Overtaking VEST

Antoine Joux[1,2] and Jean-René Reinhard[3]

[1] DGA
[2] Université de Versailles St-Quentin-en-Yvelines, PRISM
45, avenue des États-Unis, 78035 Versailles Cedex, France
antoine.joux@m4x.org
[3] DCSSI Crypto Lab
51, Boulevard de La Tour-Maubourg
75700 Paris 07 SP, France
jean-rene.reinhard@m4x.org

Abstract. VEST is a set of four stream cipher families submitted by S. O'Neil, B. Gittins and H. Landman to the eSTREAM call for stream cipher proposals of the European project ECRYPT. The state of any family member is made of three components: a counter, a counter diffusor and a core accumulator. We show that collisions can be found in the counter during the IV Setup. Moreover they can be combined with a collision in the linear counter diffusor to form collisions on the whole cipher. As a consequence, it is possible to retrieve 53 bits of the keyed state of the stream cipher by performing a chosen IV attack. For the default member of a VEST family, we present a "long" IV attack which requires $2^{22.24}$ IV setups, and a "short" IV attack which requires $2^{28.73}$ IV setups on average. The 53 bits retrieved can be used to reduce the complexity of the exhaustive key search. The chosen IV attack can be turned into a chosen message attack on a MAC based on VEST.
Keywords: Stream cipher, inner collision, chosen IV attack

1 Introduction

VEST [8] is a set of four stream cipher families proposed to the eSTREAM project [6] by S. O'Neil, B. Gittins and H. Landman . VEST-v, with $v \in \{4, 8, 16, 32\}$, is a family of stream ciphers with expected security respectively 2^{80}, 2^{128}, 2^{160} and 2^{256}, and output rate v bits by clock cycle. All families share the same design. Only the sizes of the components change to meet the target security. There also is a selection algorithm which given a parameter called the family key outputs a specific member of a VEST family.

Recently, VEST specifications have been updated [9]. Compared to the earlier specification [8], changes include some modifications in the parameters used and also the definition of additional modes, that turn VEST in a MAC or an authenticated encryption scheme.

In this paper, we point out basic weaknesses of VEST components. The weaknesses can be used to create inner collisions in the algorithm, in a way similar to local collisions in hash function of the SHA family [4]. As a consequence we are able to mount a chosen IV attack against VEST stream cipher that recovers 53 bits of the keyed state. This information enables us to reduce by 53 bits the complexity of an exhaustive key search. The chosen IV attack on VEST stream cipher can also be used as an existential forgery attack against the VEST MAC.

In section 2, we give a description of VEST components and of their internal modes of operation. Then we describe in section 3 the basic weaknesses we found on the counter and linear counter diffusor components. In section 4, we describe two efficient chosen IVs attacks that recover 53 bits of the keyed state of the cipher. In section 5, we use these 53 bits to greatly speed-up exhaustive key search. Finally, in section 6, we describe briefly how to turn the attacks of section 4 into existential forgery attacks against VEST in MAC mode.

2 Description of VEST

Members of the VEST stream cipher families are made of four components:

- a set of 16 non linear feedback shift registers, called *counter*,
- a linear counter diffusor,
- an accumulator,
- a memory-less linear output filter.

There are three main internal modes of operation:

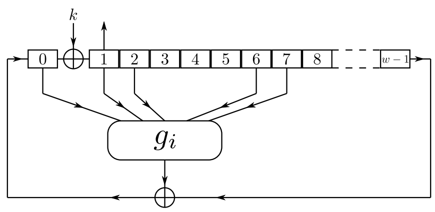
- The Key Setup mode which introduces the key into the cipher state. The state of the cipher is first set to all 0's, then the key is introduced and produces a keyed state. In this mode, the cipher does not output any data.
- The IV Setup mode which introduces an IV into a keyed state. This mode also has no output data.
- The Keystream generation mode which produces a stream of pseudo random bits.

We review in the following the four components of a VEST stream cipher and the Key Setup and IV Setup mechanisms.

2.1 Counter

The counter, a set of 16 registers c_i, $(0 \leq i < 16)$, is the autonomous part of the stream cipher in Keystream generation mode. Each register is a non linear feedback shift register (NLFSR) of size $w = 10$ or 11 bits. As in [8, 9], we note $c_{i\ j}^{r}$ the value of bit j of register i at step r. Their update function is described in Fig. 1.

