A symmetric cipher not requiring prior sender/receiver key agreement

Les Hatton
CISM, Kingston University, London, UK

Abstract—Most modern encryption / decryption systems are asymmetric, based on prime number theory, and rely on the known intransigence of factoring numbers which are the product of two very large prime numbers. The scheme described here is very different and embodies a symmetric cipher which does not require the sender and receiver to agree on a key in advance. It relies on results from digital filter theory and asymmetric inversion properties of the discrete Wiener-Hopf equations. It is highly efficient and its sensitivity and essential non-uniqueness may make it computationally infeasible to crack.

0.0.0.1 : This paper demonstrates the theory, a practical implementation, timing results and discusses possible ways in which it might be attacked.

1 Overview

Cipher systems are conventionally split into two categories.

- Asymmetric systems. In an asymmetric system, the encryption key and the decryption keys are separate. Public key cryptography exploits the current intransigence of factoring the product of two large prime numbers in a computational reasonable time. The product is published openly but the factors are kept private and using an algorithm discovered independently by Ellis, Cocks and Williamson at GCHQ in the UK in the early 1970s and by Rivest, Shamir and Adleman in 1977, [7], either of the private keys can be used to decrypt a message encrypted with the public product key. Other more modern methods include those based on elliptic curves discovered independently by Koblitz, [4] and Miller, [5].
- Symmetric systems. In contrast, symmetric systems are faster and use the same key for encryption and decryption but hitherto have required the sender and receiver to agree in advance in some way on the key to be used. In a hybrid system, the key is first exchanged using asymmetric methods.

1.0.0.2 : The current system is very different. It uses a symmetric key but this can be exchanged between sender and receiver without first agreeing on it. The method uses a Diffie-Hellman-Merkle protocol and relies on linear convolution, floating point arithmetic and asymmetric inversion properties of the discrete Wiener-Hopf equations, [2]. Conventional public key systems can in principle be broken by sufficient computational power. For example, for prime number based systems, a very large number which is the product of two large primes can eventually be factorised into those two primes and no other. It is believed that the current scheme cannot be broken in the same way as any solution is demonstrably *non-unique* without exact knowledge of the message and the filter used to encrypt it. The method can either be used in a hybrid manner as a conventional symmetric algorithm within the public-key system or on its own.

L.Hatton@kingston.ac.uk, lesh@oakcomp.co.uk

2 The theory

Consider two arrays of numbers $f = \{f_0, f_1, ..., f_{L_f-1}\}$, where L_f is the number of elements in f, and $m = \{m_0, m_1, ..., m_{L_m-1}\}$, where L_m is the number of elements in m. f will be referred to as the filter or key, and m as the message. m will typically have been encoded from its textual form using some standard scheme such as ASCII. Then the discrete convolution of f and m is defined by

$$e_j = \sum_{k=0}^{L_f-1} f_k m_{j-k} \tag{1}$$

where $e = \{e_0, e_1, ..., e_j, ..., e_{L_f+L_m-1}\}$ is the output.

2.0.0.3 : This is usually written as e = f * m, where * is the discrete convolution operator. First of all, it is relatively simple to prove that the * operator is commutative, [3], so that f * m = m * f. This is an essential property as will be seen.

2.0.0.4 : Suppose now there exists an inverse filter g such that f * g = i, the identity filter. The identity filter i = {0, 0,..., 1, 0, 0, ... 0}. This contains $L_f + L_g - 1$ samples altogether, where L_g is the number of samples in g, with the single unit value preceded by $L_l \geqslant 0$ zeroes. L_l is known as the *lag* or *delay* and is related to the phase spectrum of the filter f if it is treated as a time series. This is a most important factor in the performance of the inverse filter g as will be discussed shortly.

2.0.0.5 : Applying g to both sides as follows gives

$$g * e = g * f * m = i * m = m \tag{2}$$

In other words, it would be possible to recover m simply by applying this inverse filter. This problem was first solved in the least squares sense by the great American mathematician Norbert Wiener, [8] as follows:-

2.0.0.6 : Form the sum of squares of the difference between the actual output of a digital filtering operation using the filter g on the filter f and the desired output d as

$$S = \sum_{k=0}^{L_g+L_f-1} \{d_k - \sum_{t=0}^{L_g} g_t f_{k-t}\}^2 \tag{3}$$

where $d = \{d_0, d_1, ..., d_{L_g+L_f-1}\}$ is the desired output. If the actual output d' = g * f exactly produces d, then the sum of squared errors S, is simply zero.

2.0.0.7 : To find the best-fitting filter in the least-squares sense, take the partial derivatives with respect to each filter element g_t so that

$$\frac{\partial S}{\partial g_t} = 0, \forall t \tag{4}$$

With some small manipulation, this leads directly to the discrete Wiener-Hopf equations

$$\sum_{t=0}^{L_g} g_t \sum_{k=0}^{L_g+L_f-1} f_{k-t} f_{k-j} = \sum_{k=0}^{L_g+L_f-1} d_k f_{k-j} \tag{5}$$

for j = 0, 1, .. , $L_g - 1$. The second sum on the left hand side is the autocorrelation of the input filter whilst the right hand side is the cross-correlation of the input filter with the desired output, d.

