Global Information Assurance Certification Paper

A Review of Chaffing and Winnowing

David Spence

GSEC v1.4b

Abstract

This paper presents an overview of Chaffing and Winnowing as described by Ronald Rivest. This leads onto a review of a secure Chaffing and Winnowing scheme called Chaffinch.

Introduction

Chaffing and Winnowing was first proposed by Ronald Rivest, [1], as a means to achieve confidentiality in message transmission.

At the present time there were two major techniques used for preventing adversaries from gaining information from a transmitted message:

- **Encryption**

  This is the process of transforming the message into a random stream of characters called a cipher text. This is done using keys to encrypt and decrypt the message. Decryption of the cipher text is very difficult without knowledge these keys. Techniques like this have been around for some time and commonly used examples are DES, 3DES, RSA and AES.

- **Steganography**

  The art of hiding a secret message within a larger one in such a way as to be able to deny the message exists. An example is hiding a text message in a picture file by changing the low-order pixel bits to be the message information.

Chaffing and Winnowing introduces a novel new concept that does not use encryption keys, and as such would not be subject to import and export restrictions. Chaffing and Winnowing achieves privacy and confidentiality by using authentication keys, however, these are not to be confused with encryption keys. Authentication keys/digital signatures are not controlled by governments and most have chosen that the disclosure of these signatures is not allowed. They have taken this stance over authentication keys because of the danger of unscrupulous people being able to use someone else's personal authenticator to take over that person's identity!

This paper goes over Rivest's original paper, [1], and discusses more recent work in the area of Chaffing and Winnowing.

In the second half we look at a secure instance of Chaffing and Winnowing called Chaffinch and examine the techniques used.

1) Chaffing and Winnowing

To understand the processes involved it is first useful to familiarize us with some quite old words.

> Winnow – to separate out or eliminate the poor or useless parts
> Chaff – useless parts of wheat

Winnowing is often used when referring to separating grain from chaff.

Authenticating

When the user has a message they want to send it is broken into packets. These packets contain the message information and header information. Within this header is usually a serial number so that the receiver can reassemble the message in the correct order.
In Chaffing and Winnowing the person sending the message adds a "message authentication code", MAC, to each of the transmitted packets.
Both the sender and receiver calculate the Mac as a function of the packet contents, serial number and a secret password/key that is shared.
This MAC is attached onto the end of the packet as demonstrated in Figure 1.

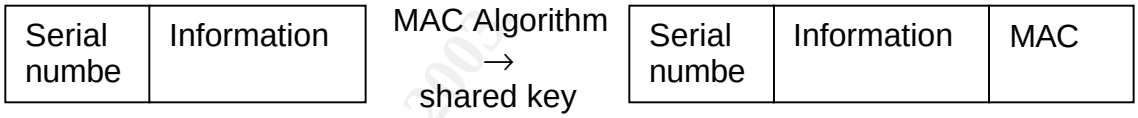

Figure 1. This shows the process of authenticating packets. The MAC is calculated and then put onto the end of the packet. These MAC's are not regarded as encryption, just authentication, as the packet is still in the clear.

Now that all of the packets are authenticated they are ready to be sent. If they are sent as they are there is no security as the information is still in the clear! An adversary need only intercept all of the packets to reconstruct the message. Confidentiality comes from the next step…

Chaffing

This is the process of "adding chaff", useless parts, to the transmitted message. The chaff are fake packets that have the correct overall format, reasonable serial numbers and reasonable content, however, they have MAC's that are not valid when computed with the shared key.

These chaff packets are interspersed randomly with the good(wheat) packets to form the transmitted data sequence. The receiver collects all of the transmitted packets in the sequence and computes the MAC that should be associated with each packet using the MAC algorithm and the shared key. Those packets with MAC's not matching those appended are discarded and the only packets left are the wheat ones with valid MAC's. The MAC numbers are stripped off and the serial numbers used to reconstruct the message.

