# Overtaking VEST

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

[1]DGA

[2]Université de Versailles-St-Quentin-en-Yvelines, PRISM

[3]DCSSI Crypto Lab

26 march 2007

# VEST

- VEST is a set of stream cipher families submitted to eSTREAM by S. O'Neil, B. Gittins and H. Landman
- HW Profile, Phase 2 candidate

| family | output by clock | security level |
|---|---|---|
| VEST–4 | 4 bits | $2^{80}$ |
| VEST–8 | 8 bits | $2^{128}$ |
| VEST–16 | 16 bits | $2^{160}$ |
| VEST–32 | 32 bits | $2^{256}$ |

- We present a chosen-IV attack against all families
- Based on inner collisions and biased differential behaviour of the IV setup
- Recovers 53 bits of the keyed state in $2^{22.74}$ IV setups

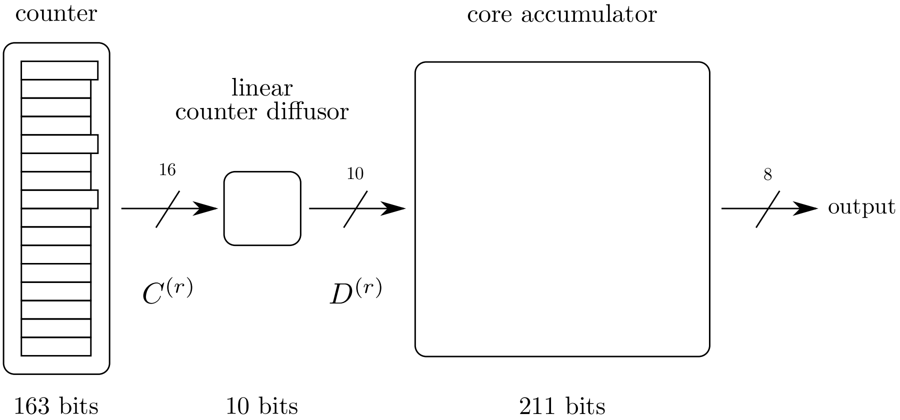
# VEST

- VEST is a set of stream cipher families submitted to eSTREAM by S. O'Neil, B. Gittins and H. Landman
- HW Profile, Phase 2 candidate

| family | output by clock | security level |
|---|---|---|
| VEST–4 | 4 bits | $2^{80}$ |
| VEST–8 | 8 bits | $2^{128}$ |
| VEST–16 | 16 bits | $2^{160}$ |
| VEST–32 | 32 bits | $2^{256}$ |

- We present a chosen-IV attack against all families
- Based on inner collisions and biased differential behaviour of the IV setup
- Recovers 53 bits of the keyed state in $2^{22.74}$ IV setups

# General description of VEST



# Description of VEST : Key and IV setups

**Key setup**

- NLFSRs are disturbed by the key bits
- every key bit enters once every NLFSRs
- Result: a keyed state

**IV setup**

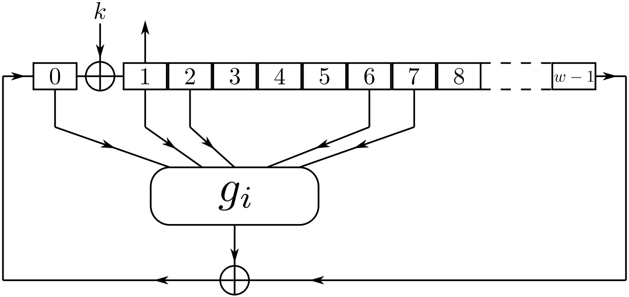
- NLFSRs 0 to 7 are disturbed by IV bits
- At each clock one byte of IV is used
- bit $i$ disturbs register $i$

Normal clock of the rest of the cipher

No ouput

# Description of VEST : NLFSRs

- Building block of the counter
- Length $w = 10$ or 11
- Non linear feedback functions $g_i$ chosen so that:
  - the registers have two cycles
  - all the cycles length are coprime