Each register is updated independently. There are two register modes of operation. While in register counter mode, the registers are autonomously updated. In register keying mode, the update function of a register is disturbed by one bit at each clock cycle. The functions g_i are non-linear and chosen such that the graphs of the registers update functions are made of two cycles of approximately



– **Register Keying mode**
- $c_{i\ 0}^{r+1} = g_i(c_{i\ 0}^r, c_{i\ 1}^r, c_{i\ 2}^r, c_{i\ 6}^r, c_{i\ 7}^r) \oplus c_{i\ w_i-1}^r$
- $c_{i\ 1}^{r+1} = c_{i\ 0}^r \oplus k_i^r$
- $c_{i\ j}^{r+1} = c_{i\ j-1}^r$

– **Register Counter mode**
- $c_{i\ 0}^{r+1} = g_i(c_{i\ 0}^r, c_{i\ 1}^r, c_{i\ 2}^r, c_{i\ 6}^r, c_{i\ 7}^r) \oplus c_{i\ w_i-1}^r$
- $c_{i\ j}^{r+1} = c_{i\ j-1}^r$

Fig. 1. Register update

same length, the length of all cycles being pairwise relatively prime. Thus, a minimum period of evolution of the set of registers can be guaranteed. 32 non linear functions meeting this requirement are specified in [9, Appendix F], 16 for registers of length 11 and 16 for registers of length 10. For a given member of a family, every register is associated to a non linear function specified for its length. Every cycle, bit 1 is extracted from each of the 16 registers.

2.2 Linear counter diffusor

The linear counter diffusor is a 10–bit value used to disturb the core accumulator. Every cycle, the linear counter diffusor is updated linearly with the 16–bit output from the counter. As in [8, 9], we note d_j^r the value at step r of bit j of the linear counter diffusor. We note

$$D^{(r)} = \begin{pmatrix} d_0^r \\ d_1^r \\ \vdots \\ d_9^r \end{pmatrix} \quad C^{(r)} = \begin{pmatrix} c_{0\ 1}^r \\ c_{1\ 1}^r \\ \vdots \\ c_{15\ 1}^r \end{pmatrix}.$$

The linear counter diffusor update function can be written as

$$D^{(r+1)} = A \cdot D^{(r)} \oplus M \cdot C^{(r)} \oplus B,$$

with

$$A = \begin{bmatrix} 0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \quad M = \begin{bmatrix} 0&1&0&0&1&1&0&0&0&0&0&1&0&1&0&0 \\ 1&0&1&0&0&0&1&0&1&0&0&0&0&0&1&0 \\ 0&0&0&1&1&0&0&1&0&0&1&0&0&0&0&1 \\ 1&0&0&1&0&1&0&0&0&1&0&0&1&0&0&0 \\ 0&1&0&0&1&0&1&0&0&0&0&0&1&0&0&1 \\ 1&0&0&0&0&0&0&1&0&1&0&0&0&1&1&0 \\ 0&1&0&0&0&0&0&0&1&0&0&1&0&0&1&1 \\ 0&0&1&0&0&1&1&0&0&0&1&0&1&0&0&0 \\ 1&0&0&1&0&0&0&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&0&1 \end{bmatrix} \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

2.3 Accumulator

The remainder of the state of a VEST stream-cipher is an accumulator. Every cycle the accumulator state goes through a substitution phase and a permutation phase. Then the output of the counter diffusor is XOR-ed to the first 10 bits of the accumulator state. For a full description of the accumulator, we refer to [9].

2.4 Output Combiner

At each clock cycle, depending on its current state the cipher outputs either 0 or n bits. Each output bit is computed by taking the exclusive-or of 6 bits taken from the accumulator state. The number of output bits is at least 18 times smaller than the size of the accumulator. The design of the core of the algorithm, that is to say the accumulator and this output filter follows the same design strategy as the LEX stream cipher, another candidate to the eSTREAM project [1, 2]. The idea is to extract a small part of a state which is updated by one round of a block cipher, the round-key being provided by an autonomous component.

2.5 Key Setup mode

The Key Setup takes as input a F–bit key K, where F is a multiple of 8, and enters it in the cipher state. At the beginning of the Key Setup, all bits of the state are set to 1. At the end of the Key Setup, the value of the state of the cipher is called the keyed state. During Key Setup, no output is produced by the cipher. The linear counter diffusor and the accumulator work as described above. The registers of the counter go through two phases:

- The first phase of the keysetup introduces K in the registers. First the key is prepended with fifteen 0's, and appended with a single 1 followed by fifteen 0's, creating a key K'. During $F + 16$ steps the registers operate in register keying mode. At each step r, the bits $K'_r, \ldots, K'_{r+15}$ are used to disturb the evolution of registers $0, \ldots, 15$ respectively.
- The second phase of the Key Setup introduces a 8–bit constant in the first 8 registers, and mixes the state of the cipher. During the introduction of the constant, the first 8 registers are in keying mode, register i being disturbed by bit i of the constant, and the last 8 registers are in counter mode. Then the state of the cipher is mixed by going through 31 steps during which all the registers are in counter mode.