This technique is called Clear-text Chaffing and Winnowing.
An example of the whole process is given in Figure 2.

Message	**Authentication using MAC and secret key**		
	Serial number	data	MAC
Hi Fran	1	Hi Fran	33cm0o3
Our meeting is	2	Our meeting is	8734fj3
In the usual place	3	In the usual place	384fjfj
At 7pm	4	At 7pm	wsl28ff
Dave	5	Dave	202jdfi

Transmitted Packets			**MAC checking by the receiver**	
1	Hi Fran	33cm0o3	1	valid
1	**Hi Fran**	**u3idjak**	**1**	**invalid**
2	Our meeting is	8734fj3	2	valid
2	**I'll meet you**	**73jdlaj**	**2**	**invalid**
3	**In the high street**	**hw629du**	**3**	**invalid**
3	In the usual place	384fjfj	3	valid
4	**At 6pm**	**6duajkc**	**4**	**invalid**
4	At 7pm	wsl28ff	4	valid
5	Dave	202jdfi	5	valid
5	**Charles**	**5ejqp98**	**5**	**invalid**

Message Reconstruction		**Message**
Serial number	data	
1	Hi Fran	Hi Fran
2	Our meeting is	Our meeting is
3	In the usual place	In the usual place
4	At 7pm	At 7pm
5	Dave	Dave

Figure 2. Here we see the Chaffing and Winnowing process in all its steps. In the example above the chaff packets are in bold. These are added on the sender side and discarded by the receiver.

Discussion of Clear-text Chaffing and Winnowing

The main security issues of this approach are that the information is still sent in the clear and possible messages are easy to work out by combining all of the information by serial number.
A good Chaffing and Winnowing scheme will add one chaff packet for every wheat serial number and the information of the chaff packets must make sense. If your message is in French and the chaff was random gibberish then it would be easy to reconstruct the message.

General security issues of the Chaffing and Winnowing MAC approach are that the MAC algorithm must be strong and act as a "random function" to the adversary. As such the adversary would not be able to distinguish wheat from chaff unless they know the shared key!

Of course brute force approaches to find the shared key will also work but given a complex enough key this approach will be lengthy.

Bit-by-Bit Chaffing and Winnowing

Another approach presented is to divide the entire message into single bits and transmit these packets with serial number and MAC.
To create the chaff the serial numbers from the wheat are added to the chaff and the other bit value is used as the message portion along with an invalid MAC. An example of a wheat and chaff packet can be found in Figure 3.

	Serial No.	Data	MAC
Wheat	114	0	u329ewfj
Chaff	114	1	j320894j

Figure 3. Here is an example of a wheat and chaff packet as produced in the bit-by-bit scheme. In this scheme every wheat packet has a corresponding chaff packet for security. The receiver checks the MAC and discards the chaff.

Discussion

This bit-by-bit scheme offers a huge advance in confidentiality over the previous approach and has been proven to meet a definition of privacy from plain-text attack [2].

It is an interesting point here that the creation of the chaff does not require knowledge of the secret key! Therefore it is feasible that two people who merely authenticate their packets and do nothing else could be using Chaffing and

Winnowing. How? Well if someone else at each end were to look at the packets to be transmitted and add reasonable chaff then confidentiality is achieved without the sender and receivers knowledge. The receiver discards the chaff as it does not have a valid MAC.

This bit-by-bit method is highly inefficient, as, putting some sizes to the parts of the packet, serial number 32 bits, MAC 64 bits, data one bit and possible extra header information 32 bits. With these sizes each one bit of a message will require a packet greater than 100bits! If I were sending a 1kb message it would require more than 8000 packets and be over 100kb in size. Also, every wheat packet needs a chaff packet, therefore a 1kb message becomes over 200kb for transmission!

A much more efficient regime was proposed in [1] and is summarized in the next section.

Package Transform

Rivest suggested that using an "All-or nothing" (AONT) or "package" transform as described in [3] would greatly increase efficiency whilst maintaining confidentiality.