2.0.0.8 : Exploiting the Töplitz form of the auto-correlation matrix (each diagonal has only one unique element), equation (5) can be solved efficiently for g using an algorithm due to Norman Levinson quoted in the Appendix of [8] in O(N^2) instead of O(N^3). The relevance of this can be seen by choosing the desired output to be the identity filter described above. In this case, i then acts as an identity operator and g acts as an inverse filter to f, such that $f * g \simeq i$. The quality of this approximation is crucial to the application described here.

2.1 Accuracy of the Wiener-Hopf inverse

It can be seen from above that the performance of an inverse filter is directly related to the size of S in equation (3). S is always ≥ 0 and the filter does a perfect job when S = 0. Using this, the filter performance of Wiener inverse filters is then defined as:-

$$P = 1 - \frac{\sum_{k=0}^{max(L_f,L_g)+L_m-1} (d'_k - d_k)^2}{\sum_{k=0}^{max(L_f,L_g)+L_m-1} d_k^2} \tag{6}$$

implying $P \in [0, 1]$ with P = 1 corresponding to perfect filter performance and P = 0 corresponding to worst case performance when a filter simply outputs zero at every point.

2.1.0.9 : The actual value of P achievable in practice turns out to depend on a number of factors including the phase spectrum of this filter, its amplitude spectrum and its length in elements, [8]. The amplitude and phase spectra referred to here are simply those obtained when the filter is considered as a time series and is then Fourier transformed, [6]. In this context, a minimum phase filter is the one which has the smallest phase spectrum for a given amplitude spectrum.

2.1.0.10 : Of particular note is the fact that there are classes of filters which can never be perfectly inverted, [8]. This is a non-trivial observation but some insights into it can be gained as follows.

2.1.0.11 : First of all and perhaps most obviously, the degree to which a given filter can be inverted depends on its amplitude spectrum. To see this, it should first be noted that discrete convolution is exactly equivalent to multiplication of the Fourier transforms of each of the filter and its inverse, so that f * g corresponds to the product FG where F and G are the Fourier Transforms of the filters f and g respectively. To invert a filter, FG = I where I is the inverse of the identity filter i, it should first be observed that the amplitude spectrum of I is 1 for all frequencies. In other words to invert the amplitude spectrum of a filter f, it suffices to multiply it by the amplitude spectrum of $\frac{I}{F}$. This is trivial provided that F $\neq$ 0 at all frequencies or there would be a division by zero. For filters for which F = 0 at some frequencies however, only an *approximate* inversion can be done, (in practice by whitening the spectrum artificially by constructing an inverse filter $\frac{I}{F+\delta}$, where δ is a small non-zero constant, removing any zeroes, but also slightly disturbing the inversion at all frequencies, [3]).

2.1.0.12 : It should also be noted that for a given length of filter and amplitude spectrum, filter inversion performance also varies dramatically depending on the phase spectrum of the filter. A minimum phase filter is best inverted with a desired output of zero delay, (that is a unit value at the start, followed by zeroes). A mixed phase filter is best inverted with some non-zero delay in the desired output. Minimum phase filters are one-sided, i.e. their non-zero values occur entirely after time zero whereas mixed phase filters also have anticipation components (some non-zero values before time zero).

2.1.0.13 : Last but not least, filter performance also depends on how quickly the amplitude spectrum is varying and the length of the filter used. However, in summary, as [8] shows, for a class of filters, there is an asymptotic limit such that $P < 1$ even as filter length $\rightarrow \infty$.

2.1.0.14 : *In the context of the present application, the central idea is to use a class of filters for encryption and decryption which can be successfully inverted, whilst denying a potential eavesdropper access to enough information to allow them to extract the filters with sufficient accuracy to permit eavesdropping.*

2.1.0.15 : The encryption operation simply involves choosing some suitable invertible filter and con-

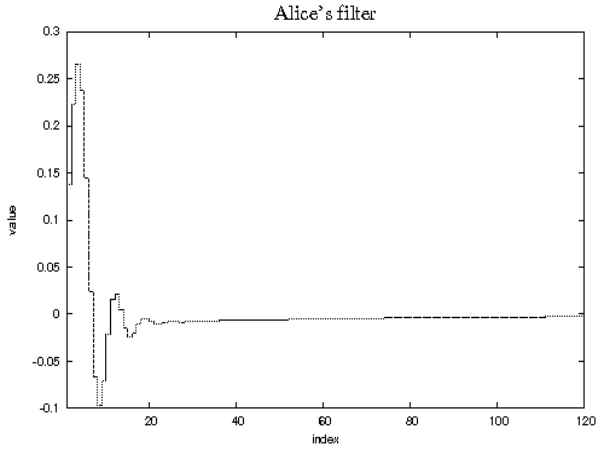

Fig. 1. Alice's minimum phase filter of 118 points

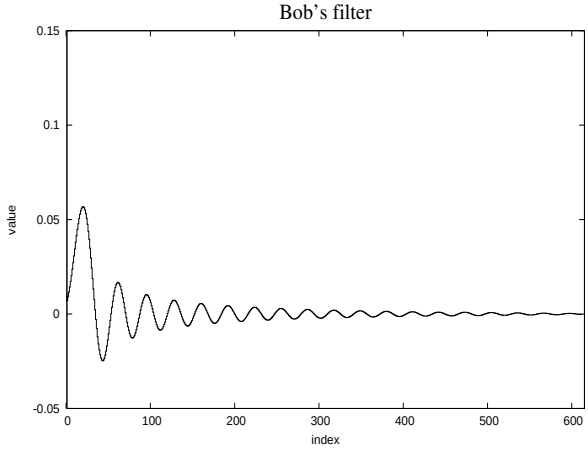

Fig. 2. Bob's minimum phase filter of 615 points

volving this with the message m as in equation (1) to produce an encrypted output e = f * m. In practice, this is done in the frequency domain which has considerable value for long messages as the FFT is particularly efficient requiring only nlogn operations for n numbers.

