- **CPU:** TI OMAP 1710 220 MHz processor,
- **OS:** Symbian OS 8.1a , Series 60 UI operating system,

Java: MIDP 2.0 midlets.

[1] "RCS Announces 2007 January-June Trading Data for the Global Cellular Phone Open Market", retrieved on 2008-09-23.

FULL PROCESS ANALYSIS

The objective of using key exchange and encryption/decryption algorithms is to develop applications that allow the user to exchange information privately. This means that there will be a key exchange phase, a key agreement phase, an encryption phase, an information exchange phase and a decryption phase.

We will call full process the sequence of operations composed by a sequence of basic operations: key generation, key agreement and encryption/decryption for a selection of the best performing configurations. For some processes: DH and AES, ECDH and AES, QP-CONJ and QP-DYN there is the need to perform the key agreement phase, while for ECIES and RSA this phase is not required.

In Figure 2 we show the times required from some relevant tested solutions to perform the full process. We can see that the difference in times required by RSA 1024 to perform a full process compared with every QP performing the same operation are really important:

- 35 milliseconds for QP;
- 35 seconds for RSA.

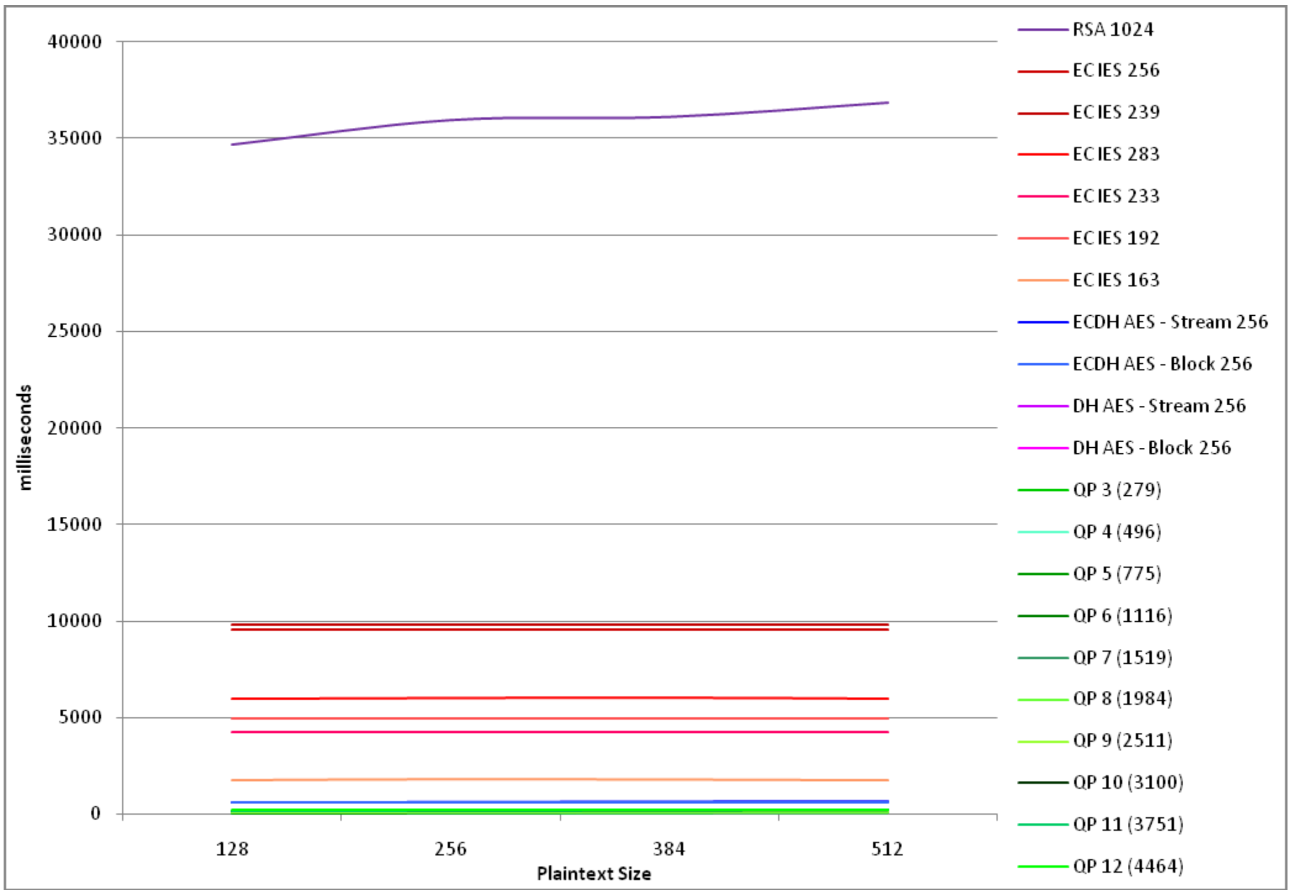

Figure 2: Timing for key operations, encryption and decryption process for all the solutions tested in normal scale

In Figure 3 timing for the same solutions shown in Figure 2 are shown in this case the scale is different, we choose to use the logarithmic scale to better illustrate the differences between the different solutions tested.

