Gray–Markel structures and approximately the same as the parallel structure.

As a measure of roundoff noise in digital filters $G = \sum_i^n k_{ii} w_{ii}$ is used which gives the noise gain of the scaled structure in terms of the diagonal elements of the matrices $\boldsymbol{K}$ and $\boldsymbol{W}$ of the unscaled one.

Mullis and Roberts [3] have proved that the realisation of narrowband LP transfer functions by DTDF leads to structures with very high roundoff noise. They have shown that under the LP–LP transformation $z \Rightarrow (z - \alpha)/(1 - \alpha z)$ the output noise gain is directly proportional to $1/(1 - \alpha^2)^{2n-2}$. Because for narrowband filters $\alpha \rightarrow 1$, the noise gain takes very large values. It will be shown in the following that for the TCTDF the output noise gain is invariant under the LP–LP transformation, i.e. if

$$\hat{H}\left(\frac{z - \alpha}{1 - \alpha z}\right)$$

is realised with TCTDF realisations for different values of α, the noise gain for any realisation will be independent of α. Under the bilinear transformation and the LP–LP transformation given above, s is transformed as $s \Rightarrow s/\omega_0$ where $\omega_0 = (1 - \alpha)/(1 + \alpha)$. Under this LP–LP transformation we obtain $H_1(s) = H(s/\omega_0)$ which assumes a state realisation R_1 obtained from the CTDF realisation R_0 through the transformation

$$\boldsymbol{A}_1 = \omega_0 \boldsymbol{A}_0 \quad \boldsymbol{b}_1 = \omega_0^{1/2} \boldsymbol{b}_0 \quad \boldsymbol{c}_1 = \omega_0^{1/2} \boldsymbol{c}_0 \quad \boldsymbol{d}_1 = \boldsymbol{d}_0$$

Using these in eqn. 5 yields $\boldsymbol{K}_1 = \boldsymbol{K}_0$, $\boldsymbol{W}_1 = \boldsymbol{W}_0$. Any other realisation of $H_1(s)$ can be obtained through the transformation

$$\boldsymbol{A}_2 = \boldsymbol{T}\boldsymbol{A}_1\boldsymbol{T}^{-1} \quad \boldsymbol{b}_2 = \boldsymbol{T}\boldsymbol{b}_1 \quad \boldsymbol{c}_2 = \boldsymbol{T}^{-t}\boldsymbol{c}_1 \quad \boldsymbol{d}_2 = \boldsymbol{d}_1$$

which yields

$$\boldsymbol{K}_2 = \boldsymbol{T}\boldsymbol{K}_1\boldsymbol{T}^t = \boldsymbol{T}\boldsymbol{K}_0\boldsymbol{T}^t$$
$$\boldsymbol{W}_2 = \boldsymbol{T}^{-t}\boldsymbol{W}_1\boldsymbol{T}^{-1} = \boldsymbol{T}^{-t}\boldsymbol{W}_0\boldsymbol{T}^{-1} \tag{11}$$

Now if we set $\boldsymbol{T} = \text{diag}\,(1, \omega_0, \omega_0^2, \ldots, \omega_0^{n-1})$, the realisation R_2 also corresponds to a CTDF structure. Using this $\boldsymbol{T}$ in eqn. 11, we obtain

$$[\boldsymbol{K}_2]_{ii}[\boldsymbol{W}_2]_{ii} = [\boldsymbol{K}_0]_{ii}[\boldsymbol{W}_0]_{ii} \qquad i = 1, 2, \ldots, n \tag{12}$$

which demonstrates that the output noise gain is invariant for CTDFs under the LP–LP transformation. Applying the bilinear transformation to the CTDF realisations R_0 and R_2, we obtain the corresponding TCTDF realisations $\hat{R}_0$ and $\hat{R}_2$. Now using eqn. 6 in eqn. 12 it can easily be shown that $[\hat{\boldsymbol{K}}_2]_{ii}[\hat{\boldsymbol{W}}_2]_{ii} = [\hat{\boldsymbol{K}}_0]_{ii}[\hat{\boldsymbol{W}}_0]_{ii}$, $i = 1, \ldots, n$, which proves that the roundoff noise is invariant under LP–LP transformation for TCTDF structures. As an example we choose the third-order LP transfer function given as

$$\hat{H}(z) = \frac{0{\cdot}01594(z + 1)}{z^3 - 1{\cdot}974861z^2 + 1{\cdot}556161z - 0{\cdot}453768}$$

