A Columnar Transposition cipher in a contemporary setting.

J Jones[1]

Abstract

A simple cryptographic method, a type of columnar transposition cipher, is described which may be used in series with other methods to provide practical hybrid encryption. The method involves the use of a deterministic Cryptographic Pseudo Random Number Generator (CPRNG) to specify an unbiased random transposition of blocks of plain-text or intermediate text. The decryption process involves applying a reverse transposition at the appropriate stage. The method may be applied at a single stage or several stages. The unit of transposition may be a bit or a byte or larger block. The method, when added in series with existing encryption methods, can deter attacks which exploit known weaknesses. The method could be applied at several scales in order to obscure structures within the plain-text, e.g. at the bit level, to obscure the almost fixed transmission headers; at the computer word-level, to disguise application record structures.

Two (incompatible) outline implementations are presented. One for use with a block cipher and one for use with a stream cipher

The specification of necessary configuration parameters required to amend or construct a software or hardware implementation is avoided in this preliminary article.

Introduction

Columnar Transposition Ciphers have been known for some time but I was unable to find any article on the combination of two simple methods in which a CPRNG is used to both provide a pseudo random stream for a one time pad stream-cipher and to specify a reversible random permutation of the cipher-text to form the code-word of a columnar transposition.

There are several methods of constructing random permutations. The method due to Fisher and Yates with improvements due to Durstenfeld [Durstenfeld_1964][Wikipedia] has a very small code footprint and is believed to be free of bias. It requires the generation of numbers

[1] Email: weprct@gmail.com

from a range that varies and there exist suggested precautions that avoid biases that might occur if a naïve implementation were used to generate random numbers for the permutation. The remainder of this article assumes such permutations are available.

Two outline implementations are presented. One is in conjunction with the block cipher AES, the other is with a stream cipher, such as WEP/RC4, with an alternative CPRNG in place of RC4. In both implementation descriptions the particular columnar transposition is applied to 128 bit blocks of text so the columnar transposition may equally be termed a permutation of symbols. The terms are synonymous in the remainder of this article.

The first implementation requires single bit manipulation which more powerful processors might find cumbersome. It is thought this variant would be particularly suitable for a hardware implementation. The alternative implementation described is suitable for processors that can manipulate bytes more nimbly than bits.

Two Example configurations

Following the initial implementation of Wired Equivalent Privacy (WEP), although few questioned the basic idea of using a deterministic CPRNG to provide a one-time pad as a means of encryption, the specification of the implementation parameters was less than ideal and meant that WEP was poorly received and rapidly superseded by WPA and WPA2. I choose configuration parameters only to simplify the description of the method and rely on the community to point out potential pit-falls in choosing them and suggest more appropriate configuration parameters.

Extending AES-256

When Random Columnar Transposition (RCT) is used with AES it is necessary to include a CPRNG initialised deterministically from the same or additional key bits. For the purpose of explanation, consider the situation of using this Random Transposition using a 256 bit key (RCT-256) in series with AES-256 on 128 bit text blocks. In order to fully exploit the RCT method with a 128 single bit columns for each cipher block a key of $log_2(128!)$ bits (slightly in excess of 700 bits) would be required. For encryption, the random transposition is applied at the bit level to the 128 bit block output by AES-256. That is, the message symbols are single bits. For decryption, the reverse transposition is applied to the enciphered text before re-applying the AES-256 procedure to obtain the original plain text. A CPRNG would have to be added to the footprint to supply random numbers for the transposition data structure.

Let the plain text be indicated by P and the output from applying AES to the plaintext by P_{AES}.