The keyed state can be stored and reloaded later into the state of the cipher, if one wants for example to speed up IV setups using the same key. The keyed state is equivalent to the key introduced.

2.6 IV Setup mode

The IV Setup may be applied after the Key Setup. It takes as input an IV of length W bits, with W a multiple of 8. As for the Key Setup, no output is produced during IV Setup and the accumulator and counter diffusor work as described above. The counter goes through 2 phases:

- During the first phase of the IV Setup, the IV is introduced into the counter. This phase lasts $W/8$ steps. Note that the IV is not introduced in all registers as the key. Instead it only affects 8 registers as the constant introduced in the second phase of the keying process. As a consequence each bit of IV affects a single register. At each clock cycle, one byte of the IV is introduced.
- The second phase is identical to the Key Setup second phase with a different value of the constant.

3 Basic weaknesses of VEST components

In this section, we identify two weaknesses of VEST stream ciphers. They concern the differential behaviour of the NLFSRs in keying mode and of the counter diffusor.

3.1 Differential characteristics of the registers

During the first phase of IV Setup, the update function of register i, $0 \leq i \leq 7$ can be viewed as a function

$$f_i : \{0,1\}^{w_i} \times \{0,1\}^n \rightarrow \{0,1\}^{w_i} \times \{0,1\}^n,$$

which modifies a state with an input IV of length n. The output of the function is the modified state and an output value of length n. Studying the differential behaviour of this function with respect to its second input, we find some kind of imbalance.

The differential patterns relevant within the context of the IV Setup are

$$0 \times \Delta \rightarrow 0 \times \alpha :$$

starting from a common value (for example the keyed register state) we introduce a pair of IVs with difference Δ and look for a collision on the register state after the first phase of the IV Setup and a fixed difference α on the output. For an IV pair, we call colliding states the states starting from which we get a collision and the expected difference α on the register output after processing each IV of the pair.

Differential behaviour over one step. In order to understand the source of the imbalance, let us consider the effect of a single bit difference in the register state on one step of the IV Setup. We can distinguish three cases:

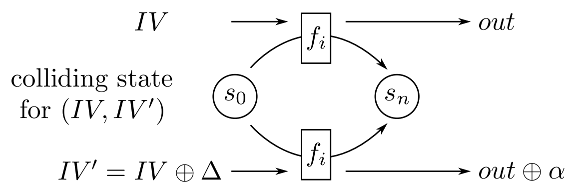


Fig. 2. Differential pattern and colliding states

- The difference is at position $w-1$: the update moves the difference to position 0,
- The difference is at position $j \notin \{0, 1, 2, 6, 7\}$: the update shifts the difference to position $j + 1$
- The difference is at position $j \in \{0, 1, 2, 6, 7\}$: the update shifts the difference to position $j + 1$ and, depending on the non linear function, may also create a difference at position 0.

We still have to take into account the IV bit introduction at this step. After the computation of the feedback and the shift of the register, but before the output at this step, one bit of IV is XOR-ed at position 1. In our differential setting this bit can be used either to introduce a new difference in position 1 or to correct a difference that was present at position 0 at the end of the previous step.

At the end of each step, the output is the bit in position 1 of the register state.

Local Collisions. We now adopt a strategy very similar to the local collisions used in [4] to attack SHA-0. The key idea is to introduce a single difference in the register state and control its propagation until it vanishes. We first consider the linear part of the update function. We then take into account the non-linear part.

At step 0 we introduce a difference through different IV bits. Thus, after step 0 there is a difference in position 1 of the register. The linear part of the update function consists solely of a rotation of the register bits. After step 1 the difference is in position 2, then after step 2 in position 3 ... and after step $w - 1$ the difference is in position 0. At step w it comes back in position 1, where we can cancel it by using a secondary difference on the IV pair. Thus when the non-linear feedback function is not used, it takes $w + 1$ steps to introduce and then cancel a difference. Looking at the output difference α, taken from bit 1, we see that it consists in a single 1 followed by w 0's. In the sequel, we show how to build collisions following the same pattern, $w + 1$ steps and same value α.

