Non-symbolic fragmentation

H. Ashman, H. Coupe, P. Smith, M. Neville-Smith and M. Gilbert,

***Abstract*--This paper reports on the use of non-symbolic fragmentation of data for securing communications. Non-symbolic fragmentation, or NSF, relies on breaking up data into non-symbolic fragments, which are (usually irregularly-sized) chunks whose boundaries do not necessarily coincide with the boundaries of the symbols making up the data. For example, ASCII data is broken up into fragments which may include 8-bit fragments but also include many other sized fragments. Fragments are then separated with a form of path diversity. The secrecy of the transmission relies on the secrecy of one or more of a number of things: the ordering of the fragments, the sizes of the fragments, and the use of path diversity. Once NSF is in place, it can help secure many forms of communication, and is useful for exchanging sensitive information, and for commercial transactions. A sample implementation is described with an evaluation of the technology.**

INTRODUCTION

Low customer confidence in the security and privacy of Web-based transactions is hindering the development of e-commerce. Internet-based fraud is a high-profile crime that receives much attention from the media, and creates an atmosphere of distrust of the new technology.

Non-symbolic fragmentation offers an additional way of securing transactions that can work either in conjunction with or independently of existing security mechanisms, including encryption. It has two primary characteristics – it firstly breaks up data into a set of partial files, with no recognisable characters in any place, and secondly that those partial files are transmitted in different ways, over different networks, at different times. The likelihood of a criminal being able to intercept all the information needed to retrieve a hidden message is much reduced, and they face the additional difficulty of reorganising the fragments into their correct positions to form the original message.

Similar principles are already in use in commercial transactions with sites such as `amazon.co.uk` offering the option of transmitting part of a credit card number via telephone and part via the Internet. Non-symbolic fragmentation extends this principle, as well as implementing what is essentially a bit-level transposition cipher that destroys symbol integrity. Separate transmission of parts of the message mean that signal processing techniques cannot be used to recreate the file.

Used in conjunction with encryption, non-symbolic fragmentation provides additional industrial-strength security mechanisms that should enhance not just the security of commercial transactions, but the perception of the security of transactions.

The next section describes the principles of non-symbolic fragmentation, and the implementation of a fragmentation process is in the following section. The next section describes the evaluations of the process and its security and the final section discusses related work.

NON-SYMBOLIC FRAGMENTATION

Non-symbolic fragmentation (NSF) is a means for processing data prior to transmission or storage in a way that a) makes it hard for any eavesdropper to intercept all parts of the transmission and b) even if in possession of all parts of the transmission, makes it very difficult to splice the parts together correctly to reconstitute the original.

The fragmentation process is essentially the breaking up of a transmission into small fragments and then ensuring that fragments are disordered in some way. NSF creates fragments whose sizes are usually different to the base symbol size, and the subsequent separation of these fragments so that fragments which were adjacent in the original file are (usually) not adjacent in the fragment streams.

Fragments are non-symbolic, i.e. the length of fragments is not usually the same as the length of the base symbol of the data. For example an email in 8-bit ASCII could be fragmented into bit strings between 1 and 14 bits in length, or a Unicode file could be fragmented into bit strings between 1 and 30 bits. This work to date has relied on the purely sub-symbolic fragments (e.g. for ASCII between 1 and 7 bit fragments) where no fragment contains an entire ASCII symbol.

By varying the size of fragments according to a secret key shared by the sender and receiver, a cryptanalyst ought not know even where the boundaries of any individual fragment occur. Varying fragment sizes also means that known-plaintext attacks are made greatly more difficult, while brute-force attacks are greatly increased in complexity.

In the implementation reported in this paper, we have dispersed fragments across a number of streams so that fragments adjacent in the plaintext are no longer (necessarily) adjacent in the fragmented data streams. The choice of which stream to allocate a given fragment was either cyclic, or governed by a shared secret key.

The fragment streams themselves are then transmitted individually, preferably with some form of network or path

H. Ashman is with the School of Computer Science and Information Technology, University of Nottingham, U.K. (e-mail: hla@cs.nott.ac.uk).

H. Coupe is with the School of Computer Science and Information Technology, University of Nottingham, U.K. (e-mail: hdc@cs.nott.ac.uk).