Let V_i be initialised to the ordinal sequence 0 …127

Construct V_{RCT} an unbiased random permutation of V_i based on an independent pre-shared random key used with the selected CPRNG. V_{RCT} is the 128 symbol code-word of the RCT

phase. Since the above steps must be repeated as part of the decryption this key must be included along with the AES-256 key as part of the Pre-Shared-Key (PSK'). E is the block obtained by applying AES-256 followed by RCT-256

Apply the bit-level random transposition as follows:

$$For\ i = 0\,..\,127$$

$$\{\quad E[i] = P_{AES}\left[V_{RCT}[i]\right]\}$$

On completion $E[0..127]$ is the final 128 bit output block.

The decryption is identical to the encryption apart from the final for loop.

Apply the reverse transposition as follows:

$$For\ i = 0\,..\,127$$

$$\{\quad P_{AES}\left[V_{RCT}[i]\right] = E[i]\}$$

On completion $P_{AES}[0..127]$ is the block to which AES is applied to yield the plain text.

One important feature which should be noted is that AES alone is a symmetric operation but the combination of AES and RCT is no longer symmetric. That may require particular care if an implementation was to be based on an existing application of AES.

Cipher text having extreme numbers of set bits.

At first glance it might be thought that if a crib coincided with a block of cipher-text having an unusually low or high number of set bits then there would be an unusually low number of possible blocks of input to the AES stage for decryption. If P_{AES} has x bits set to 1 then there are $\frac{128!}{(128-x)!x!}$ ways of distributing x 1-bits amongst the 128 slots in a block; this set of numbers correspond with the binomial coefficients $\binom{128}{x}$ giving strength in bits as,

$$log_2(128\,!/(128-x)!/x!)$$

This complexity is just 1 bit in the case of 128 bits set or clear and the complexity is 7 bits when only one bit is set or 127 bits are set. However, whilst this may reduce the overall complexity to that of the method partnering RCT, the situation applies only to the region of the crib and doesn't reveal which of the many permutations of the intermediate text results in the required permutation. There are $(128-x)!\,x!$ permutations resulting in the same intermediate text thus restoring the $log_2(128!)$ complexity for the remaining text. Furthermore the incorporation of AES renders intermediate text P_{AES} having blocks with a low or high number of set bits extremely rare.

Extending WEP

Using RCT in tandem with AES-256 required the use of a CPRNG in addition to the existing AES facilities and the bit manipulation is probably more suited to a hardware implementation. That combination relies on the sheer size of 128!. The next sample application considers an extension, WEP', to the WEP family and might be more suited to a byte processor. In this case the original method includes the RC4 CPRNG so the code footprint is extended only by the code to construct the substitution vector and to apply and reverse the substitution. However a fresh implementation (WEP') would use a more up to date CPRNG.

Assuming plain text is to be processed in 128 bit blocks or 16 bytes in order that modification of an existing WEP implementation may be simple.

It is assumed the permutation will be applied to the intermediate text resulting from the application of the pseudorandom one-time pad to the plain text.

Let the plain text be indicated by P and the output from applying WEP' to the plaintext by $P_{WEP.}$

Let V_i be initialised to the ordinal sequence 0 … 15

Using the CPRNG, generate V_{RCT} , an unbiased random permutation of V_i based on an independent pre-shared random key.

Apply the byte-level random permutation as follows:

$$For\ i = 0\,..\,15$$

$$\{\quad E[i] = P_{WEP}\left[V_{RCT}[i]\right]$$

$$\}$$

On completion $E[0..15]$ is the final output block.

The decryption is identical to the encryption apart from the final for loop.

Apply the reverse permutation as follows:

$$For\ i = 0\,..\,15$$

$$\{\quad P_{WEP}\left[V_{RCT}[i]\right] = E[i]\}$$

$$\}$$

On completion $P_{WEP}[0..15]$ is the block to pass through the application of the one-time pad to yield the plain text.