2.2 Practical construction of invertible filters

Given the comments above, attention is confined to filters which have no zeroes in the amplitude spectral response. The phase of the filter is not important but it is far more efficient to use minimum phase filters. Many classes of filters are suitable but for reasons which will become obvious shortly, those filters which can be constructed from a relatively small number of parameters in a standard way are particularly suitable, provided there are sufficient parameters to render guessing infeasible. For minimum-phase band-pass filters, which are used here to illustrated the method, four 64-bit floating point parameters can be used to specify them, (low-cut frequency l_c, high-cut frequency h_c, low-cut slope l_s, and high-cut slope h_s). Such a filter can then be described functionally by

$$f = f(l_c, l_s, h_c, h_s) \tag{7}$$

and uniquely generated from just these numbers.

2.2.0.16 : The minimum phase filter for any given amplitude spectrum can be constructed in a number of different ways but here the Hilbert Transformation [3], was used. Examples of filters generated in this way are shown in Figures 1 and 2. Filter values are specified to 6 significant figures and double precision arithmetic is used throughout. These filters will be used as keys in an Alice-Bob transaction later. For these filters, an inverse filter of 9 times the original filter length was constructed in each case. This leads to filter performances $P \geq 0.999999$ corresponding to individual letter errors of $< 0.1\%$, c.f. equation (6). This is far more than sufficient to deconvolve its filter returning the original encoded message verbatim. This invertibility is independent of any message combined with it. It depends only on the filter.

2.3 Encryption and Decryption

The above considerations lead to the following encryption / decryption scheme.

1) Encode the text by changing the text into some suitable numeric form, for example an ASCII representation. In this representation, the space character takes the decimal value 32 for example. Of course a more complex encoding employing some further form of encryption could also be used but this simple step will suffice here.
2) Choose an invertible filter f (the detailed mechanics of this will be discussed in more detail shortly) and convolve it with the encoded message m to give m * f, usually by frequency domain multiplication.
3) Design the inverse filter g such that $f * g \simeq i$. Then:-
$$(m * f) * g = m * (f * g) \simeq m * i = m \tag{8}$$
thus recovering the original encoded message. This will be discussed in more detail in the section on practical considerations but in essence for an encoding step such as ASCII which produces integers in the range between about 30 and 100, the accuracy of the reconstruction must be such that if n_e is the encrypted number and n_d is the decrypted number, then
$$|n_e - n_d| < 0.5 \tag{9}$$
In practice as will be seen, it is simple to ensure that this is the case for all messages.
4) Decode the message back into text by reversing the first step. In this case, this is achieved trivially by rounding each element to the nearest integer and looking up the values in the ASCII table.

2.4 Non-uniqueness

A novel property of this method is that it is essentially non-unique. This derives from the fact that computation

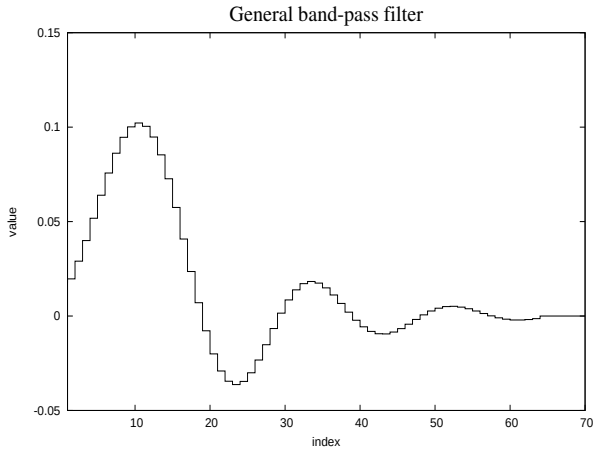

Fig. 3. An example of typical encrypting band-pass filter.

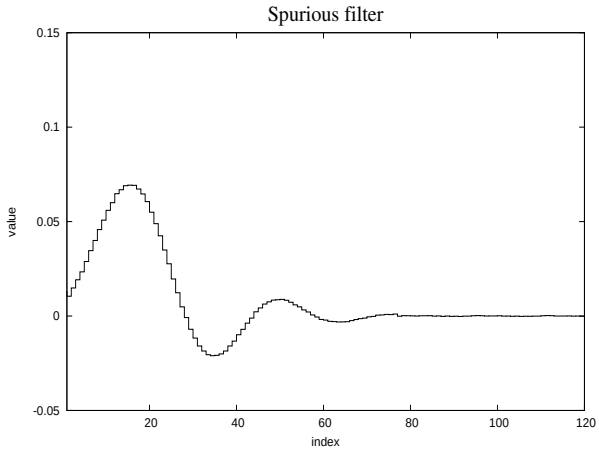

Fig. 4. An example of a spurious filter f_E which when used in a brute force attack, leads to an entirely spurious message resulting as described in the text.

is being done in floating point rather than in integer format as will be seen.