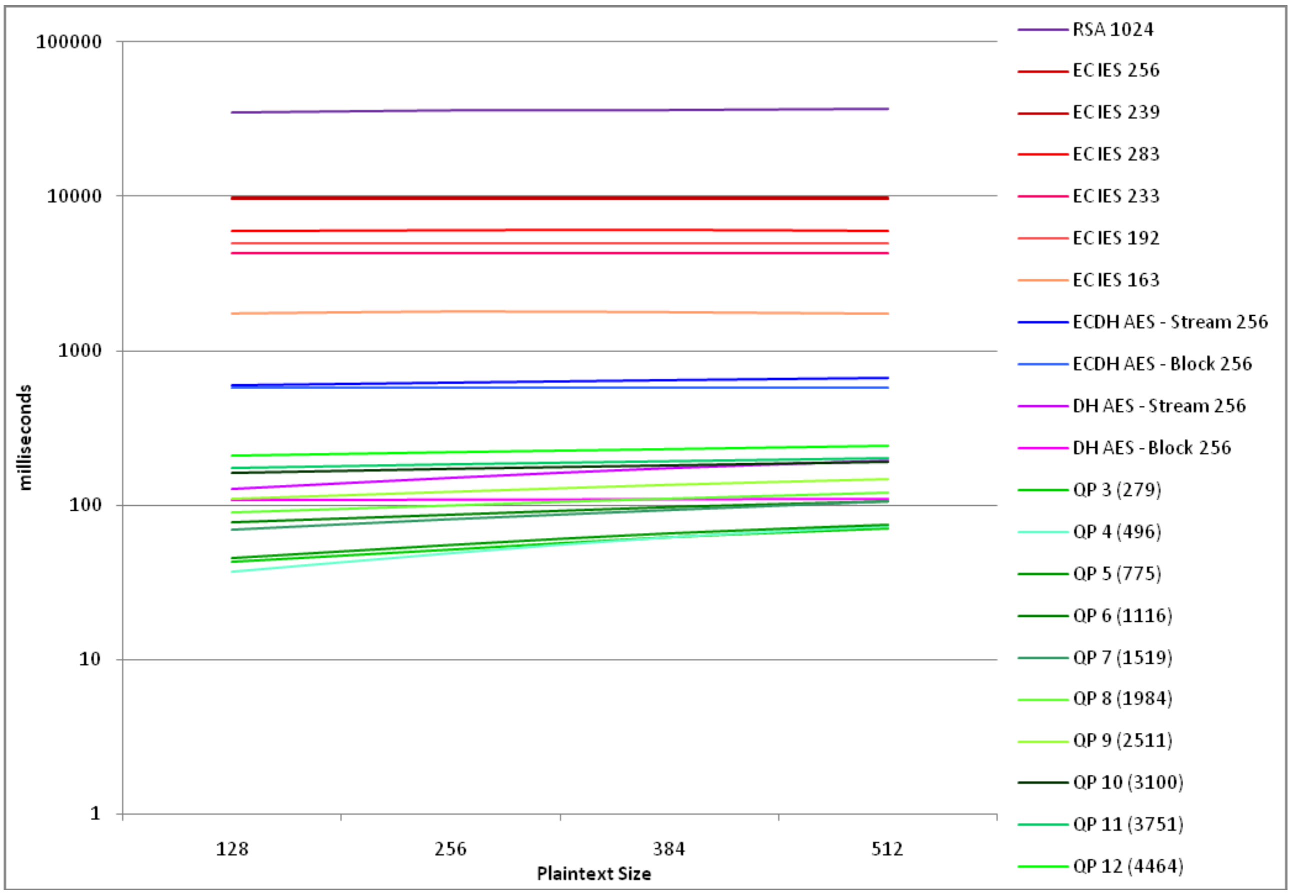

Figure 3: Timing for key operations, encryption and decryption process for all the solutions tested in log scale

In Figure 4 the timing of the full process is shown for just some of the solutions tested. Note that in Figure 4 time is represented in log-scale.

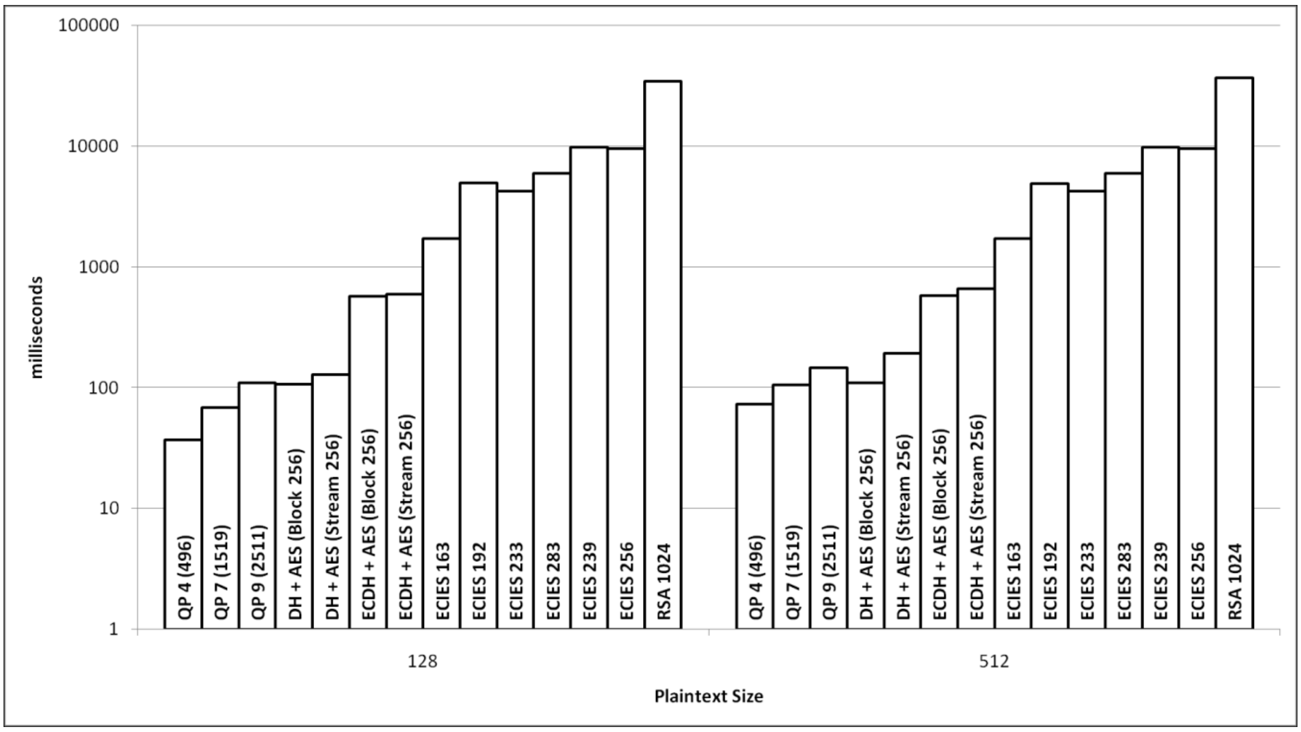

Figure 4: Comparison among all algorithms considered in mobile environment

For usability reasons, in a mobile environment, a time loss of 2 or 3 seconds spent in cryptographic operations may be considered to be too "expensive". For this reason, we present in Figure 5 only the algorithms that run in less than 2 seconds. As it can be easily seen, QP seems to be the fastest algorithm, even if it uses larger keys.