For this transfer function the roundoff noise behaviour of the TCTDF and that of the DTDF will be compared. The values in Table 1 are the normalised values for the noise gain obtained as $G_N = (G/G_{min})$ where $G_{min} = (\sum_i^n \mu_i)^2 - \{\mu_i\}$ being the eigenvalues of $\boldsymbol{KW}$ which are invariant under the LP–LP transformation [3]. Table 1 shows clearly that the noise behaviour of the new structure is invariant under the LP–LP transformation, while that of the DTDF deteriorates drastically as $\alpha \rightarrow 1$. Therefore the TCTDF is shown to be a better structure for narrowband LP transfer functions.

Table 1 NORMALISED VALUES FOR NOISE

α	DTDF	TCTDF
−0·9	111·385	1·846
−0·5	1·072	1·846
0	6·041	1·846
0·5	239·088	1·846

4th October 1991

V. Tavsanoglu (*Department of Electrical and Electronic Engineering, South Bank Polytechnic, 103 Borough Road, London SE1 0AA, United Kingdom*)

References

1 TAVSANOGLU, V.: 'Explicit evaluation of K and W matrices for second-order digital filters'. Proc. 7th European Conf. on Circuit Theory and Design (ECCTD '85), Prague, Czechoslovakia, 1985, pp. 488–491

2 POWER, H. M.: 'Two applications of the matrix transformation $A = (B + I)(B - I)^{-1}$ in dynamical system theory', *Electron. Lett.*, 1968, **4**, pp. 479–481

3 MULLIS, C. T., and ROBERTS, R. A.: 'Roundoff noise in digital filters: frequency transformations and invariants', *IEEE Trans.*, 1976, **ASSP-24**, (6), pp. 538–550

PROPORTION OF PRIMES GENERATED BY STRONG PRIME METHODS

J. Shawe-Taylor

Indexing terms: Information theory, Cryptography

The numbers of primes generated by two prime generation methods are evaluated. The methods considered generate strong primes for cryptographic use in the RSA public-key cryptosystem. The results show that they generate sufficiently many primes to resist cryptanalytic attack by prime enumeration for the sizes of primes used to guard against factorisation attack.

Introduction: A method of generating primes is described by Shawe-Taylor [3] (henceforward the S-T method) which, though reliant on random techniques, outputs numbers which are guaranteed to be prime. Criticisms have been raised [2] that the technique excludes a large number of primes. Though this is undoubtedly true numerically, the purpose of this Letter is to show that, proportionally, the exclusion rate is small compared to the number of primes of the given size. The original paper describing a method for generating strong primes [1] (the Gordon method) also aims to exclude a number of primes and we use the techniques of this paper to estimate the proportion excluded in this case. Though the prime generation techniques are widely used to resist factorisation of the RSA modulus, as yet it has not been shown that there are sufficient primes to ensure that enumeration of all primes of a given size obtained by these techniques would not be an effective method of cryptanalytic attack. At the end we give estimates of the proportion of primes generated by the different methods for sizes of cryptographic interest.

The question of whether the method of prime generation could be used to assist the cryptanalyst is not addressed here and as far as we are aware remains open both for the Gordon strong prime generation method and the S-T adaption. This Letter is only concerned with the question of how many primes can be generated by the different methods, because were there to be too few, this would be a significant cryptographic weakness.

Frequency of primes: It is well known that the frequency of primes in integers of order N is approximated by $1/\ln N$ (throughout we will use ln for natural logarithm and log for logarithm to the base 2). The frequency of primes in odd integers is therefore $2/\ln N$.