# Analysis of the counter diffusor

- Linear counter diffusor update function :

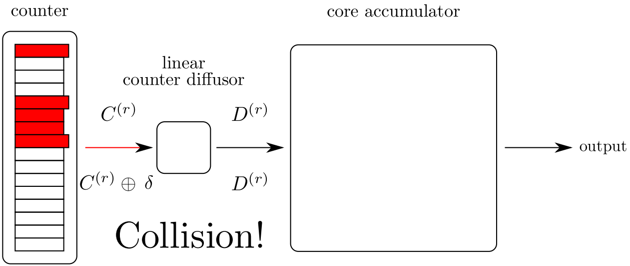
$$D^{(r+1)} = A \cdot D^{(r)} \oplus M \cdot C^{(r)} \oplus B$$

- $M$ is a $10 \times 16$ matrix
- $\ker(M)$ is non trivial

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

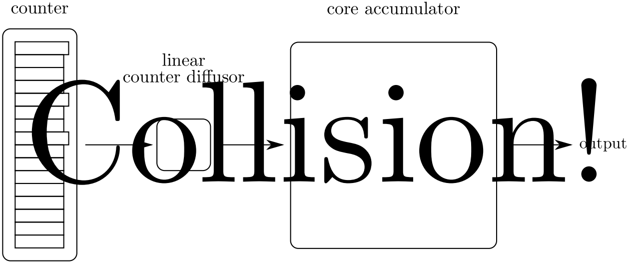
# How to use this property

- Introduce differences in the counter so that :
  - The differences in the counter cancel themselves after several steps
  - All the counter output differences are in $\ker(M)$
- We can do this during the IV setup because
  - We can control what happens in the first 8 NLFSRs
  - $(1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0)^T \in \ker(M)$



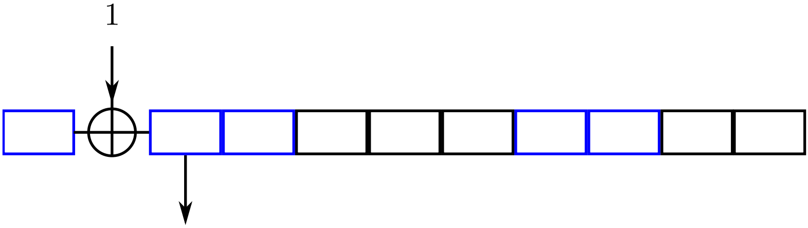
# How to use this property

- Introduce differences in the counter so that :
  - The differences in the counter cancel themselves after several steps
  - All the counter output differences are in $\ker(M)$
- We can do this during the IV setup because
  - We can control what happens in the first 8 NLFSRs
  - $(1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T \in \ker(M)$



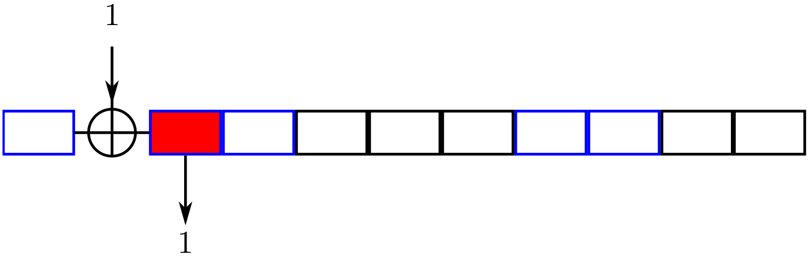
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



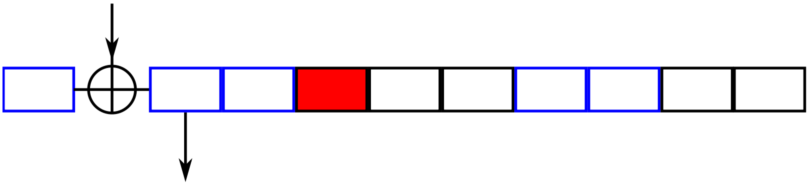
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



