A Modern Rotor Machine

Ross Anderson

Computer Laboratory, Pembroke Street, Cambridge CB2 3QG
Email: rja14@cl.cam.ac.uk

1 Introduction

The cryptologic literature contains a lot of material on both shift register and rotor machine systems. It is natural to wonder whether these two types of mechanism can be combined in one robust design.

During the 1980's, research in clock-controlled shift registers was inspired by rotor machines of the Hagelin type [G1], and a t-shirt appeared at a Crypto conference with a design consisting of a shift register and five rotors [D]. More recently, one writer proposed to filter three linear generators A,B, and C with two permutations π and ρ in order to get a keystream $K = A + \pi(B + \rho C)$ [F]; and a rump session paper at Eurocrypt this year showed that if a shift register sequence is filtered through a permutation which acts on m-bit symbols, then a correlation attack will need m times as many bits as before [B1].

In this article, we propose a different combination, which appears to be the simplest yet; it consists of an Enigma-type rotor machine (without the *Umkehrwalze*), in which three wired rotors which each implement a random permutation on 256 symbols are turned by a linear feedback shift register. It is straightforward to implement and fairly fast; yet, provided the rotors are kept secret, and the shift register is too long for its state to be guessed, it appears to resist all known attacks.

2 Specification

We will assume a basic familiarity with rotor machines; this can be acquired from books such as [DK] or [W]. Let P_1, P_2 and P_3 be three random permutations on 256 letters (we will discuss later how to construct them from a key). If we imagine each of them as a wired rotor, and turned through an offset of k_1, k_2 and k_3 respectively, then the conducting paths through them will transform the input byte x into the output byte c where

$$c = P_3(k_3 + P_2(k_2 + P_1(k_1 + x) - k_1) - k_2) - k_3 \quad (1)$$

Classical attacks on rotor machines depend on knowing how the rotors move, and exploiting this to find short sequences of text which were enciphered with all but one of the rotors in the same relative position. The Enigmas had a fairly simple stepping relationship between the wheel positions k_i, and the rotor wirings P_i were changed infrequently (and were usually known from equipment

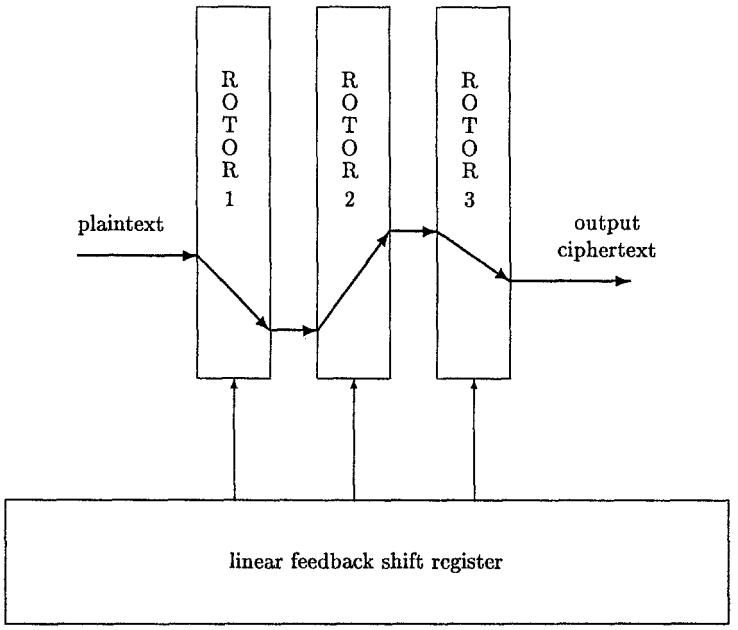

Fig. 1. A shift register driving three wired rotors

captures) [W]. Where wirings were unknown, there was often a fair amount of known plaintext, which could be used together with guessed rotor motions to deduce the permutation [K2]. Later rotor machines such as the NATO KL-7 used even more complex motions [DK].

One notorious weakness of the Enigma was the *Umkehrwalze*, a reflection rotor which ensured that no letter was ever enciphered to itself [W]. This feature was perpetuated in the the unix `crypt()` function, where one of the two permutations is self-inverse, and facilitates attacking this function [RW]. In general, it is bad practice to give an opponent any information about the cycle structure of the permutations in use.

We therefore propose that the three permutations P_i should be chosen at random from S_{256}, and the rotor positions k_i should be the three least significant bytes of the state of a shift register of at least 80 bits length whose primitive characteristic polynomial has at least 20 well-distributed nonzero coefficients. Shift register theory ensures that each possible rotor position will occur with equal probability [S].

This register must be fairly long; if its contents could be guessed, then with enough known plaintext we could use traditional methods to reconstruct the rotor wiring. Note also that if we used a sparse or bunched feedback polynomial such as $x^{127} + x + 1$, then, as in [A], we could extrapolate a guess of part of the initial state; for example, guessing 40 bits would give us 128 24-bit rotor positions, and 60 bits would give us 648 positions, which might be enough to reconstruct some permutation values using the method in [K2].