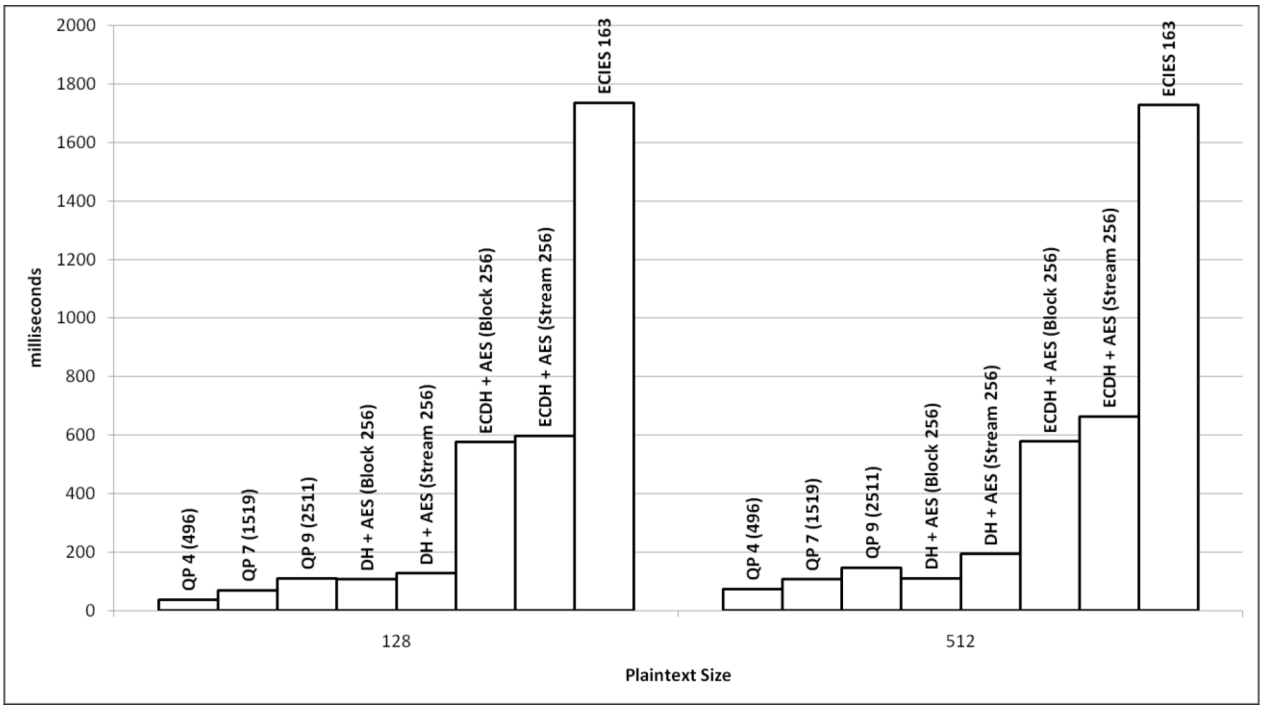

Figure 5: Comparison among usable algorithms considered in mobile environment

KEY EXCHANGE PROCESS TESTING

In this section we report the analysis performed just on the key management phases, here we show the results of all the tests performed on common key agreement algorithm.

We compared the performance of QP-CONJ to other reference algorithms, such as Diffie-Hellmann key exchange [8], Elliptic Curve Diffie-Hellman key exchange [15]. We remark that these algorithms are the most used to perform key exchange operations in desktop and mobile environments. Among the NIST suggested Elliptic Curves [6] we select both Koblitz curves (ending with a "*k*" in Figure 6) and pseudo-random curves over GF(p).

In Figure 6 it is shown the time comparison between QP-CONJ, Diffie-Hellmann and Elliptic Curve Diffie-Hellmann. We can note that QP-CONJ generates a key and a key agreement faster than the others algorithms. Since in Figure 6 the difference in generation times for the secret and the key agreement is not really appreciable, we illustrate in Figure 7 a closer look to show better the differences in time.

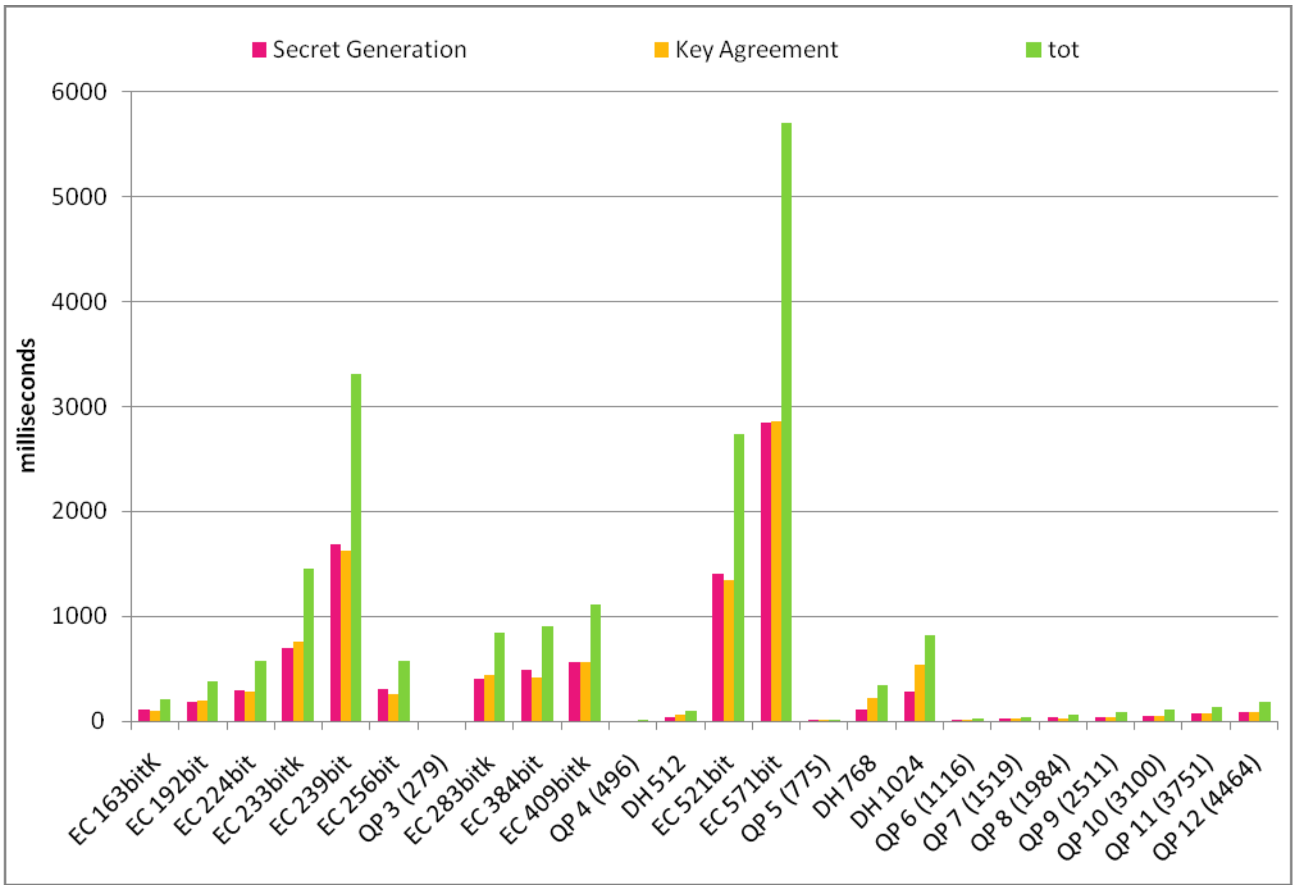

Figure 6: Key and Key Agreement generation time comparison

While a key exchange using Elliptic Curve with a 571-bit key takes 5706.3 milliseconds, a key exchange using QP-CONJ with a 5x5 matrix (775-bit key) takes only 20.63 milliseconds. This difference is significant even considering that the key generated by QP-CONJ is 50% larger than the Elliptic Curve key.

Even when considering the case of standard Diffie-Hellman, the differences in mobile environment look quite impressive; for example, a Diffie-Hellmann 768-bit key is generated and agreed in 343.44 milliseconds while a QP-CONJ 775-bit key takes only 20.63 milliseconds. These differences are illustrated in Figure 7.