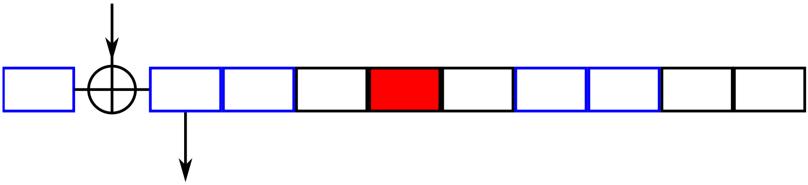
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



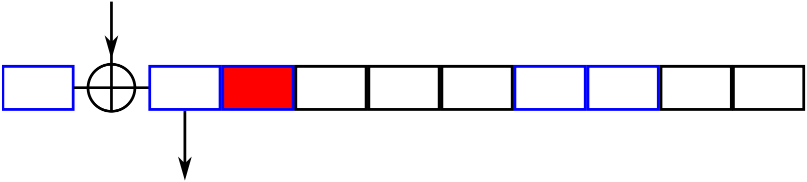
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



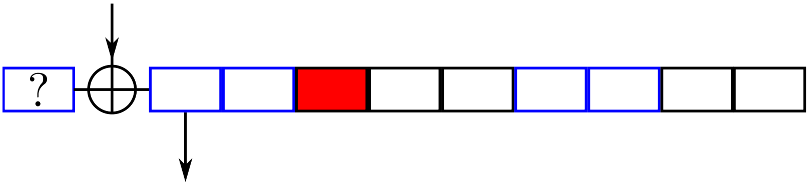
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



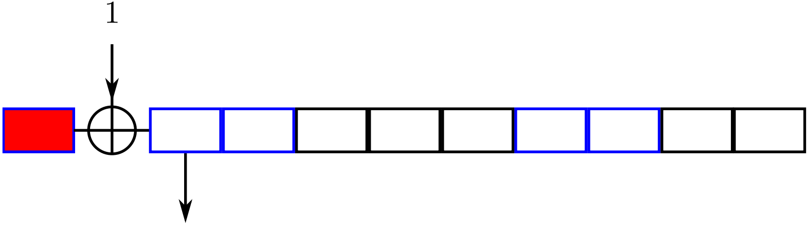
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



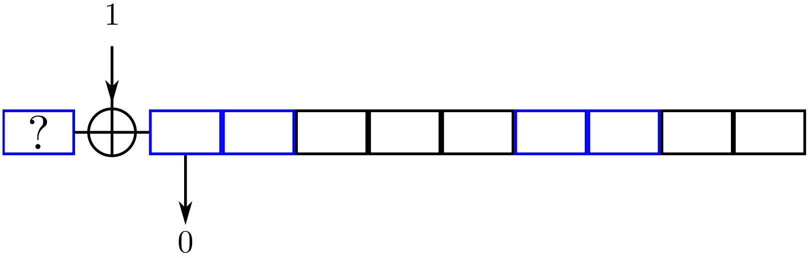
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



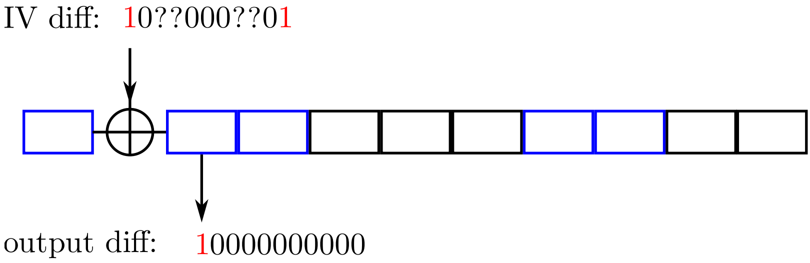
# Difference propagation in the NLFSRs

- Easy to introduce a difference during the IV Setup
- One bit difference propagation
- Ability to control an expected difference propagation