However, if we cannot guess the rotor motions, then the established rotor

analysis techniques do not work. In our proposed design, the analyst does get one piece of side information: that the current k_1, the last k_2 and the previous k_3 are equal. However, it is unclear how this helps him.

If an attacker knew the rotor wiring, then he would have a (small) chance of being able to mount a correlation attack on the shift register. Corrrelation attack techniques were used as early as 1943 on pinwheel rotor machines [G2]; wired rotor machines are less susceptible, but even here each of the eight output bits could be regarded as a Boolean function of 24 inputs, and one could search for a transformation of the output which was correlated with the driving sequence. However, when the message key determines both the register's state and the rotor wirings, the best way forward would seem to be that used in the wartime 'Heath Robinson' device [G2]: trying a lot of random Boolean functions of the ciphertext in the hope of stumbling across a correlation. Given the size of our proposed device, this would appear to have a negligible probability of success.

3 Implementation

The implementer will have to provide some means of expanding the key to provide the three permutations. The obvious approach would be to use a safe but slow block cipher, and then proceed as in Knuth [K3]; but if key set-up time is important, then there is a trick, due to the author and Roger Needham, which lets us use as few expanded keybits as possible ($nlogn$). This is based on the observation that sorting and shuffling are in some sense inverse operations; we can use our favourite sorting algorithm on the string (0,1,2,...,255) in 'reverse', in the sense that whenever the original sort would have compared two elements, we merely take our next keybit instead of the result of that comparison.

The resource required once keys have been set up is small. Each byte encryption will take three reads, in addition to the i/o and the operations needed to refresh the shift register. By suitably intercalating these tasks, we can avoid pipeline stalls and thus expect that this scheme will be as fast as any shift register system. Having only one, rather than two, shift registers, it should be significantly faster than generators using two or more shift registers.

4 Conclusion

Combining a keystream with plaintext using rotors rather than an exclusive-or gate clearly gives protection against correlation attacks. What is not clear is precisely how much protection we get, and how this varies with the number and configuration of the rotors.

In a production system, one might well build in an extra margin of safety against correlation attacks, such as by replacing the linear shift register with a clock controlled register, making the feedback polynomial key-dependent, and choosing rotor permutations which are almost bent. However, such measures involve significantly more implementation effort, and the algorithm proposed here is in any case intended to be a challenge as much as a production system.

There are indications in the literature [B2] that some countries used electronic rotor machines in military systems (although they are not very forthcoming about the algorithm details). It is therefore possible that some intelligence agencies may have developed tricks for solving systems of the type described above. If so, then these could have wider consequences; they might provide a bridge between the linear theory, as developed for shift registers, and the more group-theoretic techniques of classical rotor machine analysis. Given the wide use permutations in cryptography, such results might be quite valuable.

We therefore commend our modern rotor machine to the research community. It is conceptually simple, fast and easy to implement; it builds directly on a fair amount of existing theory; and it appears to be the simplest combination of rotors and shift registers which defeats all the known attacks on either type of system. It should therefore be a worthwhile target for analysis, and in order to lend some sparkle to the challenge, a reward is offered of a bottle of Veuve Clicquot for the first practical attack. For the sake of definiteness, we mean by this that given 64Kbytes of chosen plaintext, then for at least 1% of keys the attack should recover a key with less than 2^{45} machine operations.

References

[A] RJ Anderson, "Faster attack on certain stream ciphers", in *Electronics Letters* **29** no 15 (22nd July 1993) pp 1322 - 1323

[B1] L Brynielsson, "Information Leakage through a permutator", at the 1993 Eurocrypt rump session

[B2] Commodore AP Burgers, "State Policy and Legal Aspects on the Use of Communication Security in the Private Sector", in *Proc. Infosec 89* pp 1 - 15

[D] W Diffie, comment from the floor of the workshop

[DK] CA Deavours, L Kruh, *'Machine Cryptography and Modern Cryptanalysis'*, Artech House, Dedham 1985

[F] AJ Filipski, "How (un)secure is using a psuedo-random generator and XOR?". Message `1993May11.202241.489@gtx.com` in Usenet newsgroup `sci.crypt`

[G1] D Gollmann, "Cryptanalysis of Clock Controlled Shift Registers" in *these proceedings*

[G2] J Good, "Enigma and Fish", in *Codebreakers* (ed FH Hinsley and A Stripp) OUP 93 pp 149 - 166

[K1] D Kahn, *'The Codebreakers'*, Macmillan, New York 1967

[K2] AG Konheim, *'Cryptography - a Primer'*, Wiley 1981, pp 190 ff

[K3] D Knuth, *'The Art of Computer Programming'* v 3

[RW] JA Reeds, PJ Weinberger, "File Security and the UNIX System Crypt Command", in *AT&T Bell Laboratories Technical Journal* **63** no 8 (October 1984) pp 1673 - 1683

[S] E Selmer, *'Linear Recurrence Relations Over Finite Fields'*, University of Bergen 1966

[W] G Welchman, *'The Hut Six Story - Breaking the Enigma Codes'*, McGraw-Hill, New York 1982