The Misuse of RC4 in Microsoft Word and Excel

Hongjun Wu

Institute for Infocomm Research, Singapore
hongjun@i2r.a-star.edu.sg

Abstract. In this report, we point out a serious security flaw in Microsoft Word and Excel. The stream cipher RC4 [9] with key length up to 128 bits is used in Microsoft Word and Excel to protect the documents. But when an encrypted document gets modified and saved, the initialization vector remains the same and thus the same keystream generated from RC4 is applied to encrypt the different versions of that document. The consequence is disastrous since a lot of information of the document could be recovered easily.

1 Introduction

After more than two decades of public research on cryptography, many practically secure ciphers have been proposed. If we use those ciphers properly, adequate protection could be achieved. Unfortunately, when the ciphers are implemented in products, various security problems may arise. A well-known story is related to an old version of the Netscape browser. In the implementation of the Secure Socket Layer (SSL) in Netscape 1.1, the key of the symmetric key cipher is derived from the current time and the process ID (or the system time). The key space becomes severely limited, and even the 128-bit encryption version could be easily cracked [4].

For the implementation of stream ciphers, the basic principle is that if the same key is used for more than once, different initialization vectors should be used to prevent the same keystream from being used to encrypt more than one message. When the stream cipher is used in the data transmission, normally people would follow this principle strictly. However, in the environment where the document needs to be edited and modified, such principle may be forgotten. This kind of mistake takes place in the Microsoft Office (Word and Excel) encryption – the same key and the same initialization vector are allowed to encrypt different versions of a document. This happens as follows. We encrypt a Microsoft Office (Word or Excel) document with a password and save that file. Later that document is modified and being saved again. In this process, the key and initialization vector remain unchanged, so the same keystream is used to protect two different versions (the original and the modified versions) of the documents. By XORing those two versions, we could obtain a lot of information about the document.

The above attack could take place in real life. Suppose that Alice and Bob are working on the same Microsoft Office (Word or Excel) document. They share the same password and use that password to protect the document. They would make changes to the document and the document is encrypted and transmitted between them for a number of times. In this process, the same password and initialization vector are used to protect all the modified versions of that document and the document could be easily recovered from those intercepted files with high chance.

Here is another example. Suppose that Alice is working on a Microsoft Office document (Word and Excel) and she uses a password to protect it. During the process, Alice may need to backup her files. An attacker could retrieve a lot of information from those backup files even though the attacker does not know Alice's password.

This report is organized as follows. The background information on the security of Microsoft Office is given in Section 2. We illustrate the misuse of RC4 in Microsoft Word and Excel in Section 3 and Section 4, respectively. Section 5 discusses the countermeasure and Sections 5 concludes this report.

2 Introduction to the Security of Microsoft Office

The Microsoft Office includes Word, Excel, PowerPoint and other components [5]. We consider only the Word, Excel in this report. There are five versions of Microsoft Office: 95, 97, 2000, XP and the latest Office 2003.

The encryption in Microsoft Office 95 is to XOR the cascaded password with the message. From the cryptographic point of view, this encryption scheme does not provide any security protection. The other versions of Microsoft Office use RC4 to protect the documents. The early export version of Microsoft Office supports only the 40-bit encryption. The 40-bit encryption scheme is vulnerable to the brute force attack since several Giga instructions can now be processed in one second on a personal computer. Both the XOR and the 40-bit encryption fail to provide sufficient protection for the documents. There have been many cracking software on Microsoft Office documents, but almost all of them aim at these two types of weak encryptions.

With the 128-bit version being supported in Microsoft Office, one may expect that sufficient protection could be achieved since a random 16-byte ASCII password contains more than 100-bit secret information and the brute force attack fails. Unfortunately, the stream cipher RC4 is misused in Microsoft Office. It has been well known that the initialization vector in a stream cipher should be used properly. But Microsoft Office manages the initialization vector improperly and the same keystream could be used to encrypt different versions of a document. That is the flaw we will illustrate in detail in the rest of this report.