# Local collision pattern in the NLFSRs

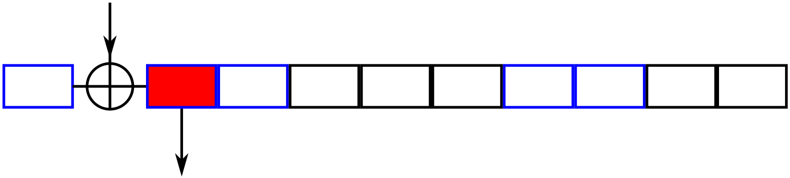
- Idea : Introduce a difference
- Control its propagation with IV bits so that only the first difference goes through bits 1 to $w$-1
- Similar to the local collision patterns in SHA



# Local collision pattern in the NLFSRs

- Idea : Introduce a difference
- Control its propagation with IV bits so that only the first difference goes through bits 1 to $w$-1
- Similar to the local collision patterns in SHA

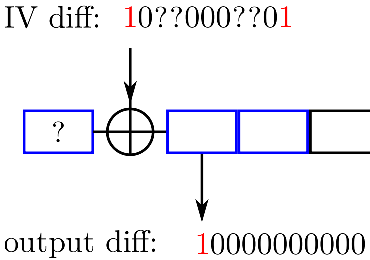
IV diff: 10??000??01



output diff: 10000000000

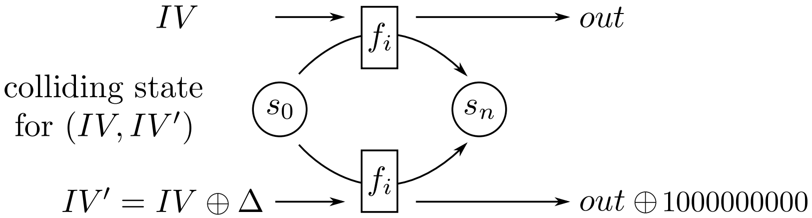
# Local collision pattern in the NLFSRs

- Idea : Introduce a difference
- Control its propagation with IV bits so that only the first difference goes through bits 1 to $w$-1
- Similar to the local collision patterns in SHA



# Colliding states

- In practice, we cannot control the difference (we cannot observe it)
- But, some differences should have good collision probability
- Key idea:
  - Fix $\Delta$ (and also best IV)
  - Randomize starting state



# Best IV pairs

- Non linearity: the IVs of the pair are important
- Small registers: we can test all IV pairs, and determine those for which there is good collision probability
- Size of the maximal colliding sets for the specified non linear function:

11–bit register functions:
expected size = 64

| $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ |
|---|---|---|---|---|---|---|---|
| 0 | 127 | 4 | 106 | 8 | 122 | 12 | 102 |
| 1 | 107 | 5 | 107 | 9 | 95 | 13 | 96 |
| 2 | 117 | 6 | 96 | 10 | 90 | 14 | 104 |
| 3 | 128 | 7 | 150 | 11 | 156 | 15 | 136 |

10–bit register functions:
expected size = 32

| $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ |
|---|---|---|---|---|---|---|---|
| 16 | 70 | 20 | 44 | 24 | 59 | 28 | 52 |
| 17 | 67 | 21 | 60 | 25 | 76 | 29 | 64 |
| 18 | 74 | 22 | 62 | 26 | 65 | 30 | 54 |
| 19 | 52 | 23 | 77 | 27 | 54 | 31 | 77 |

# Best IV pairs

- Non linearity: the IVs of the pair are important
- Small registers: we can test all IV pairs, and determine those for which there is good collision probability
- Size of the maximal colliding sets for the specified non linear function:

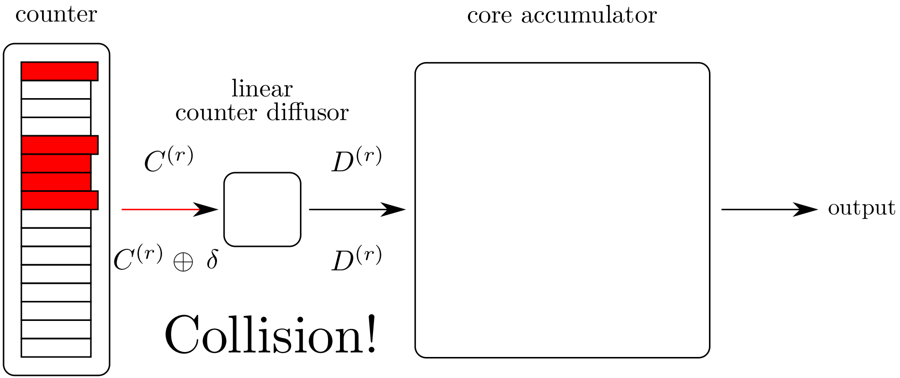
11–bit register functions:
expected size = 64