M. Neville-Smith is with Amino Communications Ltd, Cambridge, U.K. (e-mail: mnsmith@aminocom.com)

P. Smith is with Amino Communications Ltd, Cambridge, U.K. (e-mail: psmith@aminocom.com)

M. Gilbert is with Amino Communications Ltd, Cambridge, U.K. (e-mail: mgilbert@aminocom.com)

ICITA2002 ISBN: 1-86467-114-9

diversity. This involves sending fragment streams over different paths in a network or over different networks, making it harder to intercept the entire fragment set. Path diversity is also known as *spread spectrum* (see for example [1] or [4][1]).

It is still possible that all streams could be intercepted, for example when the eavesdropper is able to intercept all transmissions going into or out of a firewall host. One possible solution is the aliasing of receiving and sending hosts. Each sending and receiving host has a number of different IP addresses (possibly dynamically allocated) and each stream would be sent using a different name for that host. This makes it more difficult for an eavesdropper to recognise what transmissions are parts of the same message, as the sending and receiving host machines appear to be different for each fragment stream [3].

Rerouting from trusted intermediate machines can further disguise the destination and origin hosts. For example, one stream is sent to an intermediate host which subsequently forwards it to the destination, and another stream is sent directly to the destination. At the sending end, the streams are apparently addressed to completely different hosts, and at the receiving end, they similarly appear to come from two different hosts. It is still not impossible for an eavesdropper to recognise which streams are from the same message, but it adds to the complexity of doing this.

There can also be a temporal discrepancy in transmission of the streams, so that one arrives immediately and another is delayed for many days.

But even in the worse case when an eavesdropper knowingly and successfully intercepts all fragment streams, a good NSF implementation will leave them with the problem or reconstructing the message from the fragment streams. The eavesdropper has (at least) two major problems:

- they will not know which of the intercepted fragments are contiguous in the original data (i.e. how to correctly arrange the fragments);
- they will find that boundaries of individual fragments will be difficult to detect (i.e. how to identify individual fragments).

The use of non-symbolic fragmentation means that no whole symbol should be identifiable from the fragments, which means that standard frequency analysis attacks and other heuristic attacks which rely on content understanding are much less likely to be successful. Of course, it is possible to perform a spectral analysis based on bit patterns rather than whole letters, but in general, any attack which relies on detecting characteristics in the data will be weakened by non-symbolic fragmentation whose purpose is to destroy any whole characters present in a data file.

Whole symbols are probably (but not necessarily) present in any fragment whose size is greater than the smallest base unit of data used by the communications network or any participating applications, however a cryptanalyst will encounter difficulty in detecting at what point in the fragment the whole symbol occurs (if it does at all).

The most likely forms of attack are going to be:

- intercepting one or some of the streams, but not all streams
- intercepting all streams

The evaluation takes these possible attacks into account.

IMPLEMENTATION

A. NSF process

The following is one example of an NSF process. There are many variations possible and these are currently being implemented and tested too.

This version of the NSF process is a transposition cipher whose contents are distributed into a number of distinct fragment streams as described in the first section. While each individual stream preserves the priority of fragments, there will be missing fragments from every stream and any eavesdropper who successfully intercepts the entire set of streams will be faced with the task of splicing together the streams correctly. This is dependent upon knowing the secret key of the fragment sizing algorithm and the secret key of the stream allocation algorithm. In this implementation, both algorithms are pseudorandom number generators. However, in general, the secrecy afforded by the trialled path assignment mechanisms is far from ideal, and a much better level of security will arise from disordering the fragments within each stream. The only advantage of the mechanisms tested is that it is possible to apply them to stream data.

The NSF process creates variably-sized fragments, between 1 and 7 bits. Fragment sizes are chosen by a pseudorandom number generator (PRNG) whose seed is the first shared secret. Each fragment is subsequently assigned to one of n fragment streams, with the assignment to streams also being governed by a PRNG whose seed is the second secret key.

Each fragment stream is then transmitted using network or path diversity techniques to secure its contents. The fragment sizes are secret, as are the assignment to fragments to streams.

The first secret key makes it possible for the receiver and sender to generate the same lengths of all of the fragments. The second secret key enables both parties to reclaim each fragment from the correct fragment stream.