Remarks 1. In Microsoft Office, using the same key with different initialization vectors would not leak the secret key. The reason is that the secret key and the initialization vector are hashed together and the hash output is used as secret key in RC4. So the key schedule weakness of RC4 [3] has no effect on the security of Microsoft Office.

Remarks 2. To the best of our knowledge, the security flaw given in the rest of this report has not been reported in public.

3 The Misuse of RC4 in Microsoft Word

In this section, we show that RC4 is implemented Microsoft Word in an insecure way and the 128-bit RC4 fails to protect the document as expected. The flaw is that the same initialization vector is used when the document is modified. This flaw causes part of the documents being recovered with negligible amount of computation.

3.1 Evidence that RC4 is misused in Microsoft Word

In this subsection, we use Microsoft Word 2002 to illustrate that RC4 is misused in Microsoft Office.

We create a Word document which contains only one sentence "*Anti-virus researchers from Symantec yesterday spotted the first virus capable of infecting 64-bit Windows systems*". Before the document is encrypted, we open the document in binary format and obtain Fig. 1. Then we encrypt the file according to Appendix A (we choose the 'Microsoft Strong Cryptographic Provider' that supports the 128-bit RC4 encryption). Once encrypted, we obtain Fig. 2 and we notice that the encrypted data looks random. Then we change the sentence in the document to "*Anti-virus researchers at Symantec yesterday spotted the first virus capable of infecting 64-bit Windows systems*", i.e., the word '*from*' is changed to '*at*'. After saving the changes, the binary format of the modified document is shown in Fig. 3 (using different password or file names would result in different content, but at the end of the experiment the same conclusion would be reached).

```
000a00  41 6E 74 69 2D 76 69 72  75 73 20 72 65 73 65 61  Anti-virus resea
000a10  72 63 68 65 72 73 20 66  72 6F 6D 20 53 79 6D 61  rchers from Syma
000a20  6E 74 65 63 20 79 65 73  74 65 72 64 61 79 20 73  ntec yesterday s
000a30  70 6F 74 74 65 64 20 74  68 65 20 66 69 72 73 74  potted the first
000a40  20 76 69 72 75 73 20 63  61 70 61 62 6C 65 20 6F   virus capable o
000a50  66 20 69 6E 66 65 63 74  69 6E 67 20 36 34 2D 62  f infecting 64-b
000a60  69 74 20 57 69 6E 64 6F  77 73 20 73 79 73 74 65  it Windows syste
000a70  6D 73 2E 0D 00 00 00 00  00 00 00 00 00 00 00 00  ms..............
```

Fig. 1. Binary format of the original document (unencrypted)

The encrypted data starts from the address 0xa00. Comparing Fig. 2 with Fig. 3, we immediately notice that the same keystream has been used to encrypt the original and the modified documents. The first 23 bytes (from the address 0xa00 to 0xa16) are exactly the same since the first 23 bytes of the two documents are the same. The rest of the bytes (from the address 0xa17 to 0xa73) are different because the plaintext changes. For example, the byte at address 0xa17 is 'f' (0x66) in the original document, but 'a' (0x61) in the modified document.

```
000a00  3E 57 FB B6 64 22 4A CA  3A 74 40 E7 1D 57 C6 DB  >W..d"J.:t@..W..
000a10  A3 88 21 53 F2 DB 3B 64  21 2A AD DD A8 7C 35 85  ..!S..;d!*...|5.
000a20  9B ED E5 F6 68 9A 35 47  68 89 9A ED 44 AE BF 08  ....h.5Gh...D...
000a30  D2 D5 CB 2B 0B 6B 45 4F  42 06 DC C6 C1 A5 81 B5  ...+.kEOB.......
000a40  AF 39 6F F1 1C 84 1F 88  B0 FD E1 09 D8 B9 E0 24  .9o............$
000a50  6C 1C 42 7C B7 D6 63 10  80 0B D5 B7 7F 01 6C 9B  l.B|..c.......l.
000a60  B8 4A F9 67 0D 27 FD 49  8E 98 76 9D C5 0F B0 E4  .J.g.'.I..v.....
000a70  AF 95 AC A2 5E 61 DD 9D  71 92 3A B9 40 AE CB F3  ....^a..q.:.@...
```

