An Implementation of Secure Key Exchange by Using QR Codes

Damir Omerasevic[1], Narcis Behlilovic[2], Sasa Mrdovic[2]
[1] PBH Technologies PrinTec Group of Companies
Sarajevo, Bosnia and Herzegovina
[2] Faculty of Electrical Engineering University of Sarajevo
Sarajevo, Bosnia and Herzegovina
d.omerasevic@printec.ba

***Abstract*—This paper describes an implementation of secure RICA (Robustness, Integrity, Confidentiality and Authentication) key exchange protocol.**

Integrity, confidentiality and authentication are the base for secure message exchange. We propose adding robustness in order to ensure better availability of the system.

Robustness of presented implementation is due to Quick Response (QR) code properties. QR codes are resistant to a certain level on errors.

We used GNU[1] Privacy Guard (GnuPG or GPG), version for Windows operating system, for signing and encrypting the message, as a base for secure key exchange protocol.

***Keywords*—RICA; QR codes; Error correction; Steganography; Key Exchange; GnuPG; GPG; GNU Privacy Guard; GPG4Win; OpenPGP; PGP; RSA**

I. Introduction

This paper comes as a result of our research on open questions in our previous papers [1] [2]. In [1] we proposed using multimedia files as a source of cryptographic keys. Prior to secure message exchange, parties need to agree on a set of files to be used as key source. Encrypted messages have headers that specify index of the file used as key source and starting position in the file. The message format in [1] assumes that there are maximum of 256 files in set. Position is defined with four bytes that allows for 2^{32}, over 4 billion, positions.

In [2] we concluded that the best option from all of tested media file types is to use Flash Video (FLV) files or an open web media (WEBM) files. These files should be used as key generators.

Security of proposed encryption method in [1] is in secrecy of a set of files used. The set of files might be considered as a master key or some sort of key encryption key, while the bits of files used to encrypt messages have a role of session keys. Key size of this master key is practically limitless since the number of possible file sets is practically limitless. However, in implementation described in this paper, we limit the size of the set to 256.

Main open question in [1] was how to securely exchange information on data set. Therefore, our work in this paper is oriented towards defining a protocol for agreement on both sets and ordering of files.

[1]"GNU" is a recursive acronym for "GNU's Not Unix!".

The implementation described in this paper proposes that the data set (i.e. sets and ordering of files) is first encrypted by GPG. We use GPG, for signing and encrypting the message, as a base for secure key exchange protocol. GPG is compliant with RFC 4880 [3]. GPG-signed and encrypted message is embedded into QR code.

QR code was invented in 1994 by Denso/Toyota [4], for tracking automotive information and depending on the version set, can store up 4,296 alphanumeric characters [5].

A QR code is made of darker coloured square dots (modules), inside the square with lighter coloured background, or vice versa, according to ISO/IEC18004 standard [6]. What is important here is that contrast between background and square dots is good enough.

There are four error correction levels used for QR codes. Each correction level is appropriate for certain amount of damage, and therefore has different amount of additional data for correcting:

1) Level L up to 7% of damage
2) Level M up to 15% of damage
3) Level Q up to 25% of damage
4) Level H up to 30% of damage

QR codes provide an opportunity to visualise initial data for key exchange process. Visualisation enables data conversion to different visual/video/picture formats, without loosing original message embedded.

Moreover, error correction levels embedded into QR code are used for improving robustness of our secure information (on data set).

The QR code can be recognised/read by some kind of imaging device (like camera) and after that decoded by using Reed-Solomon error correction, until decoding is properly finished. Reed-Solomon error correction allows correct reading, even if a certain part of QR code is damaged.

The decoding speed of the QR code can be made 20 times faster than that of other matrix symbols [4]. QR decoders can be easily implemented in hardware.

There is also a very large usage of QR codes in mobile phones industry [5].

The rest of the paper is organised as follows:

Related work is addressed in section 2. Section 3 explains our idea on how to use GPG with QR code. Testing results

are presented in section 4. Conclusion and discussion, as well as directions for future research work, are in section 5.

II. Related work

To the best of our knowledge, we did not find any work related with QR codes and robustness, and therefore we briefly describe work which is the closest and related to QR codes and security in global, and after that we describe recent attacks on GPG.

In the last few years we experienced a very large application of QR codes in steganography, authentication and video watermarking.

In [7], QR code and image processing techniques are used to construct a nested steganography scheme. A lossless data is embedded into a cover image. The data do not have any distortion, when compared to the extracted data and original data. Since the extracted text is lossless, the error correction rate of QR encoding must be carefully designed. Authors of the paper found that 25% error correction rate is suitable for the goal. This scheme is also robust to Joint Photographic Experts Group (JPEG) attacks. This paper is related to our work because it shows that we need to carefully design the error correction rate.