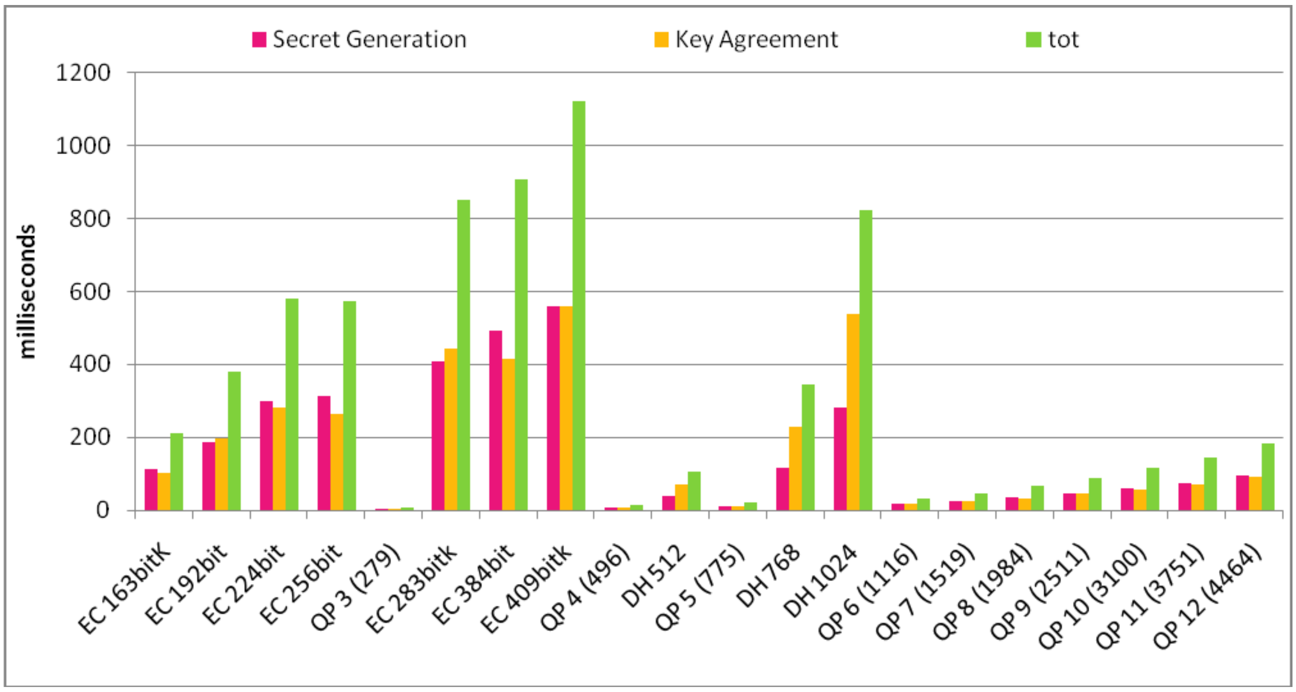

Figure 7: Key and Key Agreement generation time comparison setting an upper bound to 1 sec.

ENCRYPTION/DECRYPTION PROCESS TESTING

In this section we show the results of the performance of QP-DYN in a mobile environment compared to standard algorithms performing similar operations. We compare QP-DYN with RC4 and AES CFB Stream Cipher [9] [11] because both perform stream cipher operations as QP-DYN.

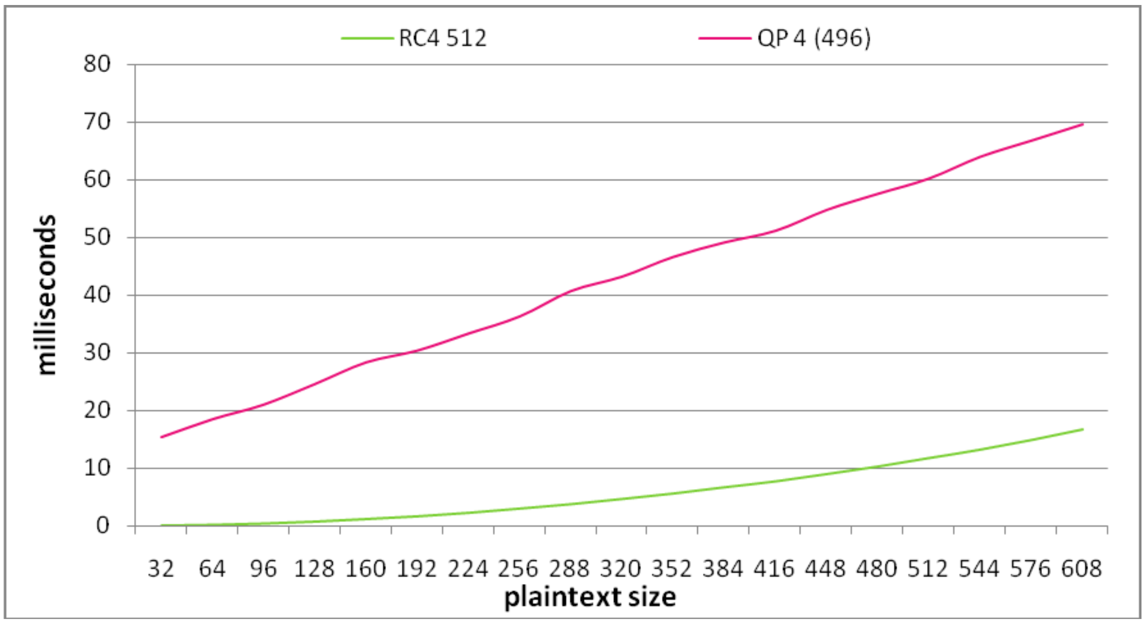

Figure 8: Overall encryption and decryption time comparison between RC4 512-bit and QP4

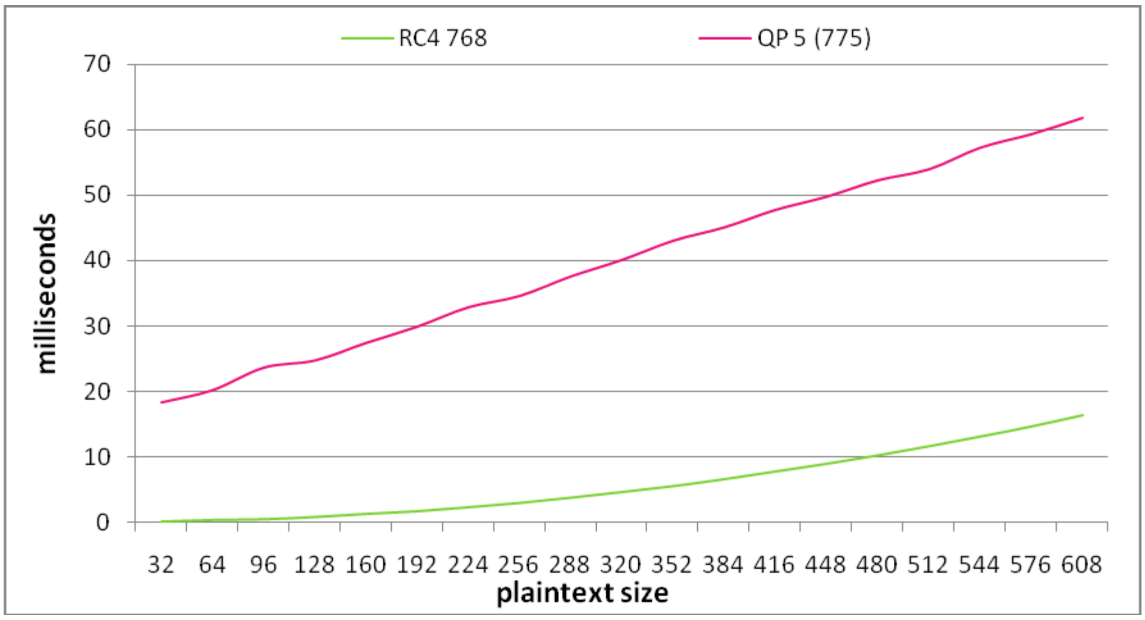

Figure 9: Overall encryption and decryption time comparison between RC4 768-bit and QP5