This procedure works by taking the plaintext and producing a packaged message that effectively looks like random noise. This randomized packaged message is then split into packets and the packets authenticated. This packaging procedure has the interesting property that the original message cannot be reproduced unless the receiver has all of the message segments. **It is important to stress that this operation can be undone by anyone who receives all of the packets.** However, the chaffing and winnowing regime now provides a high level of confidentiality because the adversaries ability to distinguish the genuine message packets from the chaff is negligible and they now need every wheat packet to generate the original message.

This transform works as follows: consider m_1, m_2, …, m_s plaintext blocks, H a hash function and K' a randomly chosen key

The transmitted blocks are:

$$m_i' = m_i \oplus H(K', i) \text{ for } i = 1, 2, \ldots s$$

K' is transmitted by sending the extra value M:

$$M = K' \oplus h_1 \oplus h_2 \oplus \ldots \oplus h_s$$

Where:

$$h_i = H(K_0, m_i' \oplus i) \text{ for } i = 1, 2, \ldots s$$

And K_0 is a publicly known key.

The mechanism represented by ⊕ is called XORing and is the process of taking an eXclusive OR of all of the bits of two arguments in order. The process means that if both bits are 1 then the result is zero, if one of them is 1 then the result is 1 and if both are 0 then it is 0. For example, if the two bit streams to be XORed are 1010 and 1100 then the result is 0110.

The receiver has all m' values and M and can work out the original message by computing hi and XORing them together with M to recover K'. Once they have K' this can be used with mi' to recover mi.

The packaged message is then appended with MAC's and chaff added with random contents and invalid MAC's as before. Again no encryption has been performed, only a pre-processing step. The random key is sent within the data stream if you know where to look. The adversary not being able to tell wheat from chaff generates the confidentiality. However, now they need every single packet of wheat to reconstruct any part of the message and the transmitted information is now randomized.

Discussion

It would now seem to be a confidential and secure means of communication has been achieved. However, [2] and [4] have found some security issues with the "package" transform as used here and in [2] they go on to prove that using a different transform as the AONT is secure. However, this introduces encryption into Chaffing and Winnowing and some would see this as undesirable. Other ciphers exist that are proposed as having an All-Or-Nothing property, notably BEAR and LION [5].

Another interesting point to consider here are the chaff packets. These can be randomly generated, however, in [1] Rivest suggests it would be possible to multiplex two streams of wheat from different sources, one acts as chaff for the other. The two groups of people communicating would only receive the messages that are destined for them. This arises because the people receiving the packets compute the MAC's and only keep the ones for which their secret key has generated the same MAC's. This in turn generates an interesting consequence that if the people communicating are forced to reveal their messages they can reveal one of them and the other would remain buried in the 'chaff'. This is similar to the technique of "deniable encryption" [6] proposed by Canetti et al. This is discussed further in the next section.

Proof of Concept

At the present time there are only two proof of concept Chaffing and Winnowing programs available [7] and [8]. These are written in PERL and JAVA and are only proof of the Chaffing and Winnowing scheme, they do not use an AONT as a pre-processing step.

2) Chaffinch [9]

This is a system proposed in [4] and based on the original Chaffing and Winnowing scheme but introduces several new ideas to improve security and to also allow the passing of concurrent messages.
In their construction they attempt to use no keys so that the scheme cannot be viewed as encryption.

Design of Chaffinch

Chaffinch is designed in such a way as to be able to send multiple messages within a single communication.
An innocuous cover message is always sent, which may or may not be accompanied by other messages. In the face of legal threats the cover message is always revealed first, it would be "plausibly deniable" that any other messages exist!

