Analysis of security problems in a medical image encryption system*

Gonzalo Alvarez[1**], Shujun Li[2] and Luis Hernandez[1]

[1] Instituto de Física Aplicada, Consejo Superior de Investigaciones Científicas, Serrano 144, 28006 Madrid, Spain
[2] Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR, China

Abstract

Recently a new system for the secure transmission and efficient storage of medical images interleaved with patient information has been proposed in 2003 by Rajendra Acharya *et al*. In this paper we analyse the security of this system, showing how to improve it to obtain a truly secure system.

Keywords: Cryptography; Medical images; Patient information

1. Introduction

Secure access to medical images stored on digital media is of greatest importance. These images may be very large in size and number, and usually contain confidential data. Therefore, two important goals are: 1) to safeguard the confidentiality of the patient's personal data; and 2) to save as much space as possible, to reduce the cost of storage and increase the speed of transmission, but without degrading quality. Today's practical digital transmission channels, such as Internet, and digital storage scenarios, such as hard disks, CD or DVD, are considered to be perfect, with no noise or other interference. Thus, no error control coding techniques are needed.

In [1], the authors propose a new technique to transmit and store medical images, interleaved with confidential patient information. As a first step to guarantee the security of the patient information, this is encrypted using an algorithm developed by the authors. Next, the encrypted information is interleaved with the medical image. The watermarking process consists of swapping each ASCII code in the encrypted text file with the least significant bit (LSB) of the grey scale bit by bit. Eight bits of the text file (thus one ASCII character) replace LSBs of eight consecutive pixels of the image. The interleaved image is thus transmitted over noisy channels and stored.

In the next section we show that the encryption procedure followed in [1] does not correctly work in practice and is extremely easy to break, and we suggest a way to encrypt the patient information via standard encryption algorithms to obtain a more secure system.

2. Analysis of the encryption algorithm

The encryption algorithm proposed in [1] can be mathematically expressed as:

$$T_e = 100 \times \ln(T_0 \times 2) - 300 \qquad (1)$$

* This paper has been published in Computers in Biology and Medicine, vol. 37, no. 3, pp. 424-427, 2007.
** Corresponding author's e-mail: gonzalo@iec.csic.es.

where T_e is the encrypted text and T_0 is the ASCII code of the original text (or graphics file). T_e is stored as an integer, which requires rounding it off to the nearest integer. Note that $T_e = -\infty$ when $T_0 = 0$, which cannot be stored as a normal integer. This means that Eq. (1) cannot be used to encrypt black pixels in the images. The decrypted text is obtained by

$$T_0 = e^{\frac{T_e + 300 - 100 \times \ln 2}{100}} = 0.5 e^{0.01 \times T_e + 3} . \quad (2)$$

Note that the formula given in [1] was wrong and has been corrected in Eq. (2). Although in [1] it is hinted that real values might be rounded off to the nearest integer to calculate T_e, and T_0 back from T_e, we have floored real values when encrypting and ceiled real values when decrypting. Otherwise, it was impossible to obtain the same results as shown in Table 2 of [1]. Given that the valid range of input values is not mentioned in [1], we assume it covers the complete ASCII value table, i.e., from 0 to 255. This range is indeed required if not only text but any other type of file is to be encrypted, as already suggested in [1].