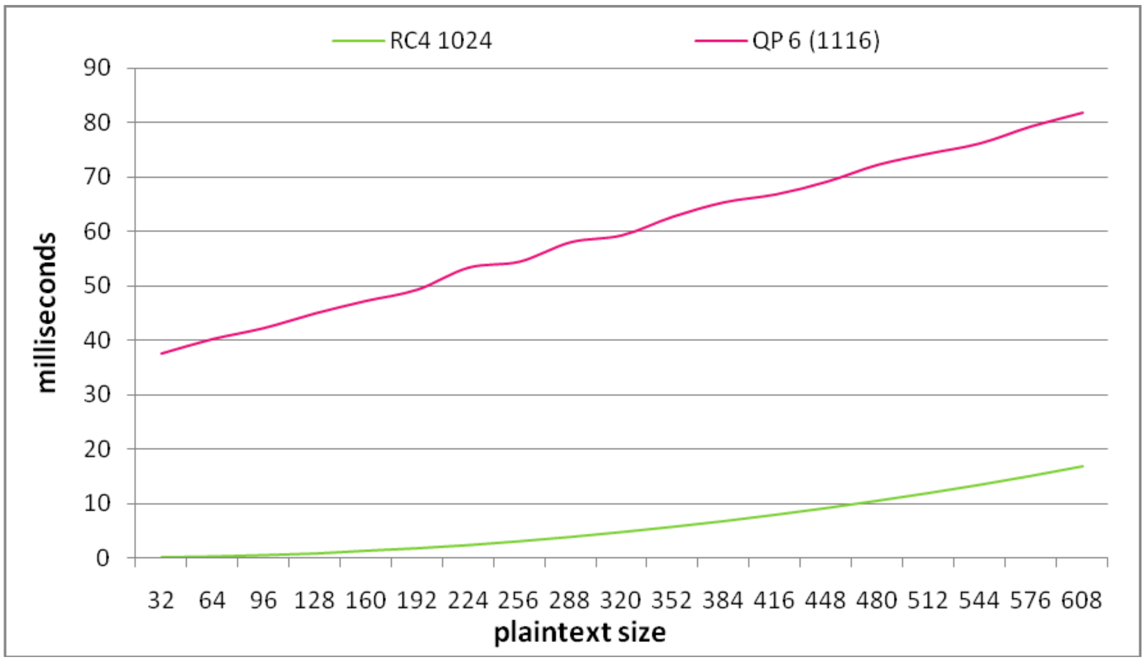

Figure 10: Overall encryption and decryption time comparison between RC4 1024-bit and QP5

In Figure 8, Figure 9 and Figure 10 the results of performance testing between QP-DYN and RC4 for different key sizes are shown. Time performances shown in the figures are the sum of the encryption and decryption times.

The sizes of the key tested are:

- 512-bit for RC4 compared to QP-DYN with a 4x4 matrix for a total of 496-bit (Figure 8);
- 768-bit for RC4 compared to QP-DYN with 5x5 matrix for a total of 775-bit (Figure 9);
- 1024-bit for RC4 compared to QP-DYN with 6x6 matrix for a total of 1116-bit (Figure 10).

We observe that the time differences in the above figures are in the following ranges:

- From 15 up to 52 milliseconds for Figure 8;
- From 18 up to 45 milliseconds for Figure 9;
- From 37 up to 65 milliseconds for Figure 10.

The time differences are within the range 15-65 milliseconds and thus they do not affect substantially the usability of QP-DYN compared to RC4. Furthermore, it is useful to remember that RC4 is not considered secure (see also the results shown in "Statistically testing *QP - Dyn* and *RC4*").

We also compared performance results on mobile environments of QP-DYN with an AES implementation performing Stream Cipher (AES CFB Stream Cipher). In particular, Figure 11 illustrates the results of a comparison between an AES CFB Stream Cipher implementation using a 256-bit key and QP-DYN with 3x3 matrixes (279-bit key).

In our experiments, the size of the plaintext where QP-DYN and AES take roughly the same time to encrypt/decrypt was about 256 bytes. As it can be seen from the figure, the time differences between AES CFB and QP-DYN are not dramatic:

- To encrypt/decrypt 32 bytes of plaintext AES CFB takes 22 milliseconds less than QP-DYN;
- To encrypt/decrypt 256 bytes of plaintext AES CFB and QP-DYN take the same time;
- To encrypt/decrypt 512 bytes of plaintext AES CFB takes 24 milliseconds more than QP-DYN.

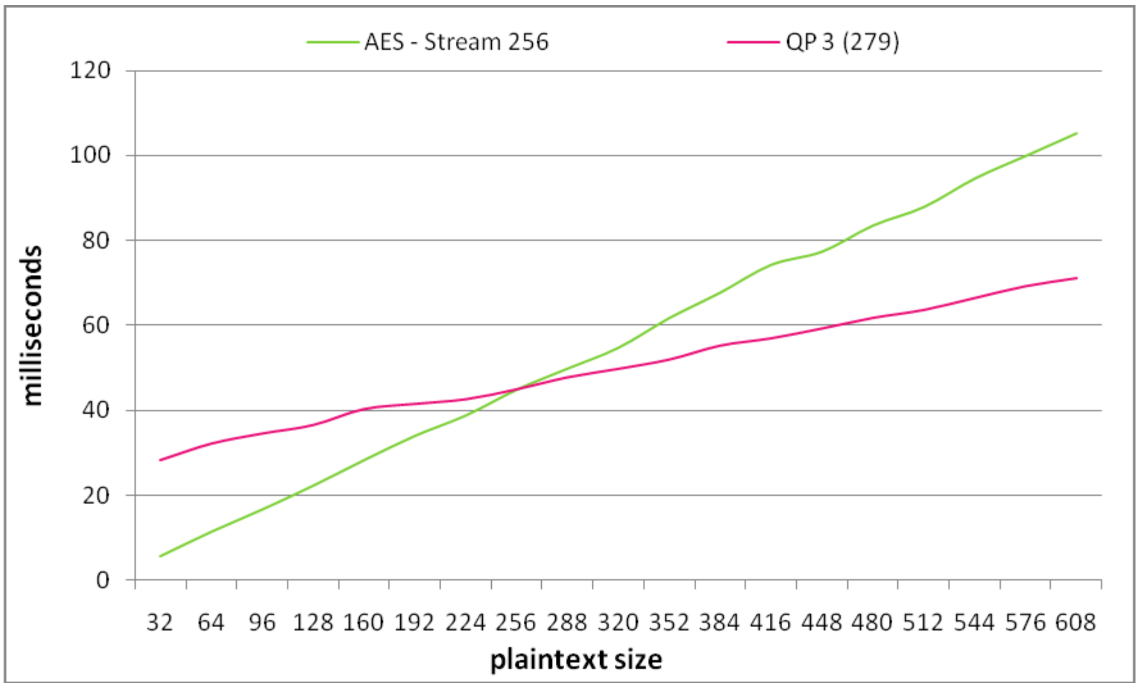

Figure 11: Overall encryption and decryption time comparison between AES CFB 256-bit and QP3

QP-DYN can be even used to perform Block Cipher operations so we compared it with AES in his standard Block Cipher implementation. As AES has been designed to perform Block Cipher operations the encryption/decryption times are better than the AES CFB. Figure 12 shows the results of our experiments for a standard implementation of AES using a 256-bit key and QP-DYN with a 3x3 matrix (279-bit).