We have implemented the NSF algorithm with two PRNGs so far. The first PRNG was Knuth's and the second was the standard PRNG available with the C++ library of functions, and found a small but measurable difference between the two. This may or may not have been due to the slightly different path allocation – the first fragmenter used the PRNG to determine which path to allocate, the second merely cycled through the available paths. Since neither of these PRNGs is particularly robust for cryptographic purposes, we are currently experimenting with the Yarrow PRNG [2].

B. Use within communications

The implementation is being tested as a default setting for all communications via TCP/IP between participating

[1] Alternatively, see the online citation index at researchindex.com, searching on "spread spectrum" which results in over 2000 entries on this topic.

machines running Linux RedHat 7.1 or Windows 2000. All communications programs, ftp, telnet, email, Web software, operate as normal, the fragmentation process being transparent to both software and users. The NSF process runs at a lower level than the application layer so existing communications software operates in this secure fragmented mode without any alteration.

The advantage of this is that all communications are passed through the fragmentation process, regardless of their purpose. Both participating hosts need to run the fragmentation software but no other changes should be necessary.

Keys governing the PRNGs for fragment sizing and path allocation can be exchanged either using handshake protocols similar to SSL or else a key-exchange protocol like Diffie-Hellman.

Higher-level implementations are possible, for example a Web server could supply an embedded fragmenter as a Java applet in an otherwise unsecured XML page. The fragmented communications could be established in a similar way as is done with existing SSL-based communications, with handshake protocols to establish pseudorandom number generator seeds (equivalent to session keys). Of course it would also be possible for SSL-based communications to directly incorporate fragmentation as an additional security measure.

EVALUATION

We performed a number of different evaluations, based on

whether some streams but not all had been intercepted – is it possible to use either signal processing techniques or heuristic approaches to reconstruct the original data?

Whether all streams had been intercepted – how easy would it be for an eavesdropper to correctly splice the streams together to reconstruct the true file.

All of the evaluations here are of uncompressed and non-encrypted data. This is to give an accurate picture of what effect the fragmentation process has.

It is of course possible to encrypt the data either before or after fragmentation as the encryption and fragmentation operations are functionally orthogonal (i.e. they don't interfere with each other). Similarly compression can disguise the original signal further.

C. *Some but not all streams intercepted*

1) *Signal processing*

The primary evaluation performed here was to consider whether signal processing processes could reconstitute a full file from the intercepted partial file.

Shannon's sampling theorem states that given a shortest wavelength in the original signal, we need to sample at least twice per smallest period in order to be able to reconstruct the signal fully.

In a digital signal, the shortest possible wavelength is 2 bits (for every bit different to its neighbour, hence oscillating). Any signal comprising in part this shortest wavelength needs 2 samples per wavelength, i.e. 2 samples per 2 bits, which is essentially the whole signal sampled.

However not all signals (i.e. files) are partly constructed of 2-bit or other very short waves. For signals with no such very short wavelengths, it would be possible that half or less of the original bits would suffice to recreate the original signal. That is, if the fragments are transmitted in two equal-sized streams, it may be possible to reconstruct the original signal.

This should not work in the NSF process for two reasons:

1) lack of data (in most cases) – each transmission can be analysed to determine its shortest wavelength. This in turn determines a minimum number of paths required over which the signal should be broken up. For example, for 2 or 3-bit shortest wavelengths, 2 paths is enough.

However other constraints make it desirable to allow an uneven allocation of fragments to paths, so that some paths could be much larger than others. For example, an employee working from a hotel room while travelling may send most data through a fast Internet link but some data through a slower mobile phone link. Again, an analysis of the shortest wavelength of transmissions can determine what maximum proportion of a transmission can be sent securely through any single link.

2) random sampling – the random nature of the "sampling", i.e. the path allocation and fragment sizing, means that Shannon's sampling theory does not apply. The sampling theorem assumes that samples are taken at regular (i.e. predictable) intervals, and hence the set of sine waves that comprise the signal can be reconstructed via interpolation. However, if the intervals of the samples changes, the positions of the samples on the time axis are not known and it is impossible to interpolate to reconstruct the set of sine waves.