2.4.0.17 : Consider the following argument. The original transmission is $m * f$. To extract the message m, the filter f would have to be known. In conventional cryptographic systems, there is a unique answer which can be identified as such enabling any search to stop and the message revealed. Here however, there is not. To see this, consider an alternative but entirely spurious message m_f such that:-

$$m * f = m_f * f_E \tag{10}$$

where f_E is some other filter. Now, if an eavesdropper came across the filter f_E in a brute force attack and applied its inverse to $m * f$, they would get $m * f * f_E^{-1} = m_f * f_E * f_E^{-1} = m_f$, an entirely spurious but possible result. Such filters exist as a direct result of the redundancy created by expanding each letter of the original message in some numeric code into a floating point number. At 64 bit double precision, this would correspond to an increase by a factor of 8 in message size.

2.4.0.18 : To see this in practice, an example of such a spurious filter f_E is shown in Figure 4. In this case, for a true message m = "9999888877776666" and a filter f as shown in Figure 3, applying the filter f_E to a message m_f = "1234567856781234" will result in exactly the same output, (i.e. $m * f = m_f * f_E$). In other words, when presented with $m * f$ where m = "9999888877776666", if the filter f_E is guessed and its inverse applied, the entirely reasonable but nevertheless completely incorrect message m_f = "1234567856781234" results.

2.5 Application using Diffie-Hellman-Merkle (DHM) key exchange

To use such filters as a cipher system, a form of the DHM key exchange process [1] is used. In its most basic form, this proceeds as follows:-

1) Alice chooses a digital filter f_A, encrypts her encoded message m to get $m * f_A$ and sends it to Bob.
2) Bob chooses a digital filter f_B, and further encrypts the encrypted message m to get $m * f_A * f_B$ and sends it back to Alice.
3) Alice calculates the inverse filter for f_A, f_A^{-1}, such that $f_A * f_A^{-1} \simeq i$ and applies it to the received double encrypted message from Bob and sends it back to him as $m * f_A * f_B * f_A^{-1}$ which since * is commutative can be written as $m * f_A * f_A^{-1} * f_B \simeq m * i * f_B \simeq m * f_B$.
4) Bob now applies the inverse of his own filter f_B^{-1} say, such that $f_B * f_B^{-1} \simeq i$ to the message $m * f_B$ to get $m * f_B * f_B^{-1} \simeq m * i \simeq m$ revealing the original encoded message m which can be decoded back into text trivially as described above.

Unfortunately, this process would not work in this form for the convolutional filters described here. These filters of course commute as is required by this method, but an eavesdropper Eve listening on all three exchanges has two opportunities to eavesdrop. Exchange 1 and exchange 2 would in principle allow Eve to recover f_B by applying the discrete Wiener-Hopf method of equation (5) to design an inverse filter $(m * f_A)^{-1}$ and applying it to $m * f_A * f_B$, yielding filter f_B. Applying an inverse to this for exchange 3 then gives the message m without Alice or Bob realising.

2.5.0.19 : Alternatively, Eve could use exchanges 2 and 3 to extract $(f_A)^{-1}$ and then exchange 1 to access the message m. (In short, there are three eavesdropped parts and three unknowns). Such inversions are not always possible and depend on the theoretical invertibility of filters like $m * f_A$ as discussed earlier.

2.5.0.20 : It is in fact possible to condition m so that $m * f_A$ has zeroes in its spectrum. This does not affect Bob but it would affect Eve, making it difficult if not impossible to eavesdrop. However, another far more

effective step can be taken as will now be seen.

2.5.0.21 : The following modified scheme depends crucially on this constraint

$$L(m * f_A) < L(f_B) \tag{11}$$

, where L() indicates *length of*. This turns out to be very easy to achieve but to make the scheme exactly the same for all messages, the initial exchange uses a short message p, *which contains only the parameters necessary to construct a filter f_p which will be used for the eventual transaction.* This is not strictly necessary but otherwise long messages would require exceptionally long filters to satisfy constraint equation (11).

1) Alice chooses a *minimum-phase* digital filter f_A, encrypts her encoded parameters p to get $p * f_A$ and sends it to Bob.
   2.5.0.22 : *Meanwhile, Eve records $p * f_A$.*
2) Bob adds some random text r to the *end* of the received encrypted message $p*f_A$ and then chooses a minimum-phase digital filter f_B *which is significantly longer than the length of $p * f_A$*, and further encrypts the encrypted parameters message p and its augmentation to get $(p * f_A + r) * f_B = p * f_A * f_B + r * f_B$ and sends it back to Alice.
   2.5.0.23 : *Eve records this too. With knowledge of $p * f_A$ from the first transaction, Eve constructs $(p*f_A)^{-1}$ and applies it to the current transaction yielding $f_B + r * f_B * (p*f_A)^{-1}$. It looks like Eve has been able to extract Bob's filter. Unfortunately, she is in trouble because the corrupting appended text $r * f_B * (p*f_A)^{-1}$ still starts after $L(p*f_A)$ samples as the filters are minimum phase but by construction $L(f_B)$ is significantly longer than that. This means that crucially, Eve only has the first $L(p*f_A)$ samples of $f_B = f'_B$, say, as the remainder are corrupted in an unknown manner, (she does not know r). As will be seen, f'_B is insufficiently accurate to be useful.*
3) Alice calculates the inverse filter for f_A, f_A^{-1}, such that $f_A * f_A^{-1} \simeq i$, the identity filter, and applies it to the received double encrypted message from Bob, adds further random text s to this, where s starts at the same place as r[1], and sends it back to him as $(((p * f_A + r) * f_B) + s) * f_A^{-1}$ which since * is commutative can be written as $p * f_A * f_A^{-1} * f_B + r * f_A^{-1} * f_B + s * f_A^{-1} \simeq p * f_B + r * f_A^{-1} * f_B + s * f_A^{-1}$.
   2.5.0.24 : Eve records this but applying $(f'_B)^{-1}$, the inverse of her corrupted estimate of Bob's f_B filter to the whole thing, just leaves an unintelligible mess.
   2.5.0.25 : Eve has another opportunity to eavesdrop however. If she uses the second transaction between Bob and Alice, $p * f_A * f_B + r * f_B$, and designs an inverse filter for the whole thing, yielding $(p * f_A * f_B + r * f_B)^{-1}$ and applies that to this third transaction between Alice and Bob, this results in $(f_A)^{-1} + s * (p * f_A * f_B + r * f_B)^{-1} * f_A^{-1}$. In theory, this allows Eve to access $(f_A)^{-1}$ and therefore f_A. However, as in the earlier attempt to eavesdrop, Eve only has access to the first $L(p * f_A)$ samples of $(f_A)^{-1}$ because s starts at the end of the original transaction $p * f_A$. So provided