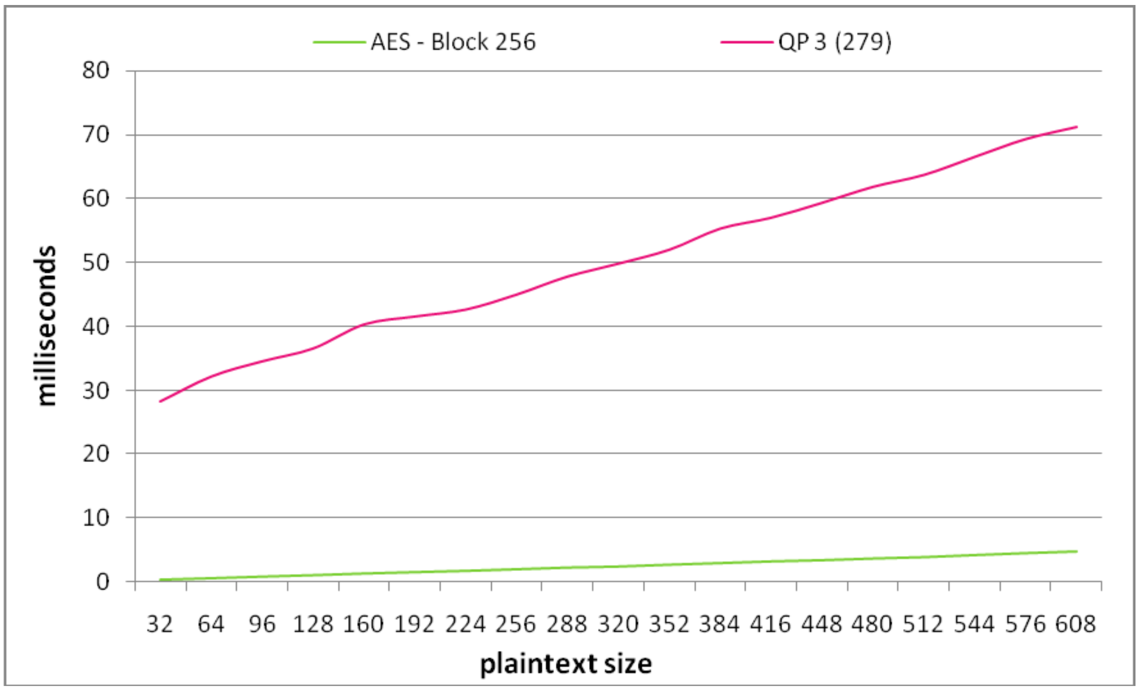

Figure 12: Overall encryption and decryption time comparison between AES 256-bit and QP3

In particular:

- to encrypt and decrypt 32 bytes of plaintext AES takes 0.3 milliseconds while QP-DYN 28.25 milliseconds;
- to encrypt and decrypt 512 bytes of plaintext AES takes 4 milliseconds while QP-DYN 63 milliseconds.

We remark again that those time differences are very small (of the order of 60 milliseconds), and thus they should not have any impact on the practical usability of QP.

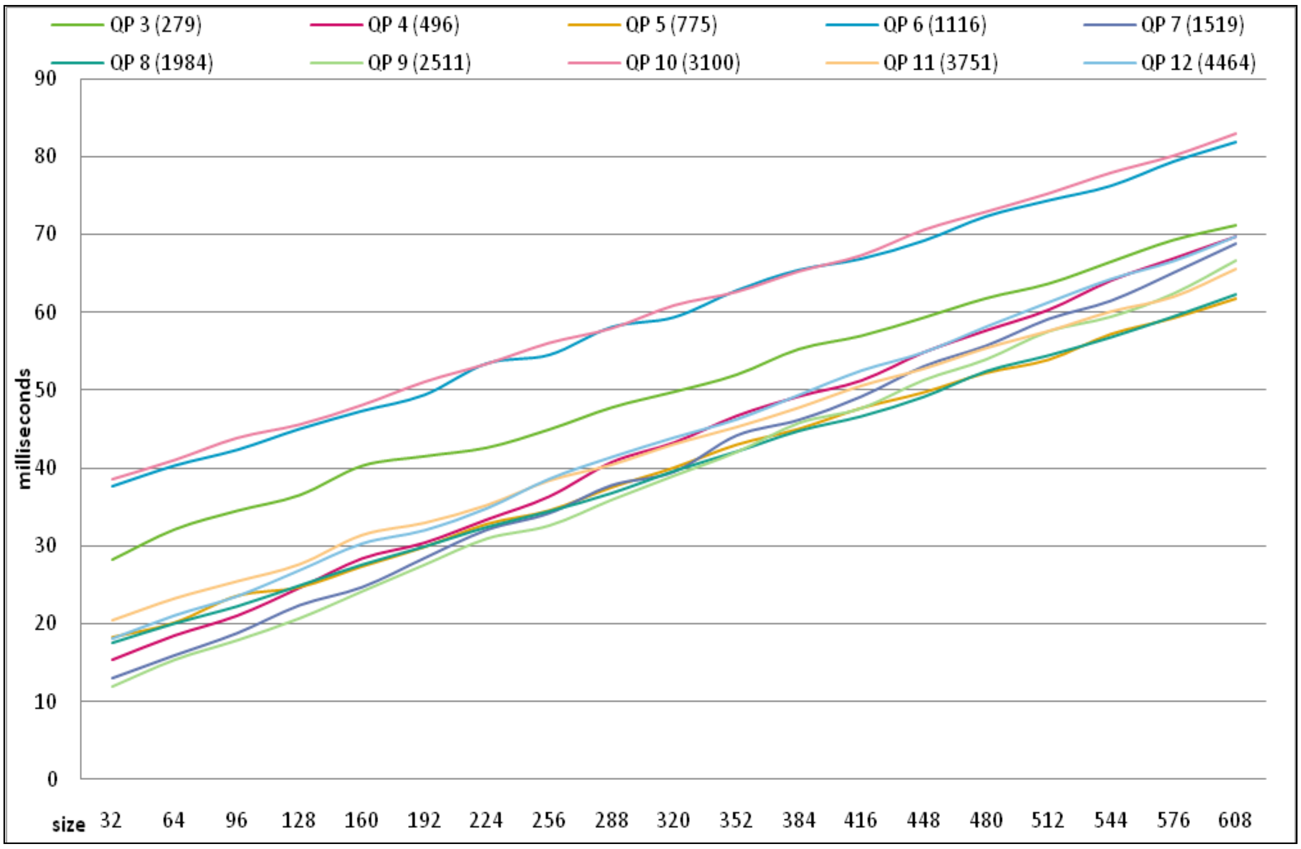

Figure 13: Overall encryption and decryption time comparison between different QP key sizes

In Figure 13 the time performance of QP-DYN for different key sizes are shown; we notice that the performance of QP-DYN appears to be stable on a larger key size range, from roughly 300 bits up to roughly 4500 bits. Table 1 shows in more details the times needed to encrypt and decrypt variable plaintext sizes using QP-DYN with different key sizes.