The method of sending messages is the same, the message is split into sections and then each section is signed with a secret key so that the receiver can identify them. The chaff can be random material, OR it can be composed of further messages signed using a different secret key. The resulting sections act as chaff for the cover message and the cover message acts as chaff for them.
So can we remove chaff altogether? The answer is no as otherwise revealing the next to last message would also reveal the last. We can however reduce the number of random chaff packets needed and so reduce bandwidth requirements.
Figure 4 shows a small part of the Chaffinch communication to give an idea of how it works. Note that the message sections are always in the correct order, but the way in which they are interleaved is random, i.e. the first section of message two occurs before the first of message one.

Message 2 Section 1	Message 2 Authenticator
Message 1 Section 1	Message 1 Authenticator
Random	Random
Message 3 Section 1	Message 3 Authenticator
Random	Random
Message 1 Section 2	Message 1 Authenticator
Message 2 Section 2	Message 2 Authenticator
Message 3 Section 2	Message 3 Authenticator
Random	Random
Message 1 Section 3	Message 1 Authenticator

Figure 4. View of part of a Chaffinch block

Differences from Chaffing and Winnowing

There are two major differences from the original Chaffing and Winnowing scheme:

- **Use of a different authenticator on the packets**, i.e. no MAC

They use a different authenticator to try to reduce the bandwidth needed for message transmission. In [1] Rivest suggests using a 64 bit MAC for security, in Chaffinch they use the agreed secret authentication key to prime a stream cipher along with a random session value that is used for all the messages being sent at the same time and is sent to the receiver in the clear. They propose that 10 bits of the cipher stream is enough for this approach to work securely. The receiver checks the stream cipher and discards those packets that don't match the expected sequence.

Although now they identify another problem, as there are only 10 bits of authenticator there is a much higher probability of a chaff packet being mistaken for a valid authenticator. Using an All-Or-Nothing transform along with this 10-bit authenticator would mean that not all of the messages would get through as you need every packet in order to produce the message. Whilst proposing a solution in the paper to stop chaff being identified as part of the message they decide not to use this. Instead they let the recipient do a brute force search of all the possible combinations of valid message segments, and the correct arrangement is detected when a valid message is generated.

- **Use of an alternate pre-processing step**

In Chaffinch they do not use a "package transform" as the AONT for the pre-processing step as they identify a weakness of Rivest's proposal. If an attacker is conducting a brute force search of chaff and wheat packets then the package transform does not generate as much work as expected because the values of h_i, see page 5, can be calculated once and then reused for any trail arrangement that incorporates the same section of the message in the same position. In [5] they propose a solution to this to make sure that the attacker has more work. This modification is to hi:

$$h_i = H(K_0, m_i' \oplus Z) \text{ for } i = 1, 2, \ldots s$$

Where:

$$Z = HASH(m_1', m_2', \ldots m_s')$$

HASH can be any 'secure' hash function like SHA-1 or MD5.

Although this method could be used they do not go on to use it in Chaffinch, instead they opt to use a secure block cipher called BEAR [5]. BEAR takes the whole message and turns it into a sea of random bits like the "package"

transform, however there is no key sent to the receiver in BEAR as it is used in a keyless manner (in “package” transform a key, K’, is transmitted in M, page5).

Discussion

Using BEAR in a keyless manner means that anyone that receives all of the message segments can reconstruct the message, confidentiality is achieved by the difficulty in an adversary picking out the correct message segments in a communication stream. Another benefit of using BEAR in this way is that if the communicating parties are taken to court the whole packaging process is keyless. Court scrutiny of the “package” transform as a non-encrypting step may be clouded due to the fact that a key is required even if it is sent with the message (if you know where to look).

The mechanisms behind Chaffinch described in the paper provide a way for two communicating parties to have a stream of communication that would be plausibly deniable. The use of an innocuous cover message means that although there may be further secret messages in the stream the users can deny they exist. So long as the parties are using secure algorithms that have a good degree of psuedo-randomness and are willing to lie then it would be incredibly difficult to prove any further communication happened.

