FAST CORRELATION ATTACKS ON STREAM CIPHERS

(Extended Abstract)

Willi Meier [1)] Othmar Staffelbach [2)]

[1)] HTL Brugg-Windisch
CH-5200 Windisch, Switzerland

[2)] GRETAG Aktiengesellschaft
Althardstr. 70, CH-8105 Regensdorf
Switzerland

For proofs and further explanations of the results presented herein we refer the reader to the full paper ([1]). A description of the cryptanalytic algorithms is appended.

I. Extended Abstract

A common type of running key generator employed in stream cipher systems consists of n (mostly maximum-length) binary linear feedback shift registers (LFSR's) whose output sequences are combined by a nonlinear Boolean function f. The output of several combining functions previously proposed in the literature is known to be correlated to some input variables with probabilities p up to 0.75 (this holds, e.g. for the generators of Geffe, Pless, or Bruer). These generators have been broken in [2] for LFSR-lengths k < 50 (roughly), according to the computational complexity of the attack (based on an exhaustive search over all phases of the LFSR). But also other generators, e.g. certain types of multiplexed sequence generators, are known to be correlated to LFSR-components. In fact any generator having such correlations may be vulnerable to a correlation attack.

Let the output sequence $\underline{z}$ of a running key generator be correlated to a linear feedback shift register sequence (LFSR-sequence) $\underline{a}$ with correlation probability $p > 0.5$. Then two new correlation attacks (algorithms A and B) are presented to determine the initial digits of $\underline{a}$, pro-

C.G. Guenther (Ed.): Advances in Cryptology - EUROCRYPT '88, LNCS 330, pp. 301-314, 1988.

vided that the number t of feedback taps is small ($t < 10$ if $p \leq 0.75$). The computational complexity of algorithm A is of order $O(2^{ck})$, where k denotes the length of the LFSR and $c < 1$ depends on the input parameters of the attack, and algorithm B is polynomial (in fact, even linear) in the length k of the LFSR. These algorithms are much faster than an exhaustive search over all phases of the LFSR, and are demonstrated to be successful on shift registers of considerable length k (typically $k = 1000$). On the other hand, for correlation probabilities $p \leq 0.75$ the attacks are proven to be infeasible on long LFSR's if they have a greater number of taps (roughly $k \geq 100$ and $t \geq 10$).

In order to set out our results in more detail, suppose that N digits of the output sequence $\underline{z}$ are given, and correlated to an LFSR-sequence $\underline{a}$, produced by a LFSR with t taps. We assume that the feedback connection is known. Observe that this is no essential restriction as there is only a very limited number of maximum-length feedback connections with few taps. Hence an exhaustive search over all primitive feedback connections is possible.

The sequence $\underline{z}$ may be viewed as perturbation of the LFSR-sequence $\underline{a}$ by a binary asymmetric memoryless noise source (with Prob(0) = p). For the purpose of reconstructing the LFSR-sequence $\underline{a}$ from $\underline{z}$ the following principle is essential to the algorithms: Every digit a_n of $\underline{a}$ satisfies several linear relations derived from the basic feedback relation, all of them involving t other digits of $\underline{a}$. By substituting the corresponding digits of $\underline{z}$ in these relations, we obtain equations for each digit z_n, which either may or may not hold. To test whether $z_n = a_n$, we count the number of all equations which turn out to hold for z_n. Then the more of these equations hold, the higher is the probability for z_n to agree with a_n. This can be justified by a statistical model, computing the corresponding conditional probabilities.

On the basis of this idea, we roughly outline algorithm A: We use the test to search for correct digits (i.e. digits z_n with $z_n = a_n$). This is done by selecting those digits which satisfy the most equations. In this way we obtain an estimate of the sequence $\underline{a}$ at the corresponding positions. Under favourable conditions these digits have high probability of being correct, which means that only a slight modification of our estimate is necessary. This results in a considerably reduced exhaustive search to rule out sufficiently many correct digits, in order to

determine the LFSR-sequence $\underline{a}$ by solving linear equations.