In [8], authors proposed a geo-location based QR code authentication scheme using mobile phone, to defeat against man-in-the-middle phishing attacks. The proposed scheme provides convenience, mobility, and security for the user. This paper is also related to our work because it shows that QR codes could be easily implemented in mobile phones.

Paper [9] proposes a video watermarking with text data (verification message) by using QR code. QR code is prepared to be watermarked by SVD (Singular Value Decomposition) and DWT (Discrete Wavelet Transform). In addition to that, logo/watermark gives the authorized ownership of video document. This paper is related to our work because it shows that QR codes could be easily implemented in video watermarking schemes.

In [10], authors proposed another algorithm for the analysis and correction of the distorted QR code, by combining Canny edge detection with contours finding algorithms. This paper is related to our work because it shows that QR codes could be corrected in different ways.

GPG could use Rivest-Shamir-Adleman (RSA) cryptosystem and therefore we briefly describe both RSA and recent attacks on GPG.

PhD work of Tromer [11] in 2007 presented how to break 1024-bit RSA keys with hardware-based cryptanalysis tools.

According to [12], RSA 4096-bit keys should be secure after year of 2031.

However, a very recent research of Genkin, Shamir and Tromer [13] [14] showed that it was possible to make an attack on specific versions of GPG, which can extract whole 4096-bit RSA keys during approximately of one hour, by using acoustic emanations from CPU of targeted/tested laptops.

Regardless of this recent side channel attacks, which are bound to specific versions of GPG, RSA still remains secure. A research of Chong and Quisquater [15] proposed a direction to construct efficient countermeasures to both side channel analyses and fault attacks on RSA.

In the next section we explain how to use GPG with QR code to prepare information on a file set for secure exchange.

III. How to use GPG with QR code (how to prepare information on a file set for secure exchange)

In this paper, we used the formal model and the message structure from our work in [1] as a reference. We propose a solution to the issue, recognised in [1], of distribution of this ”master” key.

The best source for FLV/WEBM files is YouTube. However, there are some legal restrictions in usage of YouTube files [16]. Therefore, an implementation presented here, which uses YouTube web site, is for the proof of concept only.

All YouTube video files could be accessed by the following Uniform Resource Locator (URL) syntax:

http://www.youtube.com/watch?v=key

where *key* is 11-alphanumeric YouTube video identification (YouTube Video ID), like, for example, *”voLNA8LdcCw”* (without quotes).

Our initial message format assumes that there are 256 files in a set, i.e 256 file set is described with 256 lines of 11-alphanumeric YouTube Video ID.

By using YouTube Video ID, we could access all video file formats from one place and, depending on device and appropriate web browser, automatically show the best fitted video format for device which is currently used.

In order to get specific video format from specific YouTube Video ID, we need to parse HyperText Markup Language (HTML) code, and identify exact URL locations for FLV/WEBM files, for each of 256 YouTube Video IDs separately.

Considering the fact that we have all information about complete file set in one initial message, there is no need in this implementation to have separate messages for file sets and orders.

In Fig.1 we describe secure exchange process, initiated from sender side.

The process from sender side consists of eight steps:

1) Prepare initial message by sender,
2) Sign and encrypt initial message with GPG,
3) Prepare/encode QR code,
4) Send QR code to receiver,
5) Receive QR code by receiver,
6) Decode QR code,
7) Decrypt and verify signature with GPG, and
8) Prepare/calculate identical copy of initial message.

Initial message file consist of 256 lines. Each line is 11-alphanumeric YouTube video identification (ID), plus additional end of line characters, line feed (LF) and carriage return (CR). The total of 3,328 bytes is used.

In Fig.2 we describe secure exchange process, finalized from receiver side.

The process from receiver side consists of six steps:

1) Sign (identical copy of) initial message with GPG,
2) Prepare/encode QR code,
3) Send QR code to sender,
4) Receive QR code by sender,
5) Decode QR code, and
6) Verify signature with GPG.

A. Testing procedure

We first describe operating system and platform used for testing. Then we present the results of testing.

1) Operating System and Platform for Testing: The platform on which we tested our system is Microsoft Windows 7.

We used GPG4Win command-line utility *gpg.exe*, together with appropriate parameters, for signing and encrypting of initial message. The initial message has complete file set, one file in each line, described by 11-alphanumeric YouTube ID, for implementation presented in this paper. For this test we chose YouTube Video ID manually.

For QR code generation, we used Microsoft Studio 2005, Visual C# part of the Studio, and adopted source code from [17] to create command-line applications. We prepared additional batch scripts for easier usage. Scripts include parameters needed for command-line application to achieve efficiency and performance improvement of overall measurement process.