We now add the non linear function (NLF). The collision described above does no longer occur all the time. Indeed, when the NLF is active, i.e. when there is a difference on at least one of its input, there may be a difference on its output. Heuristically, this happens with probability 1/2. Thus after a step during which the NLF is active, there may be an additional difference in position 0. To prevent the propagation of this difference, we use the IV introduction in position 1 at the next step to cancel it. Combining all the IV bit differences on the $w+1$ steps we get a corrective pattern. Note that with IV differences on $w+1$ steps, it is not possible to correct additional differences produced during step w.

Due to the propagation of the initial difference through every position of the register, the number of active steps is at least 5 : steps 1, 2, 6, 7 and w. Heuristically, after each active step there is probability 1/2 that another difference appear in position 0, thus that the next step is also active. We can expect that some corrective patterns lead to collisions with probability 2^{-5}. Thus, there should be 2^{w-5} colliding states for some corrective patterns.

In practice, $w = 10$ or $w = 11$. So we are able to compute all the differential characteristic of length $w+1$ of a counter. In fact, for a given IV and starting state, there is at most one IV difference that creates a collision after $w+1$ steps and has the correct output difference α. We computed all these values and stored for each IV pair, the initial states for which a collision occurs. We performed this exhaustive search for each of the possible non linear functions. This computation takes a few minutes an Intel Celeron 1.4 Ghz.

We observe that for some good pairs of IVs with appropriate difference patterns, the number of colliding states is higher than 2^{w-5}, sometimes exceeding 2^{w-5} by a factor of 2 or more. We give in Table 1 the maximum number of colliding states N_i for the best IV pair for each proposed non linear function in [9, Appendix F]. For the first (resp. last) 16 functions $w = 11$ (resp. 10) and the expected number of colliding states is 64 (resp. 32).

Table 1. Size of the largest colliding states for register update functions

i	N_i	i	N_i	i	N_i	i	N_i	i	N_i	i	N_i	i	N_i	i	N_i
0	127	4	106	8	122	12	102	16	70	20	44	24	59	28	52
1	107	5	107	9	95	13	96	17	67	21	60	25	76	29	64
2	117	6	96	10	90	14	104	18	74	22	62	26	65	30	54
3	128	7	150	11	156	15	136	19	52	23	77	27	54	31	77

3.2 Collision in the Counter Diffusor

In Section 2.2, a description of the linear counter diffusor was given. Matrix M having more columns than rows has a non zero kernel, generated by vectors:

$$
\begin{array}{l}
(\mathbf{1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0})^{\mathbf{T}}, \\
(1,1,1,1,0,1,1,0,1,1,1,0,0,0,0,0)^T, \\
(0,1,1,0,0,0,1,0,1,0,0,1,0,0,0,0)^T, \\
(0,1,0,1,1,0,1,0,1,0,0,0,1,0,0,0)^T, \\
(1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0)^T, \\
(0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,1)^T
\end{array}
$$

This clearly contradicts the property of collision freeness in elementary VEST components claimed in [9, Section 5.4]. If the output of the counter at step r differs by a linear combination of these vectors, the contribution of the counters to the update of the diffusion bits $\{d_j^r\}_j$ will be the same. The highlighted kernel vector only uses the first 8 registers which can, to some extent, be controlled and is especially useful in our attack.

4 Partial Keyed State Recovery

In this section, we exploit the vulnerabilities described in the previous section to recover the value of a fixed part of the keyed state. All the attacks described are chosen IVs differential attacks. The first attack assumes no constraints on the IV length. The second attack uses an IV as short as possible and can be used when the length of the IV is constrained to some standard value, namely 128 bits. In fact, we show that an attack is possible as soon as the IV length is greater than 96 bits.

4.1 Attack with long IVs

We describe in this subsection a first attack which assumes that the IV length is greater than $23 \times 8 = 184$ bits. We note that [9] doesn't specify any limit on the IV length.

The goal of the attack is to exploit the above weaknesses to build a pair of colliding IVs, that is to say a pair of IVs such that the state of the cipher is the same after the Key Setup using a unknown key K and the IV Setup using each IV of the pair. We have studied in the previous section the differential behaviour of the NLFSRs in keying mode and of the counter diffusor. Our attack uses the outlined properties to introduce differences into the counter and then cancel them, while introducing no differences in the core accumulator.