Using this key observation we can estimate the number of primes that can be generated by the Gordon and S-T

methods. The two methods of generating primes work on the principle of generating smaller primes and combining them with further random bits to progressively create larger (and 'stronger' primes) resulting in a prime of the required size. We formalise this process in the following definition of a prime generation sequence.

(i) *Definition 1:* A k-stage prime generation sequence with parameters

$$(n_1, j_1), \ldots, (n_k, j_k)$$

is a procedure for generating a prime with n_k bits, by generating a sequence of primes with n_i bits, for $i = 1, \ldots, k$. At stage i a prime with n_i bits is selected from a set of the form

$$\{a + xb \mid 2^{j_i} \leq x \leq 2^{j_i+1} - 1\}$$

with a odd and b even, where (a, b) is determined in a (1-1) fashion by a subset of the primes already generated. For $i = k$, the pair (a, b) should be determined (directly or indirectly) in a (1-1) fashion by all the primes previously generated.

Note that the Gordon method is a four-stage prime generation sequence, whereas the S-T method is an $O(\log n)$-stage prime generation sequence for primes with n bits. As an example the final stage in both the Gordon and S-T method involves taking

$$b = 2rs$$

and

$$a = \begin{cases} u(r, s) & \text{if } u(r, s) \text{ is odd} \\ u(r, s) + rs & \text{if } u(r, s) \text{ is even} \end{cases}$$

where $u(r, s) = (s^{r-1} - r^{s-1}) \bmod rs$ and r and s are the primes generated at the previous two stages. The standard procedure for selecting a prime from the available set is to start from a point determined by a string of random bits and then step through the set until a prime is found. This actually means that primes which occur after a long sequence of nonprimes are more likely to be selected. An equally effective and fairer method would be to use another batch of random bits each time a nonprime is selected, assuming the random bits are cheaply available. A fuller discussion of the distribution of primes generated is given by Maurer [2]. We now give a lemma which gives an estimate of the number of primes that can be generated by a k-stage prime generation sequence.

(ii) *Lemma 1:* The number of distinct primes that can be generated by a k-stage prime generation sequence with parameters $(n_1, j_1), \ldots, (n_k, j_k)$ is approximately

$$\frac{2^t c^k}{\prod_{i=1}^{k} n_i}$$

where $c = 2/\ln 2$ and

$$t = \sum_{i=1}^{k} j_i$$

Proof of lemma 1: Suppose we are selecting a prime number from a set of odd integers with n bits of the form

$$\{a + xb \mid 2^j \leq x \leq 2^{j+1} - 1\}$$

Using the fact that primes are irregularly distributed, we can estimate the number of primes in the set as $2^{j+1}/\ln(2^n)$. All of these primes can be selected with different values of the random bits. Consider the ith stage of the generation process. By the above observation we can select approximately $2^{j_i+1}/\ln(2^{n_i})$ primes. Because the set of primes from which the last prime is generated is a (1-1) function of all the previous primes, the total number of different final primes that can be generated is the product of the numbers generated at each stage. This is the number given in the lemma statement. ■

Proposition 1: Using the Gordon [1] method of strong prime generation to generate an n bit strong prime the number of n bit primes that can be generated is approximately

$$\frac{2^n}{n(\ln 2)^4(n - \log n)^2[(n - \log n)/2 - \log(n - \log n) - 2]}$$

Proof of proposition 1: There are four primes generated in the Gordon method. In his terminology, they are usually denoted s, t, r and p. They have $(n - \log n)/2$, $(n - \log n)/2 - \log(n - \log n) - 2$, $(n - \log n)/2$ and n bits, respectively, and the numbers of random bits used in their generation are $(n - \log n)/2 - 1$, $(n - \log n)/2 - \log(n - \log n) - 3$, $\log(n - \log n) - 1$ and $\log n - 1$. Putting these into the formula of lemma 1 gives the result. ■

Corollary of proposition 1: The fraction of all n bit primes that are generated by the Gordon method is approximately

$$\frac{1}{(\ln 2)^3(n - \log n)^2[(n - \log n)/2 - \log(n - \log n) - 2]} \geq \frac{2}{[(n - \log n)\ln 2]^3}$$

Proposition 2: Using the S-T [3] generation method to generate a prime number with n bits provides more than

$$\frac{2^{n+1}}{(n \ln 2)^2[(n/2 - \log n)\ln 2]^{\log(n/2 - \log n)}}$$

primes.