If Chaffinch users were caught and actually gave up all of their message keys to the authorities in an attempt to show that they had nothing to hide their actions could be in vain if they are not careful. Giving up all their messages would reveal the added random chaff packets. Even though the users know they have no messages left the authorities could, quite rightly, think they had been lied to. This seems like a bad situation to be in if you are the users so it may be desirable for users to be able to prove that there are no more messages by using a seed or traceable mechanism to generate the chaff. Upon questioning they could also give up this seed so as to clear them from further suspicion.

As Chaffinch provides plausible deniability for communication if a user comes under suspicion they are likely to have their computer seized and a detailed analysis of the hard drives conducted. If there are more messages found on the disks than have been declared by the parties involved then they could be forced to reveal the existence of the extra messages.
To provide system wide deniability it would be essential to have no easy way for files to be found on a computer. This could be achieved by using a steganographic file system like StegFS[10]. This file system provides several levels of plausible deniability with each level password protected.

Conclusion

This paper has gone through the implementation of an idea and presented a secure and confidential communication scheme based on Chaffing and Winnowing.

This scheme can handle many messages concurrently and has the advantage of not using any encryption.

Although this is also debatable as it depends on the definition of encryption that you use. In [2] they use the term “encryption scheme” to define “any mechanism whose goal is to provide privacy”. Clearly under this definition the key for the MAC is the decryption key even though its use here is in providing authentication. The same arguments would appear to follow in the case of Chaffinch.

However, governments seem to have a different way of viewing things. For the regulation of encryption, governments have decided to restrict the mechanisms, i.e. the algorithms, used and put no restrictions on other processes that achieve the same goal as encryption. Under the rules by which we are governed Chaffinch users would be safe from giving up their keys (for now anyway).

The discussion of whether Chaffing and Winnowing constitutes encryption will probably be argued for some time to come in the academic World whilst being ignored in the political. This will probably continue until such a time that it will probably only be settled by the first big court case. This in turn may even change the way governments define encryption in their policies.

After all, if known terrorists were accused of using Chaffinch to plot an assassination, and you were on the jury, what would you think?

References

[1] Rivest, Ronald. ”Chaffing and Winnowing: Confidentiality without Encryption”, URL http://theory.lcs.mit.edu/~rivest/chaffing.txt

[2] Bellware, Mihir and Boldyreva, Alexandra. “The Security of Chaffing and Winnowing”, ASIACRYPT 2000, 2000: 517-530.

[3] Rivest, Ronald. ”All or Nothing Encryption and the Package Transform”, Fast Software Encryption 1997, Lecture Notes in Computer Science Vol. 1267, Springer-Verlag, 1997: 210-218.

[4] Clayton, Richard and Danezis, George. “Chaffinch: Confidentiality in the Face of Legal Threats”, Lecture Notes in Computer Science Vol. 2578, Springer-Verlag 2002: 70-87.

[5] Anderson, Ross and Biham, Eli. “Two Practical and Provably Secure Block Ciphers: BEAR and LION”, Fast Software Encryption (proceedings Third International Workshop), Springer-Verlag 1996: 114-120.

[6] Canetti, Ran, Cynthia Dwork, Moni Noar, and Rafail Ostrovsky. “Deniable Encryption”, Proceedings CRYPTO 1997, Springer 1997: 90-104.

[7] Fogel, Karl and Sussman, Mike. “Chaffing and Winnowing”, Chaffwin, URL http://www.red-bean.com/~sussman/downloads/chaffwin.tar.gz

[8] Annis, William. Chaffe. “Chaffing and Winnowing”, 25 June 1998 URL http://www.biostat.wisc.edu/~annis/creations/Chaffe.html

[9] Danezis, George. “Chaffinch”, 4 October 2002 URL http://www.cl.cam.ac.uk/~gd216/chaffinchHome.html

[10] McDonald, Andrew. “StegFS-a Steganographic File system for Linux”. 5 October 2002 URL http://www.mcdonald.org.uk/StegFS/

Last Updated: July 28th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Indianapolis SEC401	Indianapolis, IN	Oct 09, 2017 - Oct 14, 2017	Community SANS