Indeed, during the first phase of the IV Setup, all the NLFSRs are not updated in the same way. NLFSRs 0 to 7 work in keying mode and are disturbed by IV bits, each IV bit being used exactly once, while NLFSRs 8 to 15 work in counter mode. As a consequence, an attacker can only influence the first 8 NLFSRs. In order to introduce no difference in the core accumulator, the counter diffusor state must remain the same while the counter state varies. The counter diffusor update depending linearly on the NLFSRs output through matrix M, this is possible if and only if the counter output difference lies in the kernel of M. Furthermore, as the attacker can only introduce differences in the first 8 NLFSRs, the counter output difference can only be non zero on the corresponding components. Unfortunately for VEST security, one such element exists in the kernel of M (see section 3.2). Thus, if the outputs of the NLFSRs are different for exactly NLFSRs 0, 4, 5, 6 and 7, the counter diffusor is updated in the same way as if there were no difference. As a consequence, if there is no difference in the counter diffusor and in the core accumulator and if the counter output difference is as above, no difference is introduced in the counter diffusor and in the core accumulator.

Furthermore, we described in section 3.1 differential patterns on the NLFSRs. The output difference for these differential patterns is a single 1 followed by w 0's. If we combine such differential patterns, one pattern for each of the five NLFSRs 0, 4, 5, 6 and 7, making them start at the same step, and introduce no difference on the three NLFSRs 1, 2 and 3, and no difference on NLFSRs 0, 4, 5, 6 and 7 after the differential patterns, we get a differential pattern on the whole counter such that:

- there is a collision on the whole counter state if the starting state of the counter is in a good set;
- the output of the NLFSRs doesn't introduce differences in the counter diffusor and in the core accumulator. Indeed the counter output difference is either all 0's, or equal to the highlighted element of the kernel of M given in 3.2.

In order to make the good set explicit, we first remind that during the IV Setup first phase different NLFSRs are updated independently. Furthermore, we can extract from the IV the part that affects a particular register. Having a global collision on the whole counter state is thus equivalent to five partial collisions on the registers 0, 4, 5, 6 and 7. Thus the set of initial states of the counter that leads to collisions is the cartesian product $\mathcal{S}$ of the sets $\mathcal{S}^r$ of registers initial states that lead to collisions for the IV pairs used. Note that due to the guaranteed output difference of the partial collisions, this leads to a collision on the whole cipher state at the end of the first phase of the IV Setup. As no difference is introduced in the second phase of the IV Setup, there is a collision on the whole cipher state at the end of the IV Setup.

The idea of the basic attack is to choose for each register an IV pair that maximizes the number of colliding states. This also maximizes the number of initial states leading to a collision on the whole counter. If we choose an initial counter state randomly, the probability it yields a collision is the number of colliding states divided by the total number of states. For the default member of a VEST family the value of this probability is:

$$p \approx 2^{-21.24}.$$

We then build IV pairs of length 23 bytes, such that the first 11 bytes are random and identical, and the last 12 bytes is a fixed IV pair, the composition of the best IV pairs for registers 0, 4, 5, 6 and 7 and some fixed values for registers 1, 2 and 3. We note that on average 11 bits are enough to completely randomize the counter state. As a consequence p is the probability that such an IV pair collides on the IV Setup, resulting in a collision on the whole cipher state after the IV Setup as explained above. To test this collision, we compare the first 32 bits of the keystream output after each IV setup. If a collision occurs in the IV Setup the keystream bits are identical. If there is no collision, the probability of having for two different IVs the same first 32 bits of keystream is 2^{-32}. After approximately $1/p$ pairs of IV setups we find with high probability a pair of IVs yielding the same first keystream bits and with high probability it is because of an inner collision on the counter.

Once such a pair is obtained, it is easy to retrieve the 53 bits of the keyed state in registers 0, 4, 5, 6 and 7 ($w = 11$ for registers 0, 4 and 7, $w = 10$ for registers 5 and 6). We proceed register by register. Having a collision after the IV Setup, the state of register i after step 11 is a colliding state for the IV pair extracted from the colliding IVs by taking the IV part relative to register r, restricted to step 12 to 23. We then make a guess on this value, among the set $\mathcal{S}^r$ of colliding states. Backtracking the 11 first steps of the IV Setup, we obtain a candidate for the register value after Key Setup. In order to test this value, starting from the colliding IV pair, we build another IV pair which only differs on its contribution to register r. The first 11 bits are chosen to ensure that the new starting value for register r after the randomizing step belongs to $\mathcal{S}^r$ if the guess is correct. The remaining 12 bits are left unchanged in both element of the IV pair. If the IV setup using this new pair does not give rise to a collision, we eliminate the guessed values from the candidates. The probability for a wrong candidate to pass one test successfully is $\#\mathcal{S}^r/2^{w_r}$. Iterating this procedure we get rapidly rid of the wrong candidates. Repeating this procedure on all interesting registers for some examples we were able to retrieve the value of the registers in the keyed state using a little less than 500 additional IV setups. It is possible to improve this basic approach and eliminate several values at once using a single IV setup. This lowers the number of additional IV setups below 200.