| $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ |
|---|---|---|---|---|---|---|---|
| 0 | 127 | 4 | 106 | 8 | 122 | 12 | 102 |
| 1 | 107 | 5 | 107 | 9 | 95 | 13 | 96 |
| 2 | 117 | 6 | 96 | 10 | 90 | 14 | 104 |
| 3 | 128 | 7 | 150 | 11 | 156 | 15 | 136 |

10–bit register functions:
expected size = 32

| $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ | $i$ | $N_i$ |
|---|---|---|---|---|---|---|---|
| 16 | 70 | 20 | 44 | 24 | 59 | 28 | 52 |
| 17 | 67 | 21 | 60 | 25 | 76 | 29 | 64 |
| 18 | 74 | 22 | 62 | 26 | 65 | 30 | 54 |
| 19 | 52 | 23 | 77 | 27 | 54 | 31 | 77 |

# Attack principle



# Basic Attack (“long” IVs)

- We choose the best IV pairs for each interesting register
- $\Rightarrow$ Global pair $(IV_0, IV_1)$
- Probability of global collision:

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

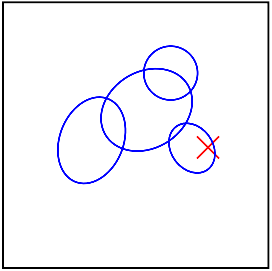
- Take a random value of 11 bytes $IV_{\text{rand}}$
- IV setups with IVs : $(IV_{\text{rand}}||IV_0, IV_{\text{rand}}||IV_1)$
- Collision is easy to observe

# Basic Attack ("long" IVs)

- Problem: this attack requires 23–byte IVs
  - 11 bytes for randomization
  - 12 bytes for the local collision pattern
- We would like to use shorter IVs
- We cannot reduce the length of the collision pattern
- Shorter randomization $\Rightarrow$ attacks fails for some keys

# Advanced Attack ("short" IVs)

- Replace single IV pair by several IV pairs
  - Many pairs covering a large portion of the state space
- Minimal IV length: 12 bytes
  - Requires a complete covering of the state space



# Advanced Attack ("short" IVs)

- How to build this covering?
- On a single register : greedy algorithm
- Notations :
  - $\mathcal{S}(P)$ : colliding set of an IV pair
  - $|A|$ : cardinality of A
- Build the colliding sets for each IV pairs $P$
- Sort them by decreasing $|\mathcal{S}(P)|$
- $i = 0$
- while (true)
  - Select the first IV pair : $P_i = (IV_0^i, IV_1^i)$
  - if $\mathcal{S}(P_i) = \varnothing$ return
  - Remove $x \in \mathcal{S}(P_i)$ from $\mathcal{S}(P), P \notin \{P_j\}$
  - Sort $P \notin \{P_j\}$ by decreasing $|\mathcal{S}(P)|$, i++

# Advanced Attack (“short” IVs)

- It is possible to build complete coverings of the state space for all update functions $g_i$

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

- Combining these families we get a global covering of the state space of the interesting registers
- Cardinality $\approx 2^{31.69}$
- During the search we test global pairs by decreasing number of additional detected states
- Average number of IV pairs tested $\approx 2^{27.73}$

# Results

- The two presented chosen IV attacks can be used as a distinguisher
- Complexity

| | IV setups | Time | Memory |
|---|---|---|---|
| “long” IV | $2^{22.74}$ | $2^{22.74}$ | 1 |
| “short” IV (worst case) | $2^{32.69}$ | $2^{32.69}$ | $2^{20}$ |
| “short” IV (average case) | $2^{28.73}$ | $2^{28.73}$ | $2^{20}$ |