We can give precise conditions under which this procedure is successful, and determine its computational complexity, which in general is of order $O(2^{ck})$, where $c < 1$ is a function of t, p and N/k . To illustrate this estimate we mention that for $t = 2$ taps, $N/k = 10^6$, and $p \geq 0.6$, the number c is smaller than 0.25, and for $p > 0.67$ Table 1 shows that c is below 0.001. This is a considerable improvement compared to exhaustive search, where $c = 1$. On the other hand, for large t ($t \geq 16$) our estimate shows, that c comes very close to H(p), where H(p) denotes the binary entropy function. This proves that algorithm A for large t gives no advantage over (a modified) exhaustive search.

p\t	2	4	6	8	10	12	14	16	∞
0.51	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.53	0.976	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.997
0.55	0.870	0.992	0.993	0.993	0.993	0.993	0.993	0.993	0.993
0.57	0.642	0.982	0.986	0.986	0.986	0.986	0.986	0.986	0.986
0.59	0.362	0.963	0.976	0.976	0.976	0.977	0.977	0.977	0.977
0.61	0.132	0.926	0.963	0.965	0.965	0.965	0.965	0.965	0.965
0.63	0.039	0.856	0.945	0.950	0.951	0.951	0.951	0.951	0.951
0.65	0.007	0.734	0.917	0.932	0.934	0.934	0.934	0.934	0.934
0.67	0.001	0.555	0.875	0.910	0.914	0.915	0.915	0.915	0.915
0.69	0.000	0.327	0.805	0.880	0.891	0.893	0.893	0.893	0.893
0.71	0.000	0.150	0.692	0.836	0.863	0.868	0.868	0.869	0.869
0.73	0.000	0.043	0.515	0.768	0.825	0.838	0.841	0.841	0.841
0.75	0.000	0.009	0.311	0.660	0.771	0.800	0.808	0.811	0.811

Table 1: $c(p,t,N/k)$ for $N/k = 10^6$

In algorithm B we do not search for the most reliable digits. Instead we take into account all digits, together with their probabilities of being correct. A priori, with probability p a digit of $\underline{z}$ agrees with the corresponding digit of $\underline{a}$. Now to each digit z_n of $\underline{z}$ we assign a new probability p*, which is the probability for $z_n = a_n$, conditioned on the number of equations satisfied. This procedure can be iterated with the varied new probabilities p* as input to every round. After a few rounds, all those digits of $\underline{z}$ are complemented whose probability p* is lower

than a certain threshold. Under suitable conditions we can expect that the number of incorrect digits decreases. In this case we restart the whole process several times, with the new sequence in place of $\underline{z}$, until we end up with the original LFSR-sequence $\underline{a}$.

To obtain conditions under which algorithm B succeeds, a function $F(p,t,N/k)$ is introduced to measure the correction effect. Thus if $F(p,t,N/k) \leq 0$ there is no correction effect and algorithm B will not be able to reproduce the LFSR-sequence $\underline{a}$. Therefore we get a definite limit to this attack (which is attained for $t \geq 10$, if $p \leq 0.75$). In the other direction, investigations of $F(p,t,N/k)$ show, that for $t = 2$ or $t = 4$ taps algorithm B still remains effective for correlation probabilities quite close to 0.5 (cf. Table 2). This implies in particular that a LFSR with two feedback taps is completely breakable if its output $\underline{a}$ shows correlation to a known sequence $\underline{z}$. The striking efficiency of algorithm B, as observed in numerous experiments, is explained by the fact that its computational complexity is of order $O(k)$ (i.e. linear in the length k of the LFSR, for fixed t, p and N/k).

For given t and $d = N/k$ Table 2 shows the value $p = p(t,d)$ with $F(p,t,d) = 0$. $p(t,d)$ turns out to be the limit probability where algorithm B may still be successful

d/t	2	4	6	8	10	12	14	16	18
10	0.584	0.739	0.804	0.841	0.864	0.881	0.894	0.904	0.912
10^2	0.533	0.673	0.750	0.796	0.827	0.849	0.865	0.878	0.890
10^3	0.521	0.648	0.727	0.776	0.809	0.833	0.852	0.866	0.878
10^4	0.514	0.629	0.709	0.760	0.795	0.821	0.841	0.856	0.869
10^5	0.511	0.620	0.699	0.752	0.787	0.815	0.834	0.850	0.863
10^6	0.509	0.612	0.692	0.745	0.782	0.809	0.830	0.846	0.860
10^7	0.508	0.605	0.684	0.738	0.775	0.803	0.825	0.842	0.855
10^8	0.507	0.601	0.680	0.733	0.771	0.800	0.821	0.838	0.852
10^9	0.506	0.597	0.676	0.729	0.768	0.797	0.818	0.836	0.850
10^{10}	0.505	0.592	0.671	0.725	0.764	0.793	0.815	0.832	0.847