Of course, this attack can easily be adapted to contexts were the IV is longer than 23 bytes. It can also be adapted when using shorter IVs, as long as the IV length is greater than 12. In that case, this version of the attack won't work anymore for all the keys, since the randomizing of the register states isn't guaranteed to turn the keyed register states into an element of $\mathcal{S}$. Thus the attack becomes key-dependent.

The limiting part of the attack is the look-up of a collision on the whole counter. It's complexity is approximately $2/p \approx 2^{22.24}$ IV setups. We implemented this attack. On a Pentium Xeon 2.8 GHz it takes a few minutes on the average to find a pair of colliding IVs.

4.2 Attack with short IVs

In the previous subsection, we described a differential attack that allows to recover part of the keyed state provided the IV length is long enough. We also mentioned that the attack becomes key-dependent when the IV length is reduced, and that the minimal IV length for which an attack may be possible is 12 bytes. In this subsection we describe an improved approach that works for all keys using IVs of minimal length, i.e. 12 bytes.

The former attack generates random states from the keyed state until one random state falls in a particular set $\mathcal{S}$. The idea of the attack described below is to generate a family of sets $\{\mathcal{S}_i\}_i$ which covers the entire state space.

Indeed, for a particular register r, we can associate to a pair of IVs C_i^r for this register the set of the states starting from which we get the expected register output difference (1 followed by 0's) and a collision on the register state after IV setups using each of the IVs of C_i^r. We saw in section 3.1 that for certain IV pairs C^r the cardinality of the associated set $\mathcal{S}^r, \#\mathcal{S}^r$, might be particularly high. The register size w being only 10 or 11 bits, we saw it was possible to compute for each IV pairs with IV length $w+1$ the set of colliding states for this pair. By doing this we get a family of sets. We have verified that the union of these sets covers the whole space of register values for any of the proposed non linear function. Thus, it is possible to select $N^{(r)} < 2^{w_r}$ sets, $\mathcal{S}_1^r, \mathcal{S}_2^r, \ldots, \mathcal{S}_{N^{(r)}}^r$, so that the union of these sets covers $\{0,1\}^{w_r}$. By composing 5 families of sets picked for registers 0, 4, 5, 6 and 7 by cartesian product, we get a family of sets whose union covers all the possible values for the 53–bit part of the keyed state made of these registers. The size of this family $\{\mathcal{S}_i\}_i$ is the product of the size of the families for each registers: $N = N^{(0)}N^{(4)}N^{(5)}N^{(6)}N^{(7)}$. The elements of the family are sets of states for which a collision occurs after the IV setup using an IV pair obtained as a combination of the IV pairs for each register.

Suppose that we have such a family $\{\mathcal{S}_i\}_i$ and the associated family of IV pairs. By doing two IV setups for each pair, we are bound to find a pair of IV for which there is a collision after the IV Setup. The complexity of this attack in number of IV setups is $2N$ in the worst case. Furthermore, if we first test the pairs whose associated set of colliding states is largest, the expected number of IV setups on the average case becomes smaller.

In order to get the best complexity we have to build for each register a family of covering IV pairs $\{C_i^r\}_i$. In order to improve the average complexity, we use a greedy algorithm to pick up these families. We first build all the colliding sets for all the IV pairs. We then sort them by decreasing cardinality and pick up the first pair. We then remove the states in the colliding set of the picked IV pair from all the unpicked sets and sort again the list of sets by decreasing size. We iterate this step until we get a list of pairs so that every state is in the colliding state of a pair in the list. We give in Table 2 the length of the obtained lists for the functions used by registers 0, 4, 5, 6 and 7 in the default member of a VEST family.

Once we have obtained the families for the five interesting registers, we have to combine them to get a family of IV pairs for the whole counter. In order

Table 2. Size of the covering families for some non linear functions

function number	covering family size
0	59
1	93
19	77
20	86
2	96

to get the best average complexity for the attack using these 5 families, we have to test the pairs by decreasing order of their colliding sets of states. For VEST-v default member the number of pairs is $\approx 2^{31.70}$. The generic sorting algorithms are difficult to apply because of the amount of memory required. However, we do not really need to store the IV pairs order, as long as we are able to enumerate them rapidly. Furthermore, the only information required to make the comparison is the size of the colliding states sets for each IV pair. Thus we are left with the following problem:

Given two lists of integers $(a_i)_{1\leq i\leq N_a}, (b_j)_{1\leq j\leq N_b}$ sorted by decreasing order, enumerate in decreasing order all the products $a_i b_j$.