Proof of proposition 2: Using the S-T method to generate a prime with n bits (we are not initially considering the final stage of ensuring a strong prime), we use a sequence of primes with $n_i = 2^i + 1$ bits, for $i = i_0, \ldots, k - 1$, and $n_k = n$, where i_0 is some suitably small value where we can check directly for primes (for example $i_0 = 4$) and $k = \lceil \log(n - 1) \rceil$. The number of random bits used in the generation at stage i is $n_{i+1} - n_i - 1$, for $i = i_0 + 1, \ldots, k$, whereas for $i = i_0$, 3 bits are used. Hence $t = n - \lceil \log(n - 1) \rceil + 1 = n - k + 1$ and

$$\prod_{i=i_0}^{k} n_i < \frac{n^{k-3} 2^{k(k-1)/2 - 6}}{2^{(k-1)(k-4)}}$$

because $n_i < n/2^{k-i-1}$, for $i = i_0, \ldots, k - 1$. Using the fact that $k + 4 > \log n$ we obtain

$$\prod_{i=i_0}^{k} n_i < n^{(k-5)/2}$$

Putting these figures into the formula of lemma 1 we obtain a lower bound for the number of primes

$$\frac{2^n}{4(\ln 2)^{k-3} n^{(k-5)/2}} \geq \frac{2^n}{4(\ln 2)^{\lceil \log(n-1) \rceil - 3} n^{\lceil \log(n-1) \rceil - 4)/2}} \geq \frac{2^n}{(n \ln 2)^{(\log(n-1) - 4)/2}}$$

In the strong prime generation we first generate two primes using the above method of size $n/2 - \log n - 2$ and $n/2 - \log n + 1$ bits. Using these primes we generate the prime r of size $n/2 + 1$ bits using $\log n - 1$ random bits and finally the prime p of size n bits and using $\log n$ random bits. Incorporating these extra values into the formula of lemma 1 we obtain a lower bound of

$$\frac{2^{n-2} c^2}{0{\cdot}5 n^2[(n/2 - \log n)\ln 2]^{\log(n/2 - \log n)}}$$

which gives the result in the proposition statement. ■

Corollary 2: The fraction of all n bit primes that are generated by the S-T procedure is approximately

$$\frac{2}{(n \ln 2)[(n/2 - \log n) \ln 2]^{\log (n/2 - \log n)}}$$

Conclusion: Table 1 shows the estimated number of primes and the fraction generated by the two strong prime generation methods for sizes of interest in RSA cryptography.

Table 1: ESTIMATED NUMBER OF PRIMES AND FRACTION GENERATED USING GORDON AND S-T ADAPTION METHODS

Size in bits	Number of primes	Fraction generated	
		Gordon method	S-T adaption
128	3.84×10^{36}	1.94×10^{-2}	1.10×10^{-11}
256	6.53×10^{74}	8.77×10^{-3}	6.18×10^{-16}
384	1.48×10^{113}	5.64×10^{-3}	1.13×10^{-18}
512	3.78×10^{151}	4.16×10^{-3}	9.96×10^{-21}

Though the fractions are small, particularly in the case of the S-T method, the actual numbers of primes generated by either method are well beyond the powers of enumeration provided we take primes of size 128 bits or more.

14th November 1991

J. Shawe-Taylor (*Department of Computer Science, Royal Holloway and Bedford New College, Egham Hill, Egham, Surrey TW20 0EX, United Kingdom*)

References

1 GORDON, J.: 'Strong RSA keys', *Electron. Lett.*, 1984, **20**, pp. 514–516

2 MAURER, U. M.: 'Some number-theoretic conjectures and their relation to the generation of cryptographic primes'. To be published in Proc. 2nd IMA Conference on Cryptography and Coding, Cirencester, 1989, MITCHELL, C. J. (Ed.)