Table 2: p with $F(p,t,d) = 0$

Algorithms A and B enable attacks on LFSR's of considerable length (e.g. k = 1000 or greater) with software implementation. However, a comparison shows that algorithm A is preferable if c << 1 and p is near 0.75, whereas algorithm B becomes more efficient for probabilities p near 0.5. (Simulations of algorithm B have shown to be successful in attacks with p = 0.55 even on a personal computer).

The methods developed for algorithms A and B allow several generalizations and conclusions. To prevent attacks based on these methods, suitable precautions are necessary. This leads to new design criteria for stream ciphers:

1. Any correlation to a LFSR with less than 10 taps should be avoided.

2. There should be no correlation to a general LFSR of length shorter than 100 (especially when the feedback connection is assumed to be known).

It is remarkable that the importance of the number of LFSR taps for the correlation analysis was not recognized in cryptologic literature so far.

II. Appendix: Description of the Algorithms

In this appendix we give a brief outline of the algorithms. Proofs and further explanations are contained in [1].

II.1. Algorithm A

Suppose that N digits of the sequence $\underline{z}$, the length k of the LFSR with t taps as well as the correlation probability p are given.

Our method exploits the linear relations of the LFSR-sequence $\underline{a}$ to find correct digits, i.e. digits with $z_n = a_n$. Linear relations can be described in terms of their feedback polynomials. By iterated squaring of the feedback polynomial, a variety of linear relations is generated for every digit a_n, all of them involving t other digits of $\underline{a}$. The

average number m of relations obtained in this way can be computed as (cf. [1])

$$m = m(N,k,t) = \log_2(\frac{N}{2k})\ (t + 1) \tag{1}$$

The probability p^* for $z_n = a_n$, given that h of m relations are satisfied, is

$$p^* = \frac{p\ s^h(1-s)^{m-h}}{p\ s^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} \tag{2}$$

where s = s(p,t) can be comuted using the recursion

$$s(p,t) = p\ s(p,t-1) + (1-p)(1 - s(p,t-1)), \tag{3}$$
$$s(p,1) = p.$$

Moreover, the probability that a digit z_n satisfies at least h of these m relations is given by

$$Q(p,m,h) = \sum_{i=h}^{m} \binom{m}{i} (p\ s^{\,i}(1-s)^{\,m-i} + (1-p)(1-s)^{\,i} s^{\,m-i}) \tag{4}$$

and the probability that $z_n = a_n$ and that at least h of m relations are satisfied

$$R(p,m,h) = \sum_{i=h}^{m} \binom{m}{i}\ p\ s^{\,i}(1-s)^{\,m-i} \tag{5}$$

Thus the probability for $z_n = a_n$, given that at least h of m relations are satisfied, is the quotient T(m,p,h) = R(p,m,h)/Q(p,m,h). These formulas show that with increasing m we have more freedom to choose a suitable h such that at the same time the two probabilities Q(p,m,h) and T(p,m,h) will be sufficiently large for an attack. The following examples illustrate these facts.

Example 1: Assume that $\underline{z}$ has length N = 5000 correlated with probability p = 0.75 to a LFSR of length k = 100 having t = 2 feedback taps. Hence in the average we obtain m = 12 relations to test the digits of $\underline{z}$. To determine the optimum number h of relations to be satisfied we generate the following table:

h = # of relations satisfied	new prob. p^*	Q(p,m,h)	1 - T(p,m,h)
12	0.9993	0.002666	0.000725
11	0.9980	0.021890	0.001855
10	0.9944	0.085554	0.004618
9	0.9847	0.214141	0.011040
8	0.9586	0.392461	0.024840
7	0.8929	0.576251	0.051090
6	0.7500	0.729409	0.092856
5	0.5192	0.843183	0.145199
4	0.2800	0.922315	0.194519
3	0.1228	0.970429	0.228367
2	0.0480	0.992595	0.244528
1	0.0178	0.999106	0.249335
0	0.0065	1.000000	0.250000