In [10] and [3], an algorithm is proposed to solve this problem. This algorithm does not store all the products. Assuming $N_a > N_b$, its time complexity is $O(N_a N_b \log N_b)$ and its memory complexity $O(N_b)$.

Implementing this algorithm and using the 5 families obtained we were able to enumerate the $2^{31.70}$ IV pairs of covering family in decreasing order in less than 2 hours on a 1.4 GHz Celeron M processor. Noting $(C_i)_{1\leq i\leq N}$ this list of pairs, $(\mathcal{S}_i)_{1\leq i\leq N}$ the associated list of colliding states and $(N_i = \#\mathcal{S}_i)_{1\leq i\leq N}$ the decreasing list of the cardinality of these sets, the mean complexity of the collision search, evaluated in number of pairs of IV setups is:

$$\mathcal{C} = \sum_{i=1}^{N} i \cdot \frac{N_i}{2^{53}}.$$

Computing this sum during the enumeration of the pairs in decreasing order, we obtain for the VEST-v default member $\mathcal{C} \approx 2^{27.73}$.

Once a collision is obtained, we have a small list of candidate values for the keyed state value of every interesting registers. By testing these candidates separately for each register, we are able to retrieve 53 bits of the keyed state. The number of additional IV setups for these tests is negligible before the number of IV setups necessary to find an IV collision.

The attack described in this subsection enables an attacker to retrieve 53 bits of the keyed state in $2^{28.73}$ IV setups on average and $2^{32.70}$ in the worst case. This is slightly more than the complexity of the basic attack. However this attack uses IVs of minimal length. Indeed, at least 12 steps are required to create a collision in registers 0, 4 and 7. With longer IVs, of length between 12 and 23 bytes, we can use the beginning of the IVs as a partial randomizer and add an early abort strategy to the attack of this subsection, to improve the overall complexity of the attack.

5 Key Recovery

In this section, we show how the partial keyed state value recovered by the attacks of the previous section can be used to recover the key of the stream cipher faster than exhaustive search. This enables to evaluate the impact on the cipher security of the collisions that were discovered in the IV Setup mechanism.

5.1 Backtracking the Key Setup second phase

We begin by backtracking the second phase of the key setup. As all the bits entering the interesting registers (0, 4, 5, 6 and 7) are known, we are able to retrieve the states of these registers after the key bits introduction. We also know that the registers are set to 1 before the key bits introduction. Thus we are able to perform a meet-in-the-middle attack on the key, even though the key bits are introduced in all the registers through a sliding window mechanism. The same kind of attack disqualified double-DES as a successor of DES.

5.2 Meet-in-the-middle attack

One can notice that bits 0 to $l-1$ (resp. l to $F-1$) of the key are introduced in register r between step $15-r$ and step $l-1+15-r$ (resp. $l+15-r$ and $F-1+15-r$). By guessing the first l bits or the last $F-l$ bits of the key, one can compute the values of register r before step $l+15-r$. This enables us to implement a time/memory tradeoff attack against VEST ciphers. We build the table $\mathcal{A}$ of the 53–bit values of the 5 interesting registers, at steps $l+15-r$. This requires 2^l memory.

Then, for each 2^{F-l} values j of the end of the key we perform the following:

- backtrack the register values assuming j: we note x the 53–bit value obtained;
- look for i so that $\mathcal{A}[i]=x$. The probability of the match is 2^{-53};
- for each match, check the key $(i||j)$, where $||$ designates the concatenation.

Complexity. This enables to explore all the keys which sets the states of the interesting registers to the recovered values. This attack recovers the key used by the cipher using $2^{\max(F-53,F-l)}$ time and 2^l memory. The average number of keys to test is 2^{F-53}.

5.3 Key recovery through related-key attack

It is also possible to mount a very efficient related key attack. Assume we are given two ciphers keyed with key K and key K' which differs of K only on bit $F-l-1$. Performing our chosen IV attack, we are able to retrieve for both keys the keyed states of the interesting registers. By guessing the last l bits of the key, we can backtrack their introduction from the keyed state up to the step after the key difference introduction for each register. For the correct guess, there should only be a difference at position 1 of the registers after the backtracking. This happens with probability 2^{-53}. Thus we are able to check our guesses if $l < 53$. For the registers to behave in a random way l should also be larger than their length.

Once we have determined the last l bits of the key, we can iterate this process on the unknown part of the key. For a 128–bit key, taking $l = 16$ we are able to recover the whole key with 8 related keys performing $\approx 2^{26}$ IV setups and $\approx 2^{19}$ partial key introduction backtracking.