3 SHAWE-TAYLOR, J. S.: 'Generating strong primes', *Electron. Lett.*, 1986, **22**, pp. 875–877

SOME NEW RUNLENGTH CONSTRAINED BINARY MODULATION CODES WITH ERROR-CORRECTING CAPABILITIES

A. S. J. Helberg and H. C. Ferreira

Indexing terms: Information theory, Codes and coding, Error-correcting codes

Five new combined error-correcting (d, k) codes for use on bandwidth limited channels are presented. These new codes are compared to known codes with similar parameters. The error behaviour of the new codes after Viterbi decoding on the binary symmetric channel is evaluated by simulation. The power spectral densities are also measured and the results presented.

Recent investigation [1, 2] has shown that under certain conditions a combined code (i.e. a constrained code that has error correcting capabilities) performs better than a concatenated coding scheme (i.e. a coding scheme where the constraints are met with an inner constrained code and the error-correction is performed by an outer code.). We present five new combined codes which were developed using different techniques.

The first two codes which we present were developed using an extension of the construction technique for constrained codes as proposed by Franaszek [3]. This extension incorporates some distance building characteristics into the constrained code. In this way the concatenability of the constrained sequences is taken care of by the original finite state transition diagram that describes the constraints. The resulting codes are shown in Figs. 1 and 2.

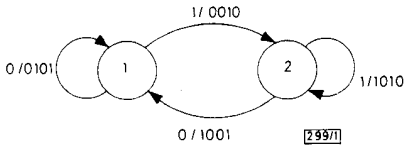

Fig. 1 *Finite state machine for rate $R = 1/4$, $(d, k) = (1, 2)$, $d_\infty = 3$ code*

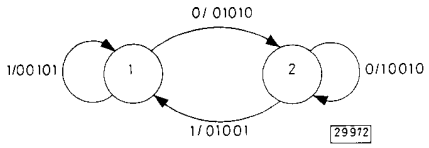

Fig. 2 *Finite state machine for rate $R = 1/5$, $(d, k) = (1, 2)$, $d_\infty = 6$ code*

The second approach which we used is to map constrained symbols onto a shift register graph to enhance the length before remergence of the distance building paths. Great care must be taken to ensure that the constraints are preserved when using this method. The code we constructed using this technique is shown in Fig. 3 and in Fig. 4 we show a code that was constructed similarly using a three state graph to enhance the length before remergence of the distance building paths.

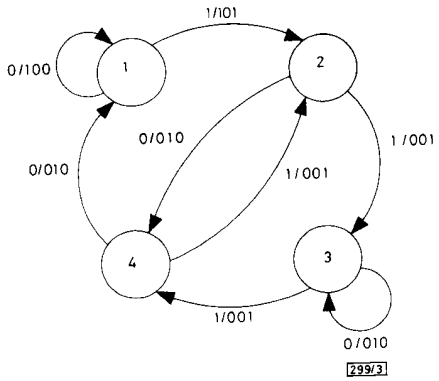

Fig. 3 *Finite state machine for rate $R = 1/3$, $(d, k) = (1, 3)$, $d_\infty = 4$ code*

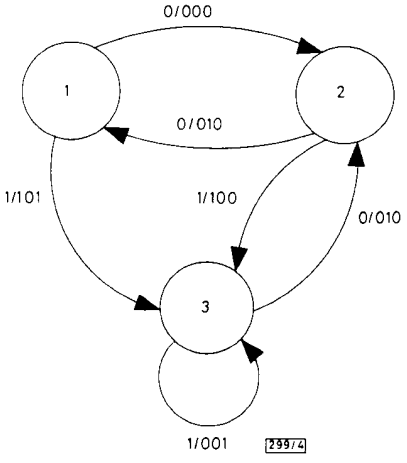

Fig. 4 *Finite state machine for rate $R = 1/3$, $(d, k) = (1, 5)$, $d_\infty = 3$ code*