$$L((f_A)^{-1}) > L(p * f_A) \tag{12}$$

   , which can easily be enforced as the inverse filters are typically several times longer than $L(p * f_A)$, Eve has the same problem as before. She can only estimate the filter f_A at a level insufficient for it to be usable.
4) Bob now applies the inverse of his own filter f_B^{-1} say, such that $f_B * f_B^{-1} \simeq i$ to the message $p * f_B + r * f_A^{-1} * f_B + s * f_A^{-1}$ to get $p * f_B * f_B^{-1} + r * f_A^{-1} * f_B * f_B^{-1} + s * f_A^{-1} * f_B^{-1} \simeq p + r * f_A^{-1} + s * f_A^{-1} * f_B^{-1}$. This will reveal the original encoded parameters p because both r and s were chosen to start at the end of the initial transaction $p * f_A$. However, since the filters f_A and f_B are minimum phase, so are their inverses, so that there is no forward leakage into the message p from either augmented text, r or s, (i.e. because the inverse filters are also one-sided). In other words, $r * f_A^{-1}$ and $s * f_A^{-1} * f_B^{-1}$ still start at the end of $p * f_A$. After this deconvolution, message p appears at the beginning and can be trivially extracted since $L(p) < L(p * f_A)$ and only the first $L(p)$ samples are significantly different from zero, (the next $L(p * f_A) - L(p)$ samples are all effectively zero before the influence of the corrupting texts r and s is felt). After extraction, these first $L(p)$ samples can then be decoded back into text as described above. Bob now has the filter parameters p for the final transaction.
5) Alice now knows that Bob has the filter parameters p she is going to use so designs a filter f_p from them, convolves it with her real message m and sends it to Bob as $m * f_p$. Bob designs the same filter knowing the filter parameters from the above exchange, designs the inverse filter f_p^{-1} and convolves it with $m * f_p$ to get $m * f_p * f_p^{-1} = m$. The real message m can be of any length. Eve does not know p because her attempts at eavesdropping are insufficiently accurate, and therefore cannot construct f_p and is left only with conventional brute force attacks on this which would fail because of the non-uniqueness described earlier.

2.5.0.26 : In short, from Eve's point of view, the three potentially eavesdropped transactions, $p * f_A$, $(p * f_A + r) * f_B$ and $((p * f_A + r) * f_B + s) * f_A^{-1}$ are insufficient to reconstruct either f_A or f_B and therefore the message p. (This time, there are three eavesdropped parts and five unknowns).

3 Some examples, timing and practical considerations

This section shows an implementation of the method described in the previous section with the full transaction

1. s needn't be text. Any floating point numbers could be used.

between Alice and Bob shown again with Eve's efforts to eavesdrop. The software necessary to encode / encrypt / decode and decrypt including all the signal processing software to construct and invert filters optimally totals around 3000 source lines of C.

3.0.0.27 : To demonstrate the method, minimum phase band-pass filters were used, although there is no practical limitation on which types could be used provided they are invertible in the sense described earlier. The only requirement are that the constraints expressed by equations (11) and (12) are satisfied. In this case, the parameter message file p is just a header with the filter type and four floating point numbers are passed representing the band-pass filter low-cut, low-slope, high-cut and high-pass values as follows:-

```
BANDPASS
0.0000000
18.000000
128.00000
72.000000
```

In the terminology of equation (7), $f_p = f_p(0.0, 18.0, 128.0, 72.0)$.

3.0.0.28 : Alice's filter $f_A = f_A(0.0, 18.0, 64.0, 72.0)$ is shown in Figure 1 and Bob's longer filter $f_B = f_B(0.0, 18.0, 64.0, 350.0)$ is shown in Figure 2. These filters were constructed in the frequency domain and are relatively slowly varying with no zeroes in their amplitude spectra and are for illustration only. In practice, less obvious (and more) parameters would be used.