Table 3

A digit that satisfies h = m = 12 relations has the highest probability $p^* = 0.9993$ to be correct. But according to Table 3 we can only expect $0.00266 \cdot 5000 \approx 13$ digits to satisfy this condition which obviously do not determine the phase of the LFSR-sequence. However $h \geq 11$ relations are expected to hold for $0.02189 \cdot 5000 \approx 109$ digits, hence a number which is greater than k = 100. Furthermore the entry in the 4th column shows that $0.001855 \cdot 109 = 0.2 < 1$ digits among these are expected to be wrong. Thus we can expect to have already found more than k = 100 correct digits. In fact this can be confirmed experimentally.

Example 2: We extend the above example to the situation N = 25000, k = 500, and let p = 0.75 and t = 2 unaltered. Thus again m = 12, and Table 3 also applies to this case. Hence $h \geq 11$ relations hold for $0.02189 \cdot 25000 \approx 547 > k$ digits. However $0.001855 \cdot 547 \approx 1$ digit among these may be wrong. Thus in order to find at least k = 500 correct digits one would have to perform a number of trials of magnitude 500, using the correlation method as referred to in [2].

In the general case the algorithm proceeds as follows.

Algorithm A

Step 1: Determine m according to formula (1)

Step 2: Find the maximum value of h such that $Q(p,m,h)\cdot N \geq k$ (e.g. by generating a table similar to Table 1). Then the average number r of errors is determined by $r = (1 - T(p,m,h))\cdot k$.

Step 3: Search for the digits of $\underline{z}$ satisfying at least h relations and use these digits as a reference guess I_0 of $\underline{a}$ at the corresponding index positions.

Step 4: Find the correct guess by testing modifications of I_0 having Hamming distance 0,1,2, ... , by correlation of the corresponding LFSR-sequence with the sequence $\underline{z}$

Under favorite conditions (cf. Example 1, where $r << 1$) step 4 is not necessary. In general it can be shown that the computational complexity of algorithm A is of order $O(2^{H(\theta)})$, where $\theta = r/k$ and where $H(x)$ denotes the binary entropy function (cf. [1]).

II.2. Algorithm B

Table 3 shows that the conditional probability p^* is small if a digit satisfies only a few relations, and hence tends to be incorrect. Roughly speaking this observation leads us to the following method of attack: Any digit of the sequence $\underline{z}$ is complemented if it satisfies less than a certain number of relations. Under favourable conditions we can expect that the "corrected" sequence has less digits differing from the LFSR-sequence $\underline{a}$.

An alternative and better approach is to leave the whole sequence $\underline{z}$ unchanged in the first instance and to assign instead the new probability p^* to every digit. This allows to iterate this process with varied new probabilities p^* at each round. After a few rounds, the wrong digits tend to have low, and the correct ones tend to have high probability.

This gives us a refined criterion to correct the sequence $\underline{z}$ by complementing the digits which have a probability p^* lower than a suitable threshold p_{thr}. Then we can restart the whole process with the new sequence in place of $\underline{z}$, this time assigning the original probability to every digit. The intuitive idea is to repeat the procedure until we end up by reproducing the LFSR-sequence $\underline{a}$.

To give a more precise description we need some additional formulas for computing probabilities:

a) The probability that a digit z_n satisfies at most h of m relations

$$U(p,m,h) = \sum_{i=0}^{h} \binom{m}{i} \left(p\, s^{\,i} (1-s)^{\,m-i} + (1-p)(1-s)^{\,i} s^{\,m-i} \right) \qquad (6)$$

b) The probability that $z_n = a_n$ and that at most h of m relations are satified

$$V(p,m,h) = \sum_{i=0}^{h} \binom{m}{i} \; p\, s^{\,i} (1-s)^{\,m-i} \qquad (7)$$

c) The probability that $z_n \neq a_n$ and that at most h of m relations are satisfied

$$W(p,m,h) = \sum_{i=0}^{h} \binom{m}{i} (1-p)\, (1-s)^{\,i} s^{\,m-i} \qquad (8)$$