Fig. 2. Binary format of the original document (encrypted)

```
000a00  3E 57 FB B6 64 22 4A CA  3A 74 40 E7 1D 57 C6 DB  >W..d"J.:t@..W..
000a10  A3 88 21 53 F2 DB 3B 63  27 65 93 84 96 64 36 90  ..!S..;c'e...d6.
000a20  90 FA A0 EC 2D 90 24 51  6E 88 89 F0 05 A4 EF 14  ....-.$Qn.......
000a30  D6 CE DA 3B 4E 7B 0D 5E  0A 05 95 D2 DB A3 D2 B7  ...;N{.^........
000a40  E6 3D 73 F0 49 94 5E 9B  B0 EF EC 0E 94 B3 A6 6B  .=s.I.^........k
000a50  63 52 4D 77 B2 C7 69 0A  8E 45 84 A3 64 57 28 8D  cRMw..i..E..dW(.
000a60  F1 69 B0 5E 00 26 EE 55  D9 98 2F 9D C8 19 A9 F2  .i.^.&.U../.....
000a70  EC EB 82 AF 5E 61 DD 9D  71 92 3A B9 40 AE CB F3  ....^a..q.:.@...
```

Fig. 3. Binary format of the modified document (encrypted)

Obviously the same keystream byte with value 0x02 is used at address 0xa17 in the original and the modified documents.

3.2 The attack on the Microsoft Word

It is quite possible that the encrypted Microsoft Word documents would be transmitted between the different users for checking, improvements and modifications. The misuse of RC4 in Microsoft Word is thus a serious threat to those who trust the 128-bit encryption provided by Microsoft.

Once it becomes clear that RC4 is misused in Microsoft Word, the attack is straightforward. It is quite easy to detect whether the same keystream has been used for more than once. For example, if the document contains only the ASCII characters, then the most significant bit of each plaintext byte remains 0 and we can simply use those bits for detection. Once we obtained two different documents encrypted with the same keystream, a lot of information could be retrieved. The detailed analysis on recovering the information from the XORed result of two plaintexts is illustrated in [2].

4 The Misuse of RC4 in Microsoft Excel

In this section, we show that RC4 is implemented in Microsoft Excel in an insecure way. The flaw is similar to that in Microsoft Word. We use Microsoft Excel 2002 to illustrate the flaw.

4.1 Modifying the Microsoft Excel document

In this subsection, we investigate how the modification of an Excel document would affect its binary format. For the binary format of that document, when some cells get modified, the modified cells would be relocated after those unmodified cells. The modification is thus different from that in Microsoft Word.

We create an Excel document that is shown in Fig. 4. When we are creating this document, the mouse is applied to locate each cell from left to right, top to bottom (i.e., following the order '1a','1b','1c','1d','2a',$\cdots$,'3c','3d'), and there is no error correction when typing those data. We save this document, and the binary format of the saved file is shown in Fig. 5.

	A	B	C	D
1	1a	1b	1c	1d
2	2a	2b	2c	2d
3	3a	3b	3c	3d

Fig. 4. The original Excel document

```
000780  FC 00 44 00 0C 00 00 00  0C 00 00 00 02 00 00 31  ..D............1
000790  61 02 00 00 31 62 02 00  00 31 63 02 00 00 31 64  a...1b...1c...1d
0007a0  02 00 00 32 61 02 00 00  32 62 02 00 00 32 63 02  ...2a...2b...2c.
0007b0  00 00 32 64 02 00 00 33  61 02 00 00 33 62 02 00  ..2d...3a...3b..
0007c0  00 33 63 02 00 00 33 64  FF 00 12 00 08 00 8C 05  .3c...3d........
```

Fig. 5. Binary format of the original Excel document

Then we change the content '1c' to 'c1', and save the file. The modified document is shown in Fig. 6 and its binary format is shown in Fig. 7.