Using command-line tools we created QR codes in JPEG, Portable Network Graphics (PNG), 24-bit Bitmap (BMP) and Graphics Interchange Format (GIF).

We used Microsoft Paint to transform initial 24-bit BMP to monochrome BMP files.

It is possible to use three error correction levels for QR codes (L, M and Q) in our proposed implementation. Due to the length of our message for QR code, we could not use level H for error correction.

We also scaled every bit of QR code, by using scales from one to four, while encoding QR code, and testing results in correlation with all tested error level codes and all tested graphic formats.

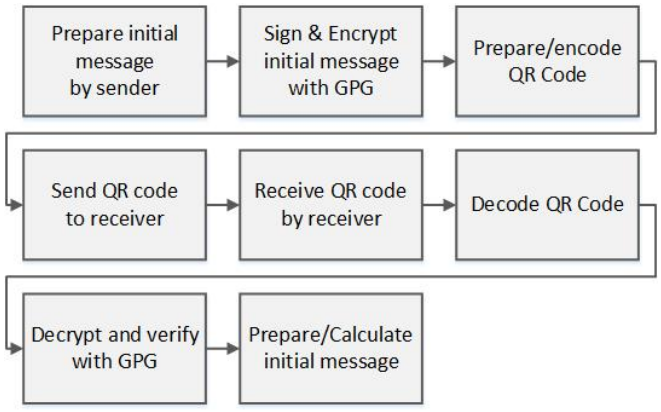

Fig. 1. Secure Key Exchange with QR code (sender side)

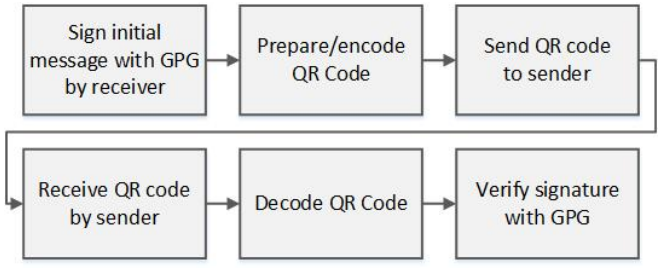

Fig. 2. Secure Key Exchange with QR code (receiver side)

IV. Testing Results

Considering the fact that we used 256 file names, which are described with 11-alphanumeric YouTube Video ID, we had constant length of input secret message/file.

Therefore, it was enough to make a test with one combination of files, because other combinations of files, due to the same length of initial message/file, would have almost similar, if not the same, results.

From initial message size of 3,328 bytes sender used, sender gets 1,415 bytes after signing and encrypting message.

Comparing QR codes, depending of the level of error correction and tested graphic formats, we got results in size of bytes, for the same signed and encrypted message (i.e. the same initial secret message of 1,415 bytes), in the following tables.

The best result for every level of error correction is one for a graphic format with the smallest length in bytes.

Table I shows comparison of results for QR code with error level correction L and tested graphic formats.

Table II shows comparison of results for QR code with error level correction M and tested graphic formats.

Table III shows comparison of results for QR code with error level correction Q and tested graphic formats.

TABLE I
Results for QR code in different picture formats, with error level correction L

JPEG (bytes)	PNG (bytes)	Mono BMP (bytes)	GIF (bytes)	QR Code scale
11.815	5.333	*2.078*	3.347	1
41.128	14.506	8.094	*7.225*	2
84.856	26.270	18.110	*12.105*	3
92.626	35.754	32.126	*17.931*	4

TABLE II
Results for QR code in different picture formats, with error level correction M

JPEG (bytes)	PNG (bytes)	Mono BMP (bytes)	GIF (bytes)	QR Code scale
14.702	6.728	*2.902*	4.020	1
51.971	18.492	10.250	*8.970*	2
107.757	33.450	23.806	*15.150*	3
117.897	45.641	40.742	*22.315*	4

In summary, the best results (the smallest length in bytes), for all error level correction tested, were for GIF files, if we use QR code scale greater than one.

If we use QR code scale equal to one (i.e. without scaling), the best results (the smallest length in bytes), for all error level correction tested, were for monochrome BMP files.

V. Conclusion

In this paper we described an implementation of secure key exchange using QR codes. RICA key exchange implemented is not only secure, but also more robust.

Robustness of presented implementation is due to QR codes properties. QR codes are resistant to a certain level on errors. In our case we showed that we could use up to 25 percent of error level correction.

QR code poses an ability to preserve correct information in a case of transformation of initial QR code/picture from one format to another. In that way we were able to compress our initial QR code almost six times, either by using monochrome BMP instead of JPEG file (in case where we had no QR code scaling), or by using GIF instead of JPEG file (in case where we had QR code scaling greater than one).