This latter observation demonstrates the critical nature of the pseudorandom fragment sizing and path allocation. Any partial file intercepted will consist of randomly-sized fragments from random parts of the original file. While individual fragments may contain small amounts of coherent samples, this is not enough to reconstruct the entire file.

2) *Heuristic attacks*

We have considered but not yet evaluated the case where heuristic attack may assist in the reconstruction of part of the signal. This is maximised if there is a relatively even spread of missing fragments within a partial file. It relies on some knowledge about the file type (which is easy enough to determine, see the next section). For example, if a file is an email, the predominant symbols will be the 80 or so keyboard characters.

Fragments will begin and/or end partway through a symbol. If the normal range of symbols is known or can be guessed, the remainder of fragment ends can be extended to include guessed values for the symbols containing the fragment end.

Additionally knowing the file type can help reconstruct at least some parts of the file, especially if the file contains regularly-spaced markers (as does MPEG) or fixed-format headers (as do emails). With enough information, especially about positions of fragments, it would be possible to reconstruct the signal by signal processing means.

D. *All streams intercepted*

In this latter case, we assume that all the partial messages have been intercepted and that the only remaining problem is to put them back together so as to create the original message.

As described in the previous section, the sampling theory does not work because of the random sampling. However there are other characteristics that may be exploited to reconstruct the transmission, especially if all parts of the transmission are intercepted.

We experimented with the spectral analyses of fragmented files. The experiments described here were intended to measure what characteristics remained in the data.

It should be emphasised that a major part of the protection afforded by the non-symbolic diversity cipher was intended originally to come from the diffusion of the paths, so that no eavesdropper can intercept all parts, and from the random sampling which prevents interpolation.

However the fragmentation process itself introduces benefits that have comparable effects to those of strong secret-key ciphers. NSF introduces a substantial measure of "randomness" so that even if all parts of the transmission were intercepted, characteristics in the data were suppressed and illicit recovery hindered.

While we were not intending to make NSF processes more "random" than secret-key ciphers, they do manifest this randomising effect and it will be one goal in future work to discover the maximum randomising effect possible with this form of transposition.

1) n-*gram spectral analyses*

This process is the *n-gram spectral analysis*, or NSA, which simply counts the frequency of each possible bit string. It tables all possible *n*-grams within the data, for example when counting 3-grams, the first three bits were checked, then the 2^{nd} to 4^{th} bits, then the 3^{rd} to 5^{th} bits and so on. It generates a table of relative frequencies of bit strings, including zero counts. The closer an *n*-gram spectral analysis appears to a constant function where the constant is the data's mean, the closer the data is to random. In random data, every bit string would be equally likely and would occur around the same number of times.

NSA is similar in principle to counting characters and character *n*-grams as is done with frequency analysis of stream ciphers. However it differs in that it operates on the bit level, a modification which was necessary because of the non-symbolic fragmentation.

2) *Results*

a) *Different fragmentation and path allocation policies*

The first thing we noted is that the two different fragmented files for each file (generated by different PRNGs) seemed to be virtually identical. This initially suggests that the choice of PRNG used to determine fragment sizes does not make much of a difference, but we intend to reserve judgement until we have performed the same tests over fragmentation with different PRNGs, in particular, an industrial-strength one such as Yarrow.

One counterintuitive result was that the two different fragmentation algorithms did not show a difference due to their path allocation. The second variant did not use a PRNG to assign fragments to a stream but instead cycled through the available paths. Hence the PRNG variant would have a one in three chance that any two fragments adjacent in the plaintext would also be adjacent in the fragmented file. We expected that this would imply a smaller randomising effect in the first variant but the results did not bear this out. The difference in relative standard deviation between the two was nominal, usually less than a few percent and in some cases the cyclic variant showed the larger relative standard deviation.

This suggests that it should be reasonably safe to use a PRNG to determine fragment assignment to files despite the one-in-*n* chance of adjacency for allocation to *n* paths, and that it would not make it any easier to reconstruct intercepted data into the original. On the other hand, cycling through the available paths more effectively balances the load between paths as well as not requiring a secret key to seed the PRNG, but gives the attacker an easier job in reconstructing the original file from fragments.

b) *Randomising effect*

We also found that the fragmentation had a "flattening" effect on the *n*-gram spectra. While we had expected some distortion of the bit strings due to the sub-symbolic fragmentation breaking up whole symbols, but a quite strong randomising effect emerged as a beneficial side effect.