	A	B	C	D
1	1a	1b	c1	1d
2	2a	2b	2c	2d
3	3a	3b	3c	3d

Fig. 6. The modified Excel document

```
000780  FC 00 44 00 0C 00 00 00  0C 00 00 00 02 00 00 31  ..D............1
000790  61 02 00 00 31 62 02 00  00 31 64 02 00 00 32 61  a...1b...1d...2a
0007a0  02 00 00 32 62 02 00 00  32 63 02 00 00 32 64 02  ...2b...2c...2d.
0007b0  00 00 33 61 02 00 00 33  62 02 00 00 33 63 02 00  ..3a...3b...3c..
0007c0  00 33 64 02 00 00 63 31  FF 00 12 00 08 00 8C 05  .3d...c1........
```

Fig. 7. Binary format of the modified Excel document

Comparing Fig. 5 with Fig. 7, we notice that the data 'c1' in the modified cell is relocated to the end of the data, and those unmodified data ('1d', '2a', '2b', '2c', '2d', '3a', '3b', '3c', '3d') proceeded by the modified cell are moved forward.

4.2 Evidence that RC4 is misused in Microsoft Excel

In this subsection, we show that RC4 is misused in Microsoft Excel. We select the 'Microsoft Strong Cryptographic Provider' that supports the 128-bit RC4 encryption. After observing the encrypted versions of a number of files, we notice that the encrypted data does not start from a fixed position (different from that in Microsoft Word). Normally the encrypted data starts 31 bytes after the data 0x8c000400 and ends before the data 0xff001200.

We create an Excel document with content as that in Fig. 4 (with binary format as that in Fig. 5). We choose the 'Microsoft Strong Cryptographic Provider', set a password and the document gets encrypted. Then we save the file twice. (Saving the file once would not affect the result of the attack, but the illustration of our attack becomes complicated. The reason is that the location of the encrypted data after the first saving is different from the locations of the encrypted data after saving the file more than once.) The binary format of the encrypted file is shown in Fig. 8.

```
000e20  11 91 18 F1 5D BA A7 A4  1E FD 8C 00 04 00 F5 E1  ....]...........
000e30  EF 05 C1 01 08 00 2E 9F  0D 34 28 9A 47 68 FC 00  ..........4(.Gh..
000e40  44 00 7C 1A A6 D9 8F F9  AD F6 7B DD 92 8B 4E 49  D.|.......{...NI
000e50  DA 26 22 C0 B8 38 F8 BE  4A 98 75 26 8D C8 71 7C  .&"..8..J.u&..q|
000e60  4B 15 D4 70 1D 87 0D E1  30 28 40 9D 39 58 02 F4  K..p....0(@.9X..
000e70  68 58 2F EC 28 5D 76 7E  51 07 56 A8 3F 90 DD 3F  hX/.(]v~Q.V.?..?
000e80  D4 61 5D 6F 3C 04 FF 00  12 00 83 7E 02 B1 A9 79  .a]o<......~...y
```

Fig. 8. Binary format of the original Excel document (encrypted)