Photo cameras are integral part of most existing mobile phones. Different QR code recognition software is usually installed on mobile phones, too. Considering the fact that there is GPG version for Android smart phones, and also the fact that QR code readers already exists in smart phones, it is reasonable to propose an implementation of the process for secure exchange of keys, for smart phones on Android platform.

Moreover, the implementation which is presented here, could also be transformed to Windows Mobile (WM) platform, i.e. on smart phones which have Windows Mobile as an operating system.

The transformation could be done to iPhone platform, on smart phones which have iPhone operating system (iOS), too.

Our future work will be oriented mostly towards:

1) Improvement of existing implementation by choosing YouTube Video ID automatically,
2) Transformation of the implementation on other platforms/operating systems, like Android, WM or iOS, and
3) Comparison of performances from different smart phone platform(s) to laptop/desktop platform based on Windows operating system.

TABLE III
Results for QR code in different picture formats, with error level correction Q

JPEG (bytes)	PNG (bytes)	Mono BMP (bytes)	GIF (bytes)	QR Code scale
19.934	9.081	*4.046*	5.110	1
70.514	24.979	14.626	*11.804*	2
146.905	45.682	31.806	*20.367*	3
159.236	62.340	55.586	*30.446*	4

References

[1] D. Omerasevic, N. Behlilovic, and S. Mrdovic, "CryptoStego - A Novel Approach for Creating Cryptographic Keys and Messages," in *Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on*, pp. 83–86, 2013.

[2] D. Omerasevic, N. Behlilovic, S. Mrdovic, and A. Sarajlic, "Comparing Randomness on Various Video and Audio Media File Types," in *Telecommunications Forum (TELFOR), 2013 21st*, pp. 381–384, Nov 2013.

[3] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and F. Thayer, "RFC 4880 - OpenPGP Message Format," tech. rep., Internet Engineering Task Force, Nov. 2007.

[4] I. T. S. C. (www.itsc.org), "QR code," *Synthesis journal*, pp. 59–78, 2008.

[5] Y. Liu, J. Yang, and M. Liu, "Recognition of QR Code with mobile phones," in *Control and Decision Conference, 2008. CCDC 2008. Chinese*, pp. 203–206, July 2008.

[6] International Organization for Standardization, "Information Technology — Automatic Identification and Data Capture Techniques — QR Code 2005 Bar Code Symbology Specification." ISO/IEC 18004:2006, 2006.

[7] C.-H. Chung, W.-Y. Chen, and C.-M. Tu, "Image Hidden Technique Using QR-Barcode," in *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, pp. 522–525, Sept 2009.

[8] K.-C. Liao and W.-H. Lee, "A Novel User Authentication Scheme Based on QR-Code.," *JNW*, vol. 5, no. 8, pp. 937–941, 2010.

[9] G. Prabakaran, R. Bhavani, and M. Ramesh, "A robust QR-Code video watermarking scheme based on SVD and DWT composite domain," in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*, pp. 251–257, Feb 2013.

[10] A. Sun, Y. Sun, and C. Liu, "The QR-code reorganization in illegible snapshots taken by mobile phones," in *Computational Science and its Applications, 2007. ICCSA 2007. International Conference on*, pp. 532–538, Aug 2007.

[11] E. Tromer, "Hardware-Based Cryptanalysis." http://cs.tau.ac.il/ tromer/phd-dissertation/, 2007. [Accessed 12.04.2014.].

[12] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for Key Management - Part 1: General (Revision 3)." http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf, 2012. [Accessed 12.04.2014.].

[13] D. Genkin, A. Shamir, and E. Tromer, "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis." http://eprint.iacr.org/2013/857, 2013. [Accessed 12.04.2014.].

[14] T. Worstall, "Researchers Break RSA 4096 Encryption With Just A Microphone And A Couple Of Emails." http://www.forbes.com/sites/timworstall/2013/12/21/researchers-break-rsa-4096-encryption-with-just-a-microphone-and-a-couple-of-emails/, 12 2013. [Accessed 26.04.2014.].

[15] C. H. Kim and J.-J. Quisquater, "How can we overcome both side channel analysis and fault attacks on RSA-CRT?," in *Fault Diagnosis and Tolerance in Cryptography, 2007. FDTC 2007. Workshop on*, pp. 21–29, Sept 2007.

[16] YouTube, "Terms of Service." http://www.youtube.com/t/terms, 03 2014. [Online; accessed 02.04.2014.].

[17] twit88, "Open Source QRCode Library." http://www.codeproject.com/Articles/20574/Open-Source-QRCode-Library, September 2007. [Online; accessed 30.3.2014.].