# Partial keyed state recovery

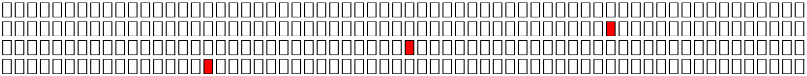
- Once we have obtained a collision on the IV setup, we can recover 53 bits of the keyed state
- Idea : process each register separetely
  - guess the state of the register (small set of candidates)
  - modify the IV pair only for the selected register and verify the guess
- "long" IV attack test:
  - modify the random IV entering the register
  - make an IV setup with the modified IV pair
  - check the guessed value
- "short" IV attack test:
  - select another pair for the register
  - make an IV setup with the modified IV pair
  - check the guessed value

# Partial keyed state recovery

- Complexity far smaller than the IV collision search
- We recover the value of the 5 interesting registers after the key setup
- With the recovered data, can we do better than exhaustive key search?
- Yes:
    - Attack with related keys
    - Meet-in-the-middle attacks

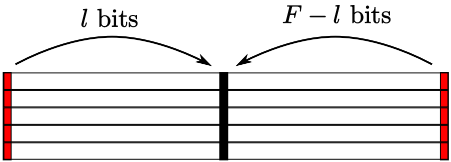
# Related key attacks

- With few related keys we can efficiently recover the key :
- The keys differ only on one bit



- **Algorithm:**
  - First recover the interesting registers
  - Guess the last bits of the key
  - Backtrack the states until just after the difference introduction
  - Check the difference
- **Result:** with 8 related keys for VEST-8 with a 128-bit key
  - perform 8 times the chosen IV attack $\approx 2^{26}$ IV setups
  - guess 8 times 16 bits $\approx 2^{19}$ key introduction backtracking

# Naive meet in the middle attack




- We know 5 registers states before and after key introduction
- Classical meet in the middle attack
- Time/Memory tradeoff
- Requires $2^{\max(F-l, F-53)}$ time and $2^l$ memory

# Realistic meet in the middle attack

- The previous model is unrealistic:
  - Accessing an element in a big memory is expensive
  - Exhaustive key search time complexity can be improved by using more processing power
- D. Bernstein proposed an attacking machine in a model taking into account processing power.
- **Result** :
  - for VEST-8 with 128–bit keys the key can be recovered in $2^{64}$ computations of the middle state and key tests using $2^{32}$ processors $\simeq$ a 100–bit exhaustive key search.

# VEST status

- Ability to distinguish its output from random : YES
- Ability to recover the key faster than exhaustive key search : YES
- Ability to recover the key faster than the claimed security level : $\simeq$

# VEST status

- Ability to distinguish its output from random : YES
- Ability to recover the key faster than exhaustive key search : YES
- Ability to recover the key faster than the claimed security level : $\simeq$

# VEST status

- Ability to distinguish its output from random : YES
- Ability to recover the key faster than exhaustive key search : YES
- Ability to recover the key faster than the claimed security level : $\simeq$

# Conclusion

- VEST is vulnerable to chosen IV attacks
  - Despite its complexity, VEST has simple weaknesses
  - Attacks recover 53 bits of the keyed state (implemented)
  - (VEST MAC mode is broken)
- IV setups MUST be collision free
- Following our attack, the authors proposed to modify the counter diffusor to remove the collision we exploited
- Attacks do not apply anymore
- The worrying differential properties of the counter remains

# Conclusion

- VEST is vulnerable to chosen IV attacks
  - Despite its complexity, VEST has simple weaknesses
  - Attacks recover 53 bits of the keyed state (implemented)
  - (VEST MAC mode is broken)
- IV setups MUST be collision free
- Following our attack, the authors proposed to modify the counter diffusor to remove the collision we exploited
- Attacks do not apply anymore
- The worrying differential properties of the counter remains