Then we modify the content '1c' to 'c1' and save the file. Its binary format is shown in Fig. 9.

```
000e20  11 91 18 F1 5D BA A7 A4  1E FD 8C 00 04 00 F5 E1  ....]...........
000e30  EF 05 C1 01 08 00 2E 9F  0D 34 28 9A 47 68 FC 00  ..........4(.Gh..
000e40  44 00 7C 1A A6 D9 8F F9  AD F6 7B DD 92 8B 4E 49  D.|.......{...NI
000e50  DA 26 22 C0 B8 38 F8 BE  4D 98 75 26 8E CD 71 7C  .&"..8..M.u&..q|
000e60  4B 15 D7 70 1D 87 0D E0  30 28 40 9D 3E 58 02 F4  K..p....0(@.>X..
000e70  69 5D 2F EC 28 5D 75 7E  51 07 56 A9 3F 90 DD 3F  i]/.(]u~Q.V.?..?
000e80  D3 61 5D 6F 6C 51 FF 00  12 00 83 7E 02 B1 A9 79  .a]olQ.....~...y
```

Fig. 9. Binary format of the modified Excel document (encrypted)

In Fig. 8 and Fig. 9, the encrypted data both start from the address 0xe4d and ends at the address 0xe85. Comparing Fig. 5 with Fig. 7, we know that the first ten bytes of the unencrypted data are the same. Comparing Fig. 8 with 9, we notice that the first ten bytes in the encrypted files are the same. It shows that the first ten bytes of the keystream being used in those two files are the same.

We demonstrate another two examples below. The data at the address 0xe57 and 0xe58 are 0xbe4a and 0xbe4d in Fig. 8 and 9, respectively. From the Fig.

5 and 7, we know that the unencrypted data there are '1c' (0x3163) and '1d' (0x3164) respectively. It becomes clear that the keystream with value 0x8f29 is used to encrypt the data at the address 0xe57 and 0xe58. The data at the address 0xe5c and 0xe5d are 0x8dc8 and 0x8ecd in Fig. 8 and 9, respectively. From the Fig. 5 and 7, we know that the unencrypted data there are '1d' (0x3164) and '2a' (0x3261) respectively. It becomes clear that the key stream with value 0xbcac is used to encrypt the data at the address 0xe5c and 0xe5d.

The examples above show that the same keystream is used to encrypt both the original and the modified Excel documents. The attack to retrieve the information from the different versions of the Excel documents is similar to that being applied to the Word document (illustrated in Subsection 3.2).

5 The Countermeasures

For the protection of documents, we assume that a user (or many users) would use one password to protect many documents and each document may be edited many times. It is the multi-user, multi-document and multi-edition environment.

When we apply a stream cipher to encrypt a document, we should ensure that a different initialization vector is generated whenever the stream cipher is invoked for encryption in these applications.

We provide here a very simple way of using stream cipher in the document protection. Whenever a stream cipher is invoked, the HMAC [1], with the password as the key and the document as the message, is applied to generate a random number (256 bits if SHA-256 [7] being used) and that random number is used as the initialization vector in the stream cipher. Note that we must use the HMAC instead of the hash; otherwise, some information of the document could be retrieved from the hash output (if the attacker happens to know most of the content of a document). This method does not require random or pseudorandom source, and it ensures that the same initialization vector would not be used for different documents. At the decryption stage, one can easily check the integrity of the document. The drawback of this scheme is that the same message is always encrypted to the same ciphertext (with the same password), and the document is accessed twice in the encryption process. However, we believe that this method would be suitable for almost all the document protection applications.

If the initialization vector is generated independent of the document, it must be generated from random (or pseudorandom) source. It is difficult to maintain a counter in the multi-user environment. We can hash the current time (as accurate as millisecond), the number of clock cycles that has passed since the operating system being started, the number of clock cycles that has passed since the document processing application program being started, and the previous initialization vector (if it exists) together to generate a new initialization vector for the stream cipher. All these parameters are used to minimize the chance that the collision of the initialization vectors could occur.

An alternative approach is to use block cipher, instead of stream cipher, for document protection. We can use some secure block cipher (such as AES [8]) in CBC mode [6] to encrypt the documents. The damage is not very severe if the same initialization vector is used for different documents.