5.4 Security Discussion

The attacks described above show that the differential attack result can theoretically be exploited to recover the cipher key faster than exhaustive key search. It seems that this is also the case in more practical attacker models. This breaks VEST cipher when it is used with keys of size of the security parameters. In [9, Section 3.3], the authors of VEST recommend to use keys of size at least twice the security parameter. In this case VEST could be considered as resistant to this attack since the complexity of our time memory tradeoff may remain greater than the security parameter. It will nevertheless fall to the same related-key attack. Anyway, it is usually considered as a bad practice to use cryptosystems where part of the key material can easily be recovered.

The current specification of VEST stream cipher does not forbid the common usage consisting of taking a key the length of the security parameter. In the case of use of a key the length of the security parameter, it even recommends the use of long IVs, which makes our attack more efficient. In its latest version, VEST does not meet its claimed security.

6 Existential forgeries for VEST hash MAC mode

VEST can be used as a keyed hash function using the procedure described in [9, Section 3.4]. The VEST cipher is first keyed. Then the data to be MAC-ed is introduced into the cipher as an IV during the IV Setup, using a different constant for the second phase. Finally $2n$ bits are output by the cipher in counter mode. In order to finish the description of this keyed hash function, a padding should be described to hash messages of arbitrary length. Independently of this padding we can transpose the chosen IV attack into an existential forgery attack.

Indeed, IVs in the previous attacks can be replaced by messages in the current setting. Thus, asking an oracle for approximately $2^{22.24}$ chosen message MACs enables an attacker to retrieve 53 bits of the keyed state of a VEST-v default member. The attacker can then create a pair of messages that collide and ask for the MAC of one of the messages. This MAC is the MAC value for the other message of the pair. This provides an easy existential forgery chosen message attack.

7 Conclusion

In this paper, we showed that despite its apparent complexity, the VEST stream cipher has simple properties which allows for the easy creation of internal collision. The overall result gives an efficient partial key recovery attack against VEST.

Our chosen IV attacks are practical. We were able to generate collisions for both attacks. The simple attack requires about 30 minutes on an Intel Xeon 2.8 GHz, the short IV attack required a few hours on the same machine.

Once again this shows that IV Setup is a very crucial part of a stream cipher security [11, 7, 5, 12].

Following this cryptanalysis, the authors of VEST proposed a modified version of their algorithm. The update function of the linear counter diffusor has been changed so that there is no inner collision in the diffusor during the IV setup. This makes the modified version immune to the differential attack presented in this paper. However, its security remains to be analyzed in the light of the weaknesses presented here.

Acknowledgements. We would like to thank Dan Bernstein for his parallelization of the key recovery attack.

References

1. A. Biryukov. A new 128 bit key stream cipher : LEX. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/013, 2005. `http://www.ecrypt.eu.org/stream`.
2. A. Biryukov. The Design of a Stream Cipher LEX. In E. Biham and A. Youssef, editors, *Selected Areas in Cryptography – SAC 2006*, Lecture Notes in Computer Science. Springer, 2006. to appear.
3. D. Boneh, A. Joux, and P. Nguyen. Why Textbook ElGamal and RSA Encryption are Insecure. In T. Okamoto, editor, *Advances in Cryptology — Proceedings of ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2000.
4. F. Chabaud and A. Joux. Differential Collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology — Proceedings of CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998.
5. C. Cid, H. Gilbert, and T. Johansson. Cryptanalysis of Pomaranch. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/060, 2005. `http://www.ecrypt.eu.org/stream`.
6. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. `http://www.ecrypt.eu.org/stream`.
7. E. Jaulmes and F. Muller. Cryptanalysis of ECRYPT Candidates F-FCSR-8 and F-FCSR-H. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/046, 2005. `http://www.ecrypt.eu.org/stream`.
8. S. O'Neil, B. Gittins, and H. Landman. VEST – Hardware-Dedicated Stream Ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/032, 2005. `http://www.ecrypt.eu.org/stream`.

9. S. O'Neil, B. Gittins, and H. Landman. VEST Ciphers. eSTREAM, ECRYPT Stream Cipher Project, 2006. `http://www.ecrypt.eu.org/stream/p2ciphers/vest/vest_p2.pdf`.
10. R. Schroeppel and A. Shamir. A $T = O(2^{n/2}), S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM Journal on Computing*, 10(3):456–464, 1981.
11. H. Wu and B. Preneel. Chosen IV Attack on Stream Cipher WG. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/045, 2005. `http://www.ecrypt.eu.org/stream`.
12. H. Wu and B. Preneel. Key Recovery Attack on Py and Pypy with Chosen IVs. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/052, 2006. `http://www.ecrypt.eu.org/stream`.