3.0.0.29 : The minimum phase filter for any given amplitude spectrum can be constructed in a number of different ways but here the Hilbert Transformation [3] was used. Filter values are specified to 6 significant figures and double precision arithmetic is used throughout. For this example, an inverse filter of 9 times the original filter length was constructed in each case as described earlier. In practice, longer or shorter filters may be necessary but this can be computed automatically from the filter performance, equation (6). f_B and f_A^{-1} obey the constraint equations (11) and (12) respectively.

3.0.0.30 : Filter computation is very rapid and takes only a few 10s of msec on a typical desktop and it should also be noted that the errors are related only to the filter not the message being transmitted. *In other words, if the inversion works for one message, it works for all.*

3.1 Without randomising text

Initially the transaction was run without adding randomising texts r and s, to see how accurately Eve can eavesdrop the message p. The complete transaction is shown as Figure 5. The parts of the figure are:-

a) The parameters comprising the message file p, encoded as ASCII are shown as part a). The newline character (encoded ASCII decimal value 10) can be seen at element number 9 (end of BANDPASS), element number 19 (end of 0.0000000), element number 29 (end of 18.000000), element number 39 (end of 128.00000) and element number 49 (end of 72.000000).

b) The initial transaction to Bob, $p * f_A$, showing the blurring, lengthening effect of convolution.

c) Bob's returned transaction to Alice $p * f_A * f_B$

d) Alice's deconvolved message returned to Bob is $p * f_A * f_B * (f_A)^{-1} = p * f_B$.

e) Eve's attempts to extract p using the eavesdropped $p * f_A$ and $p * f_A * f_B$ to construct f_B and from this $(f_B)^{-1}$ to apply to $p * f_B$.

f) Bob's reconstruction using his own directly built $(f_B)^{-1}$. As can be seen by comparison with a), this is a perfect reconstruction. Bob's reconstruction takes just a few milliseconds.

Eve's reconstruction in e) necessitated inverting a mixed-phase filter optimally and took some 5000 times longer and as can be seen from the first 49 samples, although not accurate enough, is sufficiently close perhaps to worry Alice and Bob. If enough time is spent, this can be improved further although it may not be possible to get the correct result as filter performance for mixed-phase filters may never get close enough to 1 as discussed above. It is however not reassuring enough for cryptographic purposes.

3.2 With randomising text

In contrast, Figure 6 repeats each step but this time with the confounding randomised text appended after the initial $p * f_A$ transaction and also in the final transaction. The parts of the figure are:-

a) Same as Figure 5, part a).

b) Same as Figure 5, part b).

c) Bob's returned transaction to Alice $(p * f_A + r) * f_B$. The influence of r can clearly be seen from around sample 170.

d) Alice's deconvolved message returned to Bob is $((p * f_A + r) + s) * f_B * (f_A)^{-1} = p * f_B + r * f_B * (f_A)^{-1} + s * (f_A)^{-1}$.

e) Eve's attempts to extract p using the eavesdropped $p * f_A$ and $(p * f_A + r) * f_B$ to construct f'_B and from this $(f'_B)^{-1}$ to apply to $p * f_B$. By restricting Eve to the first $L(p * f_A)$ samples of f_B, the attempt to eavesdrop is foiled completely. The inversion attempt shown here can be compared with the same thing without randomising text as shown in Figure 5, part e).

f) Bob's reconstruction using his own directly built $(f_B)^{-1}$. As can be seen by comparison with a), this is again a perfect reconstruction.

The addition of the confounding random text for transactions 2 and 3 between Alice and Bob stops Eve extracting the parameters and so she is unable to eavesdrop on the final transaction $m * f_p$. Bob however, with exact knowledge of f_B has no problem extracting an accurate

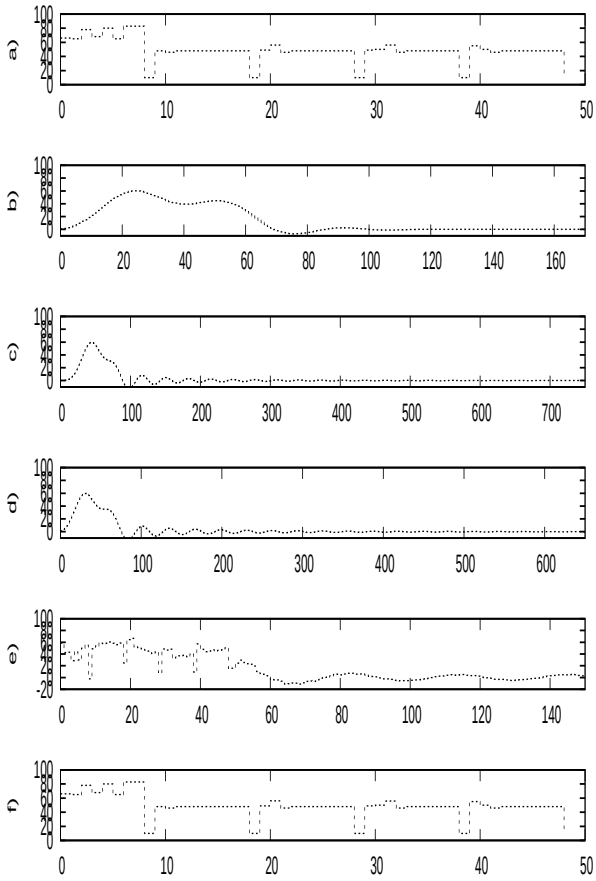

Fig. 5. The complete transaction between Alice and Bob without using randomly added text. a) is the initial parameter message p, b) is $p * f_A$, c) is $p * f_A * f_B$, d) is $p * f_A * f_B * (f_A)^{-1}$, e) is Eve's attempt to reconstruct the message in a) and f) is Bob's reconstruction of a) which is an exact match.