In the plaintext files there had been a wide variation in the counts for individual *n*-grams, but the fragmented data featured a much smaller variation of counts. Since truly random data would have an NSA approaching a constant graph centred around the data's mean, this tendency of NSF files toward the horizontal demonstrates that they are less "featured" than plaintext files, i.e. that fragmentation introduces a measure of "randomness".

The mean value for files was essentially the same before and after fragmentation, as would be expected for a transposition-style cipher (since the number of 0s and 1s remains the same). However the relative standard deviation of post-fragmentation files was reduced by wildly varying amounts up to 75% of the pre-fragmentation standard deviation.

We noted that the reduction in relative standard deviation was increased for the larger *n*-grams, so that the randomising effect was most apparent for large *n*-grams, although it was not trivial for smaller *n*-grams (e.g. 5-grams still saw a reduction of RSDs to no more than 92% of the non-fragmented RSD).

Despite the "randomising" effect, features in the plaintext spectra are not obliterated totally, but are reduced. This means that the most frequently-occurring *n*-grams often remain the most frequently-occurring in proportion to other *n*-grams.

Also we noted that the two different fragmentation algorithms gave generally very similar peaks and troughs in the count, but a few troughs were deeper and some peaks higher, suggesting that the fragmentation algorithms manifest some superficially different results.

In general, while some peaks and troughs disappeared from the fragmented data, the overall general shape is roughly the same but with the detail being suppressed.

The preservation of peaks and troughs is most visible for small values of *n*. This is to be expected since the smaller n-grams are more likely to coincide with whole fragments from the original data. For example, a 5-gram spectrum will find every fragment of size 5 or above that is preserved from the original data (i.e. where fragment boundaries have not broken the contiguity of the data from the original), which will be slightly less than half of all such fragments. In contrast, a noticeable drop in the relative standard deviation occurs as n-grams become larger, sometimes by as much as 80% of the original standard deviation.

It is possible that this preservation of a file's *n*-gram profile at lower values of *n* could assist in determining the type of file, if not the actual file's contents. However, a way to alter the profile of the data would be to splice it with another type of data so that the resulting *n*-gram spectrum would be corrupted – this is called chaffing [5]. Another alternative we are currently trialling is to flip bits on selected fragments.

E. Discussion

This evaluation has concentrated on showing the technical strength of the technology, rather than showing an application of the technology in a specific Web situation. The reason for this is that the technology has a broad range of possible uses which are not limited to any one Web application but which can benefit a whole range of applications. The positioning of the current implementation at the communications layer rather than at the application level demonstrates this.

However, some real-world implementations of the technology require it to operate at the application level. Setting up fragmentation processes at lower levels secures all transmissions, whether needed or not, and requires some investment into installation, maintenance and operation of the software.

The alternative of embedding fragmentation software into mobile code such as Java applets in XML pages. This would be much easier to implement, as it could be controlled by the server, could even be enabled with little or no browser modification (depending on how well and how fast it ought to work). This option also has the benefit of being highly visible to the user[2].

In summary, this approach to securing transmissions can be seen as an alternative or an additional security measure for transactions of varying levels of secrecy.

RELATED WORK

The main aspect of this work is the use of non-symbolic fragmentation to break up the original data. This must then be used with network diversity, fragment disordering or some other means of removing the adjacency of fragments such as chaffing.

[2] Unless the user turns off the warning messages informing them, which is a good HCI decision but a terrible security decision as it never differentiates between secured and unsecured transmissions.

Network diversity, as discussed in the second section, is in common use for many purposes, including security of transmissions and will not be further discussed here.

Fragment disordering is essentially the same as transposition of fragments, making this implementation of NSF one sort of transposition cipher, another very common encryption mechanism.

However, the use of non-symbolic fragmentation brings a new aspect to these existing mechanisms, and forms the basis of the work.