PLAIN TEXT SIZE	QP 3 (279)	QP 4 (496)	QP 5 (775)	QP 6 (1116)	QP 7 (1519)	QP 8 (1984)	QP 9 (2511)	QP 10 (3100)	QP 11 (3751)	QP 12 (4464)
128	36,55	24,69	24,69	45,02	22,35	25,01	20,77	45,62	27,66	26,88
256	44,99	36,39	34,53	54,54	34,21	34,53	32,65	56,11	38,43	38,6
384	55,33	49,23	45,01	65,45	46,22	44,85	45,77	65,3	47,81	49,37
512	63,75	60,31	53,88	74,39	59,21	54,53	57,5	75,31	57,66	61,26

Table 1: Overall encryption and decryption time in milliseconds for different QP key sizes and plaintext sizes

STATISTICALLY TESTING *QP - DYN* AND *RC4*

QP-DYN performs encryption and decryption in a stream cipher mode; in particular, it generates a key-stream of the same size of the plaintext to be ciphered. This key-stream is XOR-ed with the plaintext generating the ciphered text.

Such operations are the same as those performed by other stream cipher algorithms e.g., RC4 [10]. In 2005, Andreas Klein presented an analysis of the RC4 stream cipher, showing correlations between the RC4 keystream and the key ([1], [5], [4]), and again in 2008 Klein presented a successful attack on RC4 key-stream based on his 2005 work ([2], [3]). These works show that if there are correlations in the key-stream generated from a stream cipher, the stream cipher itself is reversible and can be statistically attacked recovering the key.

The National Institute of Standards and Technologies (NIST) sets the guidelines to verify a stream cipher algorithm based on pseudo-random numbers generators (PRNG) [13]. A PRNG should successfully "pass" some statistic tests in order to be usable to cipher classified information[2].

These tests are a subset of other sets of tests used to discover correlations in bit sequences generated from a PRNG. In Table 2 the tests required by NIST to use a PRNG for classified information are shown.

FREQUENCY (MONOBITS) TEST	The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether that number of ones and zeros in a sequence are approximately the same as it would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to ½, that is, the number of ones and zeroes in a sequence should be about the same.
TEST FOR FREQUENCY WITHIN A BLOCK	The focus of the test is the proportion of zeroes and ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones is an M-bit block is approximately M/2.
RUNS TEST	The focus of this test is the total number of zero and one runs in the entire sequence, where a run is an uninterrupted sequence of identical bits. A run of length k means that a run consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.
TEST FOR THE LONGEST RUN OF ONES IN A BLOCK	The focus of the test is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Long runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests.

[2] "**Classified information** is sensitive information to which access is restricted by law or regulation to particular classes of persons. A formal security clearance is required to handle classified documents or access classified data. The clearance process requires a satisfactory background investigation. There are typically several levels of sensitivity, with differing clearance requirements. This sort of hierarchical system of secrecy is used by virtually every national government. The act of assigning the level of sensitivity to data is called data classification" **Errore. L'origine riferimento non è stata trovata.**.

RANDOM BINARY MATRIX RANK TEST	The focus of the test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence.
DISCRETE FOURIER TRANSFORM (SPECTRAL) TEST	The focus of this test is the peak heights in the discrete Fast Fourier Transform. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.
NON-OVERLAPPING (APERIODIC) TEMPLATE MATCHING TEST	The focus of this test is the number of occurrences of pre-defined target substrings. The purpose of this test is to reject sequences that exhibit too many occurrences of a given non-periodic (aperiodic) pattern. For this test and for the Overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window is reset to the bit after the found pattern, and the search resumes.
OVERLAPPING (PERIODIC) TEMPLATE MATCHING TEST	The focus of this test is the number of pre-defined target substrings. The purpose of this test is to reject sequences that show deviations from the expected number of runs of ones of a given length. Note that when there is a deviation from the expected number of ones of a given length, there is also a deviation in the runs of zeroes. Runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests. For this test and for the Non-overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window again slides one bit, and the search is resumed.
MAURER'S UNIVERSAL STATISTICAL TEST	The focus of this test is the number of bits between matching patterns. The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. An overly compressible sequence is considered to be non-random.
LEMPEL-ZIV COMPLEXITY TEST	The focus of this test is the number of cumulatively distinct patterns (words) in the sequence. The purpose of the test is to determine how far the tested sequence can be compressed. The sequence is considered to be non-random if it can be significantly compressed. A random sequence will have a characteristic number of distinct patterns.
LINEAR COMPLEXITY TEST	The focus of this test is the length of a generating feedback register. The purpose of this test is to determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized by a longer feedback register. A short feedback register implies non-randomness.
SERIAL TEST	The focus of this test is the frequency of each and every overlapping m-bit pattern across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. The pattern can overlap.
APPROXIMATE ENTROPY TEST	The focus of this test is the frequency of each and every overlapping m-bit pattern. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence.
CUMULATIVE SUM (CUSUM) TEST	The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the random walk should be near zero. For non-random sequences, the excursions of this random walk away from zero will be too large.
RANDOM EXCURSIONS TEST	The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is found if partial sums of the (0,1) sequence are adjusted to (-1, +1). A random excursion of a random walk consists of a sequence of n steps of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a state within a random walk exceeds what one would expect for a random sequence.

RANDOM EXCURSIONS VARIANT TEST	The focus of this test is the number of times that a particular state occurs in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of occurrences of various states in the random walk.

Table 2: Tests required by NIST to use a PRNG for classified information

As there is a lot of work about RC4 stream cipher cryptanalysis and about the correlations notable in the key-stream generated, we decided to start analyzing the differences, noticed after performing NIST tests, between RC4 and QP-DYN.

We tested RC4 and QP-DYN using the TestU01 [17] C library for statistical testing; this library provides more tests than those required by NIST; moreover some of these tests are harder to be passed.

The tests in TestU01 library are divided in some batteries: SmallCrush, BigCrush, Rabbit, Alphabit, FIPS-140-2, pseudo DIEHARD. There is not a battery performing all the tests required by the NIST but the tests are available in the library [9]; we implemented a battery of tests performing all the tests required.

The results of our NIST test battery running for RC4 and QP-DYN algorithm are shown in Table 3: in this table, the results for RC4 appear in the left column, while the results for QP-DYN appear in the right column.

We report only one of all the runs performed, since all the runs performed, with different keys, gave always similar results.