With regard to the described method to correct digits if they satisfy at most h relations, these formulas enable us to compute the total number of digits of $\underline{z}$ changed by

$$U(p,m,h) \cdot N \qquad (9)$$

Moreover the number of erroneously changed digits is

$$V(p,m,h) \cdot N \qquad (10)$$

and the number of correctly changed digits is

$$W(p,m,h)\cdot N \tag{11}$$

Thus the increase of correct digits is the difference of the values in (11) and (10), and the relative increase is

$$I(p,m,h) = W(p,m,h) - V(p,m,h) \tag{12}$$

Next we determine the value $h = h_{max}$ such that $I(p,m,h)$ is maximum for given p and m. To this purpose we generate a table as illustrated in the following example:

Example 3: As in example 1 let N = 5000, p = 0.75, t = 2 and k = 100. Then m = 12 and we obtain the table

h = # of relations satisfied	new prob. $p^*(p,m,h)$	U(p,m,h)	I(p,m,h)
0	0.0065	0.000894	0.000882
1	0.0178	0.007405	0.007161
2	0.0480	0.029571	0.027201
3	0.1228	0.077685	0.063500
4	0.2800	0.156817	0.098325
5	0.5192	0.270591	0.093949
6	0.7500	0.423749	0.017370
7	0.8929	0.607539	-0.127036
8	0.9586	0.785859	-0.290587
9	0.9847	0.914446	-0.415237
10	0.9944	0.978110	-0.478191
11	0.9980	0.997334	-0.497337
12	0.9993	1.000000	-0.500000

Thus we see that $I(p,m,h)$ is maximum for $h_{max} = 4$ relations. Under these conditions 1250 digits are expected to be wrong. Carrying out the correction with respect to 4 relations, $0.1568\cdot 5000 \approx 793$ digits are complemented. According to the fourth column, the number of wrong digits decreases by $0.0983\cdot 5000 \approx 492$ from 1250 to 758 digits.

For our (alternative) refined method as described above, taking p^* into account, we need a appropriate probability threshold. An optimum correction effect is obtained with the choice

$$p_{thr} = \frac{1}{2}\,(p^*(p,m,h_{max}) + p^*(p,m,h_{max}+1)) \tag{13}$$

After the first round the expected number N_w of digits with p^* below p_{thr} is

$$N_{thr} = U(p,m,h_{max}) \cdot N \qquad (14)$$

Basically, the whole attack will swap between two phases:

I. A computation phase assigning the new probability p^* to every digit of $\underline{z}$.
II. A correction phase complementing those digits with p^* below p_{thr} and resetting the probability of each digit to the original value p.

Phase I can be iterated. To this purpose, formula (2) for $s(p,t)$ has to be generalized to the situation where each of the t digits may have different probabilities $p_1, p_2, \ldots, p_t$:

$$s(p_1,..,p_t,t) = p_t s(p_1,..,p_{t-1},t-1) + (1-p_t)(1-s(p_1,..,p_{t-1},t-1))$$
$$s(p_1,1) = p_1 \qquad (15)$$

This generalization carries over to all other formulas, in particular to formula (2) for p^*.

It is natural to iterate phase I until there are enough digits with p^* below p_{thr}. However, after a few iterations a strong polarization can be observed between digits having probability p^* either very close to 0 or very close to 1. Apart from a few digits, this polarization tends to become stable, which means that we needn't iterate phase I any longer. This gives us another criterion to terminate phase I after a limited number α of iterations. (In many cases $\alpha = 5$ is a suitable choice.) Based on these ideas we are now prepared to formulate algorithm B.

Algorithm B

Step 1: Determine m according to formula (1).

Step 2: Find the value of $h = h_{max}$ such that $I(p,m,h)$ is maximum. If $I_{max} = I(p,m,h_{max}) \leq 0$ there will be no correction effect in phase I which means that the attack fails. If $I_{max} > 0$ compute p_{thr} and N_{thr} according to (13) and (14), else terminate.

Step 3: Initialize the iteration counter i = 0

Step 4: For every digit of $\underline{z}$ compute the new probability p^* (cf. (2) and (15)) with respect to the individual number of relations satisfied (phase I). Determine the number N_w of digits with $p^* < p_{thr}$.