The code is different to the AES case above only in respect of the loop size and data component size. In terms of a column transposition cipher the code-word has just 16 values and there are 2^8 symbols in the message text. The strength in bits is given by $log_2(2^8\ 16!)$ which is just over 54 bits to be added to the one-time pad strength. This is considerably weaker than $log_2(128!)$, just over 700 bits, in the AES case above. This could be improved by simply increasing the number of elements in the permutation for example 512 bytes for a file system encryption. This would be more difficult in the case of a communication link where plain text length is variable. For the purposes of evaluating the method this strength seems reasonable but for a 'production' application the 54 bit strength may be deemed inadequate.

Both of the outline implementations given above do not specify whether the crucial permutation vector V_{RCT} is constructed at the start of each 128 bit/ 16 byte block. It would be possible to re-use a single V_{RCT} for many or all text blocks. I would anticipate the application and reversal of the permutation to be of low complexity and, particularly in the case of the WEP'/byte alternative, the extra code might be added to a wireless router software update and run on existing hardware. The fact that the inclusion of RCT renders the combination no longer symmetric may also affect the ease of modifying such software. If an implementation of RCT were to be used with a CPRNG being used by the cooperating method then access to the CPRNG would have to be carefully coordinated because the different order of access to the CPRNG would not deliver the correct bits to the individual methods. The use of two independent CPRNG streams, one for each method, would be a simple solution.

Further considerations

The methods described address only the problems of encryption/decryption. Other vital aspects, such as sharing a private, unused key and obtaining unbiased permutation vectors, are assumed to be available.

It is thought that using the RCT method to extend either block or stream ciphers would involve very modest effort. The strength can be varied by restricting the number of bits in the shared key. The RCT substitution vector described in the extension of AES-256 consists of a vector of 128 bytes. In fact such a data structure could be extended to a 256 component permutation. Further increase is possible if the byte sized elements of the substitute vector are increased in size.

The RCT method was selected to work with a WEP-like stream cipher to avoid the problem of fixed parts within transmission headers and the possibility to alter the text of messages without detection.

The only implementation of RCT known was developed by the author in the early 1990s. It used bytes as the unit of permutation in blocks of 256 bytes. The RCT method was used in

tandem with a WEP like stream cipher. The combination exhibited weaknesses resulting from a naïve use of a general purpose random number generator for the one-time pad and use of a sort package to build the permutation vector (one per file transferred). It was used for secure file transfer over the public internet using ftp. Completely new implementations would be required if the method was to be used in contemporary settings.

The author would like to involve others who have suitable experience in practical cryptography in the process of specifying and building a tool or tools for public challenges e.g. Wireless router encryption or File-system/Hard-drive encryption. In the two configurations described, a couple of configuration parameters were selected without great care and may need further discussion. Here is a list of other parameters/conditions which the author feels may require discussion.

- The RCT method is described as being used in partnership with AES or a WEP like stream cipher. Need it always be used in partnership? Would its use partnered with just a file compression be safe?
- Would the method be safe when used in a context where the key/permutation vector is long lived? (File system encryption).
- Would it be safe where the user has control of the input? Does AES in partnership with RCT render the combination safe?
- Is the order of application of the partner methods just a matter of taste or is there a snag lurking in the choice?
- Does RCT at the bit level cover all eventualities or should RCT application be done at multiple scales in some circumstances?
- Could the combined cost of AES and RCT be safely be reduced by weakening the AES component without significantly degrading overall complexity?

Bibliography

Federal Information Processing Standards Publication 197 Advanced Encryption Standard(AES) http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf Retrieved Dec 2015.

Durstenfeld, R. (July 1964). "Algorithm 235: Random permutation".*Communications of the ACM* 7 (7): 420.doi:10.1145/364520.364540

IEEE Standard 802.11* 1997 – 2008 includes specification of original WEP. Now deprecated.

Kiselyov, O.(Sep 2001). http://okmij.org/ftp/Haskell/perfect-shuffle.txt. Retrieved Dec 2015.

Wikipedia http://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle. Retrieved Dec 2015.

Wikipedia http://en.wikipedia.org/wiki/Transposition_cipher Retrieved Dec 2015