6 Conclusion

RC4 is misused in the Microsoft Office (Word and Excel). The initialization vector remains the same when an encrypted document gets modified and saved. The consequence is that the same keystream is used to encrypt the different versions of a document and a lot of information could be retrieved from those encrypted files. If anyone has used the encryption in the Microsoft Office in the way similar to that described in this report, then it is time for him/her to assess the damage that has been caused.

The security flaw reported in this report emphasizes that using a secure cipher does not automatically guarantee that the data could be securely protected. The design and implementation of encryption software should be treated seriously.

References

1. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication", in *Advances in Cryptology - Crypto 96*, LNCS 1109, pp. 1-15, Springer-Verlag, 1996. See also RFC 2104, "HMAC: Keyed-Hashing for Message Authentication", February 1997.
2. E. Dawson and L. Nielsen. Automated cryptanalysis of XOR plaintext strings. Cryptologia, (2):165-181, April 1996.
3. S. Fluhrer, I. Mantin, and A. Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4", in *Selected Areas in Cryptography (SAC 2001)*, LNCS 2259, pp. 1-24, Springer-Verlag, 2001.
4. I. Goldber, and D. Wagner, "Randomness and the Netscape Browser". *Dr. Dobb's Journal*, January 1996, pp. 66-70.
5. Microsoft Office. At http://office.microsoft.com/home/default.aspx
6. National Institute of Standards and Technology, "DES Modes of Operation", Federal Information Processing Standards Publication (FIPS) 81. Available at http://csrc.nist.gov/publications/fips/
7. National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication (FIPS) 180-2. Available at http://csrc.nist.gov/publications/fips/
8. National Institute of Standards and Technology, "ADVANCED ENCRYPTION STANDARD (AES) ", Federal Information Processing Standards Publication (FIPS) 197. Available at http://csrc.nist.gov/publications/fips/
9. R.L. Rivest, "The RC4 Encryption Algorithm". RSA Data Security, Inc., March 12, 1992.

A Encryption in Microsoft Word 2002

To encrypt a Word document, from the menu 'Tools', choose 'Options', then 'Security'. We only set the 'Password to open'. From the 'Advanced', select the 'RC4, Microsoft Strong Cryptographic Provider'. The default key length is 128 bits. Set a password. The window for the encryption is given in Figure 10.

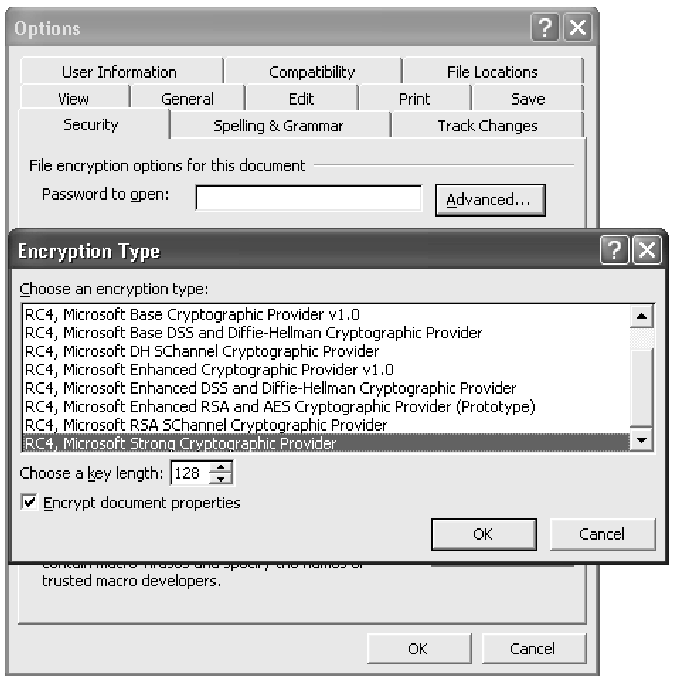

Fig. 10. Setting the Micorsoft Word Encryption