Step 5: If $N_w < N_{thr}$ or $i < \alpha$ increment i and go to step 4

Step 6: Complement those digits of $\underline{z}$ with $p^* < p_{thr}$ and reset the probability of each digit to the original value p (phase II).

Step 7: If there are digits of $\underline{z}$ not satisfying the basic feedback relation go to step 3.

Step 8: Terminate with $\underline{a} = \underline{z}$.

Under conditions for which algorithm B succeeds, its computational complexity is of order O(k), i.e. linear in the length k of the LFSR. To obtain such conditions a function F(p,t,N/k) is introduced in [1] to measure the correction effect ($F(p,t,N/k) = I(p,m,h_{max})\cdot(N/k)$, for details we refer to [1]). If $F(p,t,N/k) \leq 0$ algorithm B definitely fails.

We conlude with a simulation of algorithm B.

Example 4: We consider the following situation; N = 20,000, k = 200, t = 4 and p = 0.60. Then N/k = 100 and F(p,t,N/k) turns out to be 0.615. The parameters of the algorithm B can be computed as $p_{thr} = 0.481$, $N_{thr} = 1154$. Thus 1154 digits are expected to be changed in the first iteration resulting in a decrease of wrong digits by $0.615\cdot 200 = 123$. The following table shows the intermediate results after each step. The terms round and iteration refer to the outer loop and the inner loop, respectively. The entry in the third column always indicates the decrease of wrong digits if phase II had been applied.

	# of digits with $p^* < p_{thr}$	# of wrong digits with $p^* < p_{thr}$	decrease of wrong digits	# of wrong digits after phase II
round 1				
iteration 1	1784	998	212	7998
phase II	0	0	0	7786
round 2				
iteration 1	264	151	38	7786
iteration 2	1354	838	322	7786
phase II	0	0	0	7464
round 3				
iteration 1	133	80	27	7464
iteration 2	880	601	322	7464
iteration 3	2364	1537	710	7464
phase II	0	0	0	6754
round 4				
iteration 1	62	44	26	6754
iteration 2	623	474	325	6754
iteration 3	1693	1244	795	6754
phase II	0	0	0	5959
round 5				
iteration 1	26	26	26	5959
iteration 2	515	443	371	5959
iteration 3	1499	1223	947	5959
phase II	0	0	0	5012
round 6				
iteration 1	36	28	20	5012
iteration 2	617	550	483	5012
iteration 3	1594	1383	1172	5012
phase II	0	0	0	3840
round 7				
iteration 1	52	50	48	3840
iteration 2	675	619	563	3840
iteration 3	1578	1425	1272	3840
phase II	0	0	0	2568
round 8				
iteration 1	73	72	71	2568
iteration 2	650	604	558	2568
iteration 3	1317	1231	1145	2568
phase II	0	0	0	1423
round 9				
iteration 1	66	66	66	1423
iteration 2	509	498	487	1423
iteration 3	921	905	889	1423
iteration 4	1002	984	966	1423
iteration 5	1039	1022	1005	1423
phase II	0	0	0	418

	# of digits with $p^* < p_{thr}$	# of wrong digits with $p^* < p_{thr}$	decrease of wrong digits	# of wrong digits after phase II
round 10				
iteration 1	32	32	32	418
iteration 2	183	183	183	418
iteration 3	289	287	285	418
iteration 4	306	305	304	418
iteration 5	314	313	312	418
phase II	0	0	0	106
round 11				
iteration 1	4	4	4	106
iteration 2	62	62	62	106
iteration 3	96	96	96	106
iteration 4	106	106	106	106
phase II	0	0	0	0

Rounds 1 to 8 are terminated by $N_w \geq N_{thr}$, and rounds 9 to 10 by the criterion $i = \alpha$ ($\alpha = 5$). Observe that iteration 4 and 5 in rounds 9 and 10 have only small correction effect. This justifies the termination of a round after a certain number of iterations. It also shows that $\alpha = 3$ would have been a suitable choice as well. Finally round 11 is terminated since the corrected sequence after iteration 4 satisfies the basic feedback relation. Thus we have reconstructed the original LFSR-sequence after 35 iterations in total.

References:

[1] W. Meier and O. Staffelbach, "Fast correlation attack on stream ciphers", full paper, to appear.

[2] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only", IEEE Trans. Comput., vol. C-34, pp. 81-85, Jan. 1985