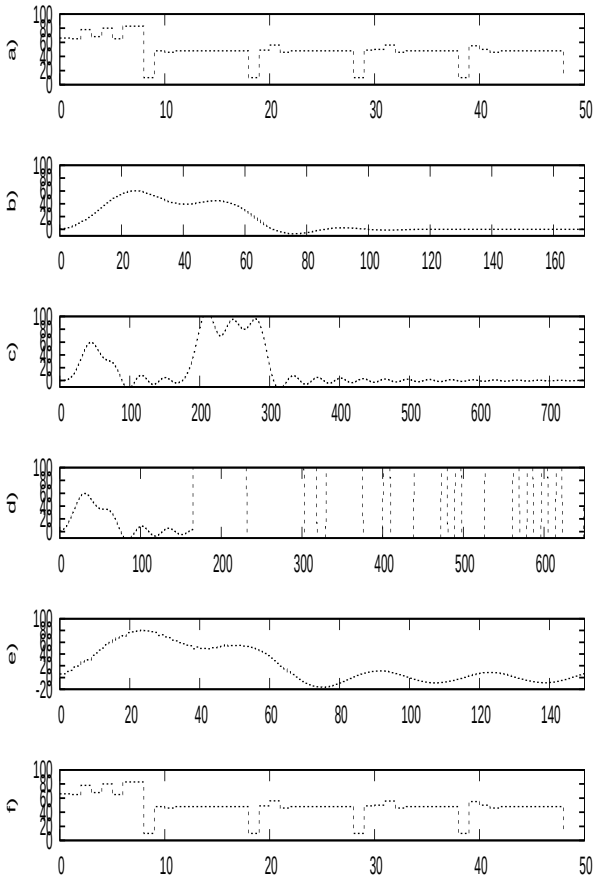

Fig. 6. The complete transaction between Alice and Bob using randomly added text. a) is the initial parameter message p, b) is $p * f_A$, c) is $(p * f_A + r) * f_B$, d) is $(((p * f_A + r) * f_B) + s) * (f_A)^{-1}$, e) is Eve's attempt to reconstruct the message in a) and f) is Bob's reconstruction of a) which is again exact. The randomising texts can be see clearly in c) from sample 170 onwards.

version of p, constructing f_p and thus f_p^{-1}, and retrieving the final message m.

3.3 Timing on different message lengths

To test the algorithm timing, three standard texts were used as messages. For each of these the time to encode, encrypt, decrypt, decode and verify was measured on a single processor AMD XP2800+ machine, (Linux Bogomips rating 4170.08). The texts were treated in one block although it would be perfectly feasible to split a very large text into pieces and splice the pieces together allowing for the filter or inverse filter length appropriately, however the n log n nature of the FFT timing means that this is hardly worth it. The results follow:-

Text	Length in Kb.	User timing (s.)
Licence agreement	1.4	0.300
Hamlet	182.6	4.104
King James Bible	4445.3	130.256

3.3.0.31 Table 1: Timings to encode / encrypt and decrypt /decode various texts with the filter of Figure 1.:

3.3.0.32 : The frequency domain implementation means that the cost of the operation is effectively independent of the length of the filter provided the filter is shorter than the message.

3.4 Sensitivity of the inversion

Although brute force attacks are ruled out because of the non-uniqueness described above, it is worthwhile performing a sensitivity analysis to see at what level of imprecision in one of the significant figures in one element of the encrypting filter, does the decryption fail ? Convolutional algorithms such as this have the property of spreading the error over many adjacent characters and the algorithm is very sensitive as a result. As an example, consider a message comprising the opening dialog of Shakespeare's Midsummer Night's Dream.

```
THESEUS   Now, fair Hippolyta, our nuptial
hour draws on apace; four happy days
bring in another moon: but, O, methinks,
how slow this old moon wanes! she
lingers my desires, like to a step-dame
or a dowager long withering out a young
man revenue.
```

The decrypted text with just one unit change in the second significant place of one of the elements of a 118 point filter looks like:-

```
THESEUS   Now, fair Hippolyta, our nuptial
 hour draws on apace; four happy days
 bring in another moon: buy+^^M)
mbukosll(#how^Zuprx^_ ^_^^^\Sols^Yrmc^_ln
pj&{cn]s""tig"igk`iup&rzcaomteq%^M""!!^[P
kne^Yvq^_`^^mxhk/gdof"qp^\\^]dj}ejgo^H%#
^Y^\Onnm!wgrgdsfpo#pmw _^^poxrk"rck^
```

As can be seen, after the first few characters when the incorrect filter element begins to be used in the convolution, the convolutional error spreads rapidly and inversion accuracy rapidly degrades. Note that this means that any errors in transmission will quickly render the message impossible to decrypt even if the correct key is known.

3.5 Constructing filters

3.5.1 Efficiency

Unlike public keys, the private filters used in the method described here can be calculated extremely efficiently allowing the possibility of choosing a new filter for say every IP address contacted, or even every communication if desired to defeat plain-text attacks as discussed shortly.