Breaking data into packages of a given size is normal with digital transmissions, for example, TCP/IP data is broken into "packets". Breaking data into normal TCP/IP packets is what is done in [6], packets being transmitted as usual over a TCP/IP network, but using a form of network diversity combined with renaming of packets based on a secret-keyed algorithm for its security.

One similar cipher to NSF breaks up data into blocks which are "complete or fractional TV lines", the contents of blocks reversed and individual blocks sent out at randomly-delayed intervals. However, there appears to be no deliberate variation in block size nor any secret method governing how the block size is chosen, as is the case with NSF, with the security of the algorithm here being dependent upon the introduction of random delay [7].

A similar delay-based tactic is used in [8] where messages are broken into "elements" with an element being a section of a signal as determined by time, the elements are those parts of the message in between time interval markers. For example, a message could be broken into one-second elements. Every second element is delayed by an amount of time determined by the time of the interval (e.g. a multiple of one second) while remaining elements are transmitted with no delay. Elements are of fixed size, determined by time, and there is a disordering of elements which is governed by a shared secret. However there is no variation of element size and no suggestion that the element size is governed by a need to disguise base symbols in the underlying data.

One technology breaks up data into what might be called sub-symbolic fragments [9]. However these are not intentionally sub-symbolic, nor fragments, because the data is broken strictly into individual bits. Additionally there is a specific means for allocating bits to paths for separate transmission in a network diversity fashion, with the bits being allocated through a cycling of paths through the available options. NSF does not preclude this option although we prefer the variable path allocation as is makes illicit recovery of the message more difficult.

This latter algorithm is also likely to be relatively easy to decipher with incomplete information (and in particular with complete information should all paths be intercepted). The strict bitwise sizing of "fragments" means that it would be feasible to partially or wholly reconstruct the original signal from one or some of any intercepted transmissions, merely by using standard signal processing techniques as described above. This is one of the reasons NSF uses variable fragment sizing.

The most apparently similar work is from Adobe [10] which combines fragmentation with chaffing. Data is fragmented and each fragment put into an output unit of arbitrary size which is strictly larger than the input fragment, then filling up the rest of the output unit with junk or alternative materials. One option is even to take "chaff" from the tail end of the data to be transmitted. All output units are channelled into a single path for transmission. Essentially this is padding data at arbitrary intervals with arbitrary amounts of chaff. The fragments are not disordered at all, nor is any network diversity used, the adjacency of fragments being removed by the chaff.

Addition of chaff incurs a sizeable transmission overhead as it will be necessary to add a large enough proportion of it to disguise the true nature of the data being transmitted, especially since the true data is present in its entirety and in the correct order within the transmission. As discussed under the *n*-gram spectral analyses, it is possible to detect some characteristics of the data after fragmentation, and while chaffing may reduce this, depending on the proportion of chaff, taking materials from the tail end of the file will not.

REFERENCES

[1] All.net, 2000. *CID Security Database*, http://www.all.net/CID/Defense/Defense69.html (on path diversity) and http://www.all.net/CID/Defense/Defense68.html (on spread spectrum)

[2] Counterpane Internet Security, 2001, *Yarrow: A secure pseudorandom number generator*, http://www.counterpane.com/yarrow.html

[3] Ashman, H. and Gilbert, M., *And now for something completely different: looking ahead to new encryption and secrecy protocols*. Proceedings of Communications Design Conference, October 2001.

[4] M. Kowatsch. B.O. Eichinger and F.J. Seifert, 1985, *Message protection by spread spectrum modulation in a packet voice radio link*, Proceedings of Eurocrypt '85, pp 273-277.

[5] Rivest, R., 1998, *Chaffing and Winnowing: Confidentiality without Encryption*, theory.lcs.mit.edu/~rivest/chaffing.txt

[6] Shiroshita, T. and Yokosukashi, K., 1996, *Communication method and system with packet scrambling*, European patent 0779727A2.

[7] Bar-Zohar, M., 1986, *Video Scrambler System*, United States Patent Number 4575754.

[8] Guanella, G., 1976, *Method and device for the coded transmission of messages*, United States Patent 3970790.

[9] Kiichiro, I., 1987, *Packet transfer method*, Japanese Patent number 62195949.

[10] Amerige, S. 2000, *Secure data encoder and decoder*, International Patent number WO 00/01111.