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**000**	-231	-162	-121	-93	-70	-52	-37	-23	-11	-01	009	017	025	033	040
1	046	052	058	063	068	073	078	082	087	091	095	098	102	106	109	112
2	115	118	121	124	127	130	133	135	138	140	143	145	147	149	152	154
3	156	158	160	162	164	166	168	170	171	173	175	177	178	180	182	183
4	185	186	188	189	191	192	194	195	196	198	199	201	202	203	204	206
5	207	208	209	211	212	213	214	215	217	218	219	220	221	222	223	224
6	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
7	241	242	**242**	243	244	245	246	247	248	**248**	249	250	251	252	**252**	253
8	254	255	256	**256**	257	258	259	**259**	260	261	262	**262**	263	264	**264**	265
9	266	**266**	267	268	269	**269**	270	271	**271**	272	273	**273**	274	**274**	275	276
A	**276**	277	278	**278**	279	**279**	280	281	**281**	282	**282**	283	284	**284**	285	**285**
B	286	**286**	287	288	**288**	289	**289**	290	**290**	291	**291**	292	**292**	293	294	**294**
C	295	**295**	296	**296**	297	**297**	298	**298**	299	**299**	300	**300**	301	**301**	302	**302**
D	303	**303**	304	**304**	**304**	305	**305**	306	**306**	307	**307**	308	**308**	309	**309**	310
E	**310**	**310**	311	**311**	312	**312**	313	**313**	**313**	314	**314**	315	**315**	316	**316**	**316**
F	317	**317**	318	**318**	319	**319**	**319**	320	**320**	321	**321**	**321**	322	**322**	323	**323**

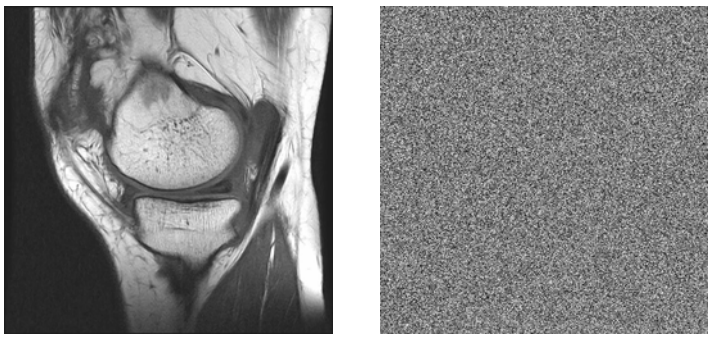
Table 1. Performance of the encryption system proposed in [1]. Values correspond to the ciphertext. Repeated values are printed in bold face, meaning that correct decryption is impossible. This table proves that the method presents errors for normal text (ASCII values between 32 and 127) and does not work at all for the rest of ASCII values, as required when encrypting graphic files.

From a practical point of view, this algorithm cannot work because we have found that, as a consequence of the compressing nature of the logarithmic function, Eqs. (1) and (2) do not yield exact reconstruction: there are many ASCII values for which the corresponding ciphertext is the same, thus preventing the correct decryption of the given encrypted value. In Table 1 all ASCII values and their corresponding encrypted values are shown. As can be observed in Table 1, there are 61 repetitions, rendering the system useless. For instance, the values 113 (“q”) and 114 (“r”), 71 and 72 in hexadecimal respectively, are both encrypted as 242 (“_”) and when decrypting both will be deciphered as 113 (“q”). All these reasons make the system impractical because it cannot work correctly. This table was generated using the following C source code:

```
for ( i=0; i<256; i++ )
        printf( "%c %c %c\n", i, ciphertext[i]=floor(100.0*log((double)(2.0*i))-300.0),
ceil(0.5*exp(0.01*ciphertext[i]+3)));
```

From a security point of view, even if it had worked in practice, this would have been a very weak encryption algorithm for two reasons. First, there is no secret key. Therefore, it is not a true encryption scheme, but an encoding scheme. Anyone who knows its operation method can easily recover the original text. Second, even if the operation method is unknown to an attacker or even if a secret key is introduced, the algorithm is a simple substitution cipher, which means that the same plain-character will always be encrypted into the same cipher-character under the same key. For instance, in Fig. 2(a) of [1], the text "Name of the" appears twice. In Fig. 2(b) of [1] it is observed that it results in the same encrypted text. Given the highly formatted nature of the information to be protected due to standard headers in file formats, etc., it would be a trivial task to decrypt such a cryptogram even with no knowledge of the key. As a conclusion, this encryption method offers no protection at all.