3.5.2 Stability

The principle point here is that the amplitude spectrum of a filter must be non-zero at each frequency to allow correct inversion. It is also a property of the Fourier domain that steepness in the amplitude spectrum corresponds to duration in the time domain, [6]. In other words if the amplitude spectrum is very spiky, the corresponding filters are long. This is no drawback of course as the filters are applied by frequency domain multiplication.

3.5.3 Optimum delay

For the DHM exchange described above, minimum phase is selected as it requires no search for the optimum delay or lag L_l and there is no chance the augmentation text can bleed into the original message. For minimum phase the optimum delay is simply zero and this allows the filters to be chosen more efficiently. Note that the amplitude spectrum can be selected in many ways. It could be generated randomly or it could be parameterised as some kind of smooth polynomial as was done here.

3.6 A practical example with credit cards

Here 1000 credit card identification records, (16 digit credit card number, 3 digit security code, name, address and the expiry date in DD-MMM-YYYY format) were constructed and sent cryptographically using the DHM exchange method described above. To demonstrate the efficiency, different pairs of cooperating filters (Alice's and Bob's) and different augmentation texts were constructed for every credit card record. On the same machine as used for the timing tests reported above, filter generation takes around 0.1 second for filters around 200 points on average and the DHM transaction takes about the same. No effort has been made to optimise this and the DHM exchange is done in five separate stages. These times also included a file comparison to check for transmission corruption. It is expected that very significant speed-ups could be obtained.

4 THREATS

4.1 Attack by brute force and non-uniqueness

As was described earlier, it is a property of many cryptosystems that they are susceptible to brute force attacks. This is essentially because there is a unique answer. In the case of public key cryptosystems, the public key is openly circulated and there is only one factorisation possible in the case of prime number systems. Eventually, given enough computer power, the factorisation can be identified independently of any message and then any messages encrypted with that public key can be successfully decrypted.

4.1.0.1 : The current method is completely different. There is *no* unique solution which can be identified.

4.1.0.2 : This means that brute force attacks can never work because there would be no means of determining which of a potentially very large number of possible messages was the correct one.

4.1.1 Attack by letter frequency analysis

Since the filter is unknown and convolutional filters smear data across perhaps many adjacent encoded letters, there is no simple relationship between these and their natural frequency. Such attacks would be irrelevant.

4.1.2 Plaintext attack

If Eve got hold of a message m, and the initial transaction of Alice and Bob, $m * f_A$, she could trivially extract f_A. However, since filters can be computed so efficiently, it is simple to require a different filter to be generated for each transaction rendering any knowledge of a particular filter used previously irrelevant, just like a one-time pad. This makes the DHM exchange described above whereby details of the filter are unnecessary for the transaction, of crucial importance.

4.2 Disadvantages

4.2.1 Message size

It should be noted that encrypted messages are about 8 times longer than the original message because each character, normally encoded as a number no greater than 127 in ASCII is converted into a double-precision (64 bit here) number. However this encryption allows the extraordinarily efficient floating point capabilities of modern chipsets to be used effectively. Although the resulting redundancy neutralises any possibility of a brute-force attack as described above, it would be a disadvantage in communications with limited bandwidth.

4.2.2 Person in the middle attacks

Whilst it seems there is little possibility of attacking by normal eavesdropping, there remains the enduring problem of person in the middle attacks which would have to be resolved by other means. In this case an eavesdropper, Eve, could take over the exchange with Alice using her own private key and augmentation text r and after receiving the message using the DHM protocol described earlier, initiate a second exchange with Bob using another private key and augmentation text so that Bob receives the genuine message. As usual with such attacks, without any trusted means of identifying each other, then Eve will be able to read the original message and neither Alice nor Bob will be any the wiser.

5 Conclusions

A new cryptographic scheme based on floating point convolution and the discrete Wiener-Hopf equations has been presented along with examples of timing and its feasibility demonstrated by successful Diffie-Hellman-Merkle key exchange. The scheme is believed to be unique in that it is based on a symmetric key but without the need to communicate a private key. Messages will typically be longer by a factor of 8 but the implied redundancy means that this scheme appears to be impossible to break by any brute force attempts because of non-uniqueness, although that will no doubt be addressed by the cryptographic community.

5.0.2.1 : The new scheme is highly efficient and does not need any public key to function although it can work within the public key system. Since the most time-consuming part of the algorithm is usually Fourier Transformation, its timing will normally depend on nlogn where n is the length of the message. Finally, its sensitivity means that it will not tolerate errors in transmission.

6 Acknowledgements

I would like to thank Keith Gibson for a series of penetratingly useful remarks about earlier drafts of this paper.

References

[1] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22, 1976.

[2] L. Hatton. Wiener-hopf cryptography, 2008. UK patent application No. 0822870.2, 16 Dec 2008.

[3] L. Hatton, M.H. Worthington, and J. Makin. *Seismic Data Processing, Theory and Practice*. Blackwell Scientific, 1st edition, 1986.

[4] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[5] V.S. Miller. *Use of elliptic curves in cryptography*, volume 218 on Advances in cryptology—CRYPTO 85 of *Lecture notes in computer sciences*. Springer-Verlag New York, Inc, 1986. ISBN 0-387-16463-4.

[6] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc, New Jersey, 1975. ISBN 0-13-214635-5.

[7] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, February 1978.

[8] N. Wiener. *The Extrapolation, Interpolation and Smoothing of Stationary Time Series*. MIT Press, 1942.