RC4

```
========= Summary results of Nist =========

 Version:          TestU01 1.2.2
 Generator:        RC4
 Number of statistics:  60
 Total CPU time:   00:32:13.93
 The following tests gave p-values outside
[0.001, 0.9990]:
 (eps  means a value < 1.0e-300):
 (eps1 means a value < 1.0e-15):

       Test                          p-value
 ----------------------------------------------
  1  HammingWeight2, r = 0            eps
  2  HammingWeight2, r = 20           eps
  3  HammingWeight2, r = 0            eps
  5  Run of bits, r = 0               eps
  5  Run of bits, r = 0               eps
  6  Run of bits, r = 20              eps
  6  Run of bits, r = 20              eps
  9  LongestHeadRun, r = 0            eps
  9  LongestHeadRun, r = 0          1 - eps1
 10  LongestHeadRun, r = 20           eps
 10  LongestHeadRun, r = 20         1 - eps1
 12  MatrixRank, 60 x 60              eps
 13  MatrixRank, 300 x 300            eps
 14  MatrixRank, 300 x 300            eps
 15  MatrixRank, 1200 x 1200          eps
 16  MatrixRank, 1200 x 1200          eps
 17  Fourier1                         eps
 18  MultinomialBitsOver              eps
 19  AppearanceSpacings, r = 0      1 - eps1
 20  AppearanceSpacings, r = 20     1 - eps1
 21  LempelZiv                      1 - eps1
 22  LinearComp, r = 0              1 - eps1
 22  LinearComp, r = 0              1 - eps1
 24  RandomWalk1 H (L = 90)           eps
 24  RandomWalk1 M (L = 90)           eps
 24  RandomWalk1 J (L = 90)           eps
 24  RandomWalk1 R (L = 90)           eps
 24  RandomWalk1 C (L = 90)           eps
 25  RandomWalk1 H (L = 90)           eps
 25  RandomWalk1 M (L = 90)           eps
 25  RandomWalk1 J (L = 90)           eps
 25  RandomWalk1 R (L = 90)           eps
 25  RandomWalk1 C (L = 90)           eps
 26  RandomWalk1 H (L = 1000)         eps
 26  RandomWalk1 M (L = 1000)         eps
 26  RandomWalk1 J (L = 1000)         eps
 26  RandomWalk1 R (L = 1000)         eps
 26  RandomWalk1 C (L = 1000)         eps
 27  RandomWalk1 H (L = 1000)         eps
 27  RandomWalk1 M (L = 1000)         eps
 27  RandomWalk1 J (L = 1000)         eps
 27  RandomWalk1 R (L = 1000)         eps
 27  RandomWalk1 C (L = 1000)         eps
 28  RandomWalk1 H (L = 10000)        eps
 28  RandomWalk1 M (L = 10000)        eps
 28  RandomWalk1 J (L = 10000)        eps
 28  RandomWalk1 R (L = 10000)        eps
 28  RandomWalk1 C (L = 10000)        eps
 29  RandomWalk1 H (L = 10000)        eps
 29  RandomWalk1 M (L = 10000)        eps
 29  RandomWalk1 J (L = 10000)        eps
 29  RandomWalk1 R (L = 10000)        eps
 29  RandomWalk1 C (L = 10000)        eps
 ----------------------------------------------
 All other tests were passed
```

QP-DYN

```
========= Summary results of Nist =========

 Version:          TestU01 1.2.2
 Generator:        QPDyn
 Number of statistics:  62
 Total CPU time:   04:00:07.85

 All tests were passed
```

Table 3: Summary of NIST's test passed by RC4 and QP

As it can be easily seen, RC4 does not pass some tests, while QP-DYN passes all the tests required by NIST. Moreover while QP-DYN does not show correlations in the keystream generated, RC4 does it in a very short time if compared with the QP-DYN times (Total CPU time for RC4 = 00:32:13.93, Total CPU time for QP-DYN = 04:00:07.85). In every run performed, RC4 failed always the same tests, while QP-DYN always passed all the tests performed.

We tested QP-DYN even using standard test batteries that are harder to be passed; these batteries are provided by the TestU01 library (see Table 4). We report here some interesting findings:

- Some of the tests performed in the TestU01 library batteries sometimes are not passed. However, while RC4 fails always on the same tests, QP-DYN fails tests in a non-predictable way.
- Some keys in QP-DYN generate a keystream that passes all statistical tests in all batteries; this suggests that when QP-DYN is initialized with “strong[3]” keys the keystream generated seems to be much more robust.

The results of the tests performed give a clear indication that QP-DYN, when used properly with strong keys, is a very strong and robust stream cipher.

[3] QP implementation provides mechanism for determining if a generated key can be “safely” adopted by QP-Dyn. If a key seems not to be safe the mechanism modifies the starting matrix (i.e. the key) until the expected properties are satisfied.

BATTERY \ ITERATION	1	2	3	4	5
RABBIT	Multinomial BitsOver	none	Multinomial BitsOver	none	RandomWalk1
ALPHABIT	none	none	none	none	none
PSEUDODIEHARD	none	none	none	none	none
FIPS-140-2	none	none	none	none	none
CRUSH	none	none	none	none	none
BATTERY \ ITERATION	**6**	**7**	**8**	**9**	**10**
RABBIT	Multinomial BitsOver	Multinomial BitsOver	none	ClosePairs BitMatch	none
ALPHABIT	none	none	none	Multinomial BitsOver RandomWalk1	none
PSEUDODIEHARD	OPSO	none	OPSO ClosePairs	MatrixRank OPSO OQSO	MatrixRank DNA
FIPS-140-2	none	none	none	none	none
CRUSH	Hamming Corr	none	ClosePairs ClosePairs BitMatch Run	RandomWalk1	Birthday Spacings RandomWalk1 LinearComp HammingWeight2

Table 4: Results on test batteries provided by TestU01 library run on QP keystream

REFERENCES

[1] Andreas Klein, *Angriffe auf RC4;* Tagung der Fachgruppe Computeralgebra 2005, Kassel

[2] Andreas Klein, *Attacks on the RC4 stream cipher;* Designs, Codes and Cryptography 48(3), 2008

[3] Andreas Klein, *Different Attacks against the RC4 stream cipher;* Proceeding of the contact form, 2008, Brussels

[4] Andreas Klein, *Härten von RC4 artigen Cryptosystemen;* CRYPTO-TAG 2006, Bochum

[5] Andreas Klein, *Zur Sicherheit von RC4;* CRYPTO-TAG 2005, Darmstadt

[6] Certicom Research, *Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography*, Version 1.0, 2000

[7] Digital Opportunity site, *Differences in the use of infrastructure (international)*; http://www.dosite.go.jp/e/do/in-state_infra.html

[8] E. Rescorla, RFC 2631 - *Diffie-Hellman Key Agreement Method*, 1999

[9] FIPS, *FIPS PUB 197: the official AES standard*, 2001

http://www.iro.umontreal.ca/~simardr/testu01/guideshorttestu01.pdf

http://www.mozilla.org/projects/security/pki/nss/draft-kaukonen-cipher-arcfour-03.txt;

[10] K.Kaukonen, R.Thayer, *A Stream Cipher Encryption Algorithm "Arcfour"*, 1999

[11] Morris Dworkin, *Recommendation for Block Cipher Modes of Operation Methods and Techniques*, NIST, 2001

[12] NIST, *NIST IR 7298, Glossary of Key Information Security Terms*; Richard Kissel editor, 2006

[13] NIST, *Random number generation*; http://csrc.nist.gov/groups/ST/toolkit/rng/index.html

[14] NIST, *Recommended Elliptic Curves for Federal Government Use*, 1999

[15] NIST, *Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, 2006

[16] Pierre L'Ecuyer and Richard Simard, *TestU01 A Software Library in ANSI C for Empirical Testing of Random Number Generators User's guide, compact version (pp 156-157)*, 2009

[17] Pierre L'Ecuyer and Richard Simard, *TestU01;* http://www.iro.umontreal.ca/~simardr/testu01/tu01.html;