If the security of the information being protected is to be improved, it should be advisable to use any of the standard encryption algorithms widely accepted today in all sorts of secure applications, such as Triple-DES [2], AES [3], or many others [4]. All of these algorithms use a secret key of variable length (usually ranging from 128 to 256 bits), which makes unfeasible a brute force attack to try all possible combinations of the secret key. They are very fast and easy to implement in any application, due to the large amount of software libraries and packages that give support for them.



a) b)

Fig. 1. Image encryption using AES with 128-bit secret key A87B43FF04E109CD5AB011E62AC890DB: a) Original radiography; b) encrypted radiography.

Just as way of example, in Fig. 1 a sample medical image encrypted using the library implementation of AES running in CBC mode found in Microsoft Visual Studio .NET 2003 Professional Edition is shown. As can be observed, no information is revealed to the attacker. The level of protection achieved using this encryption algorithm is considered to be 100% secure by the cryptographic community considering current state of the art. Furthermore, these encryption algorithms are extremely fast, achieving encryption speeds of 488 Mb/s in a 2.1 GHz processor [5].

Both the original and encrypted images have dimensions of 512 x 512 pixels with 256 grey levels, consuming a total memory size of 512*512*8 = 2097152 bits = 256 KB. The encryption operation here described transforms one image into another of exactly the same size, thus the encrypted image has also 256 KB. There is no size change due to the encryption algorithm. In fact, any graphic format of the user's choice can be used for the images (original or encrypted).

a)

NGEE ANN HEART FOUNDATION SINGAPORE
Patient Ref.No:49342911
Name of the doctor:Dr.Chee
Name of the patient:Ms.Kwang Liu
Age: 56 years
Address:Kismis Avenue, Block 92, #02-02, Singapore
Date of Admission:12.01.2001
Results:T wave inversion

b)

A8F9B3E6CBE977953AA16B3496DF7D7CC9E1DC261A23BCB9E153A2064C3B646601
EE0A719E6D838CCA00A878E51018DB2DE0C0BD519E02901BDD0131047C69F44A93
A47443731382C422E3F7DE2FD7A80113E7680910BEEDE0F68689D9DE97B5A1E35DC
AB6F8728CA28ADDAC423824BC30448BFECEF78EDE372CA68C880AD5F427BE75E8
D0CAA1ACE40D4C5CA9BC27174BEE15CEB36DDC338755579864C9455826DC916375
32CB35441EAB17D27CEE562402BD97AD4C9932288276353708006873 3F199DD4546BB
93B31D590773615D10AC6DD7BB75CA783B5480F5EA27947CDE17105C4F1EB0A7FBE
83ABB836C2126B

Fig. 2. Text encryption using AES with another 128-bit key: a) Original text; b) encrypted text. The encrypted text is itself encoded in hexadecimal to be printed.

As a final example, let us consider the clear text of Fig. 2(a) and its encrypted version using the Advanced Encryption Standard (AES) with a 128-bit key, obtaining a result considered unbreakable in the long term.

3. Conclusions

In its present form, the system proposed in [1] lacks security and cannot be used in practice. We have pinpointed the security defects of [1] and suggested a very simple way of encrypting medical images resorting to publicly available standard algorithms so that the final scheme is truly secure.

Acknowledgements

This work was partially supported by Ministerio de Educación y Ciencia (Spain), research grant SEG2004-02418, and by Consejería de Sanidad de la Junta de Castilla y León (Spain), research grant SAN/1052/SA29/05.

References

[1] Rajendra Acharya U., P. Subbanna Bhat, Sathish Kumar and Lim Choo Min, Transmission and storage of medical images with patient information, Computers in Biology and Medicine 33, 303-310 (2003).

[2] ANSI X9.52, “Triple data encryption algorithm modes of operation”, draft, 1996.

[3] Joan Daemen and Vincent Rijmen, The Design of Rijndael, Springer (2002).

[4] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, (1996).

[5] W. Dai, Speed comparison of popular crypto algorithms, available online at http://www.eskimo.com/~weidai/benchmarks.html.