TWO GENERIC METHODS OF ANALYZING STREAM CIPHERS

Lin Jiao, Bin Zhang, Mingsheng Wang
Chinese Academy of Sciences

Outline

- Introduction
- Time-Memory-Data Tradeoff Attack against Grain-v1
- Security Evaluation of ACORN
- Conclusion

TCA

Outline

- Introduction
- Time-Memory-Data Tradeoff Attack against Grain-v1
- Security Evaluation of ACORN
- Conclusion

TCA

Introduction

- Motivation

  The security analysis against stream ciphers becomes more difficult;

  It is urgent and significant to propose new generic methods.

- Essential point:

  We introduce guess-and-determine techniques to two traditional analysis methods: Time-Memory-Data Tradeoff and Linear Approximation;

  We make the new approaches methodological for generalization.

- Attack models:

  We show the power of the new methods by analyzing two stream ciphers: Grain-v1 and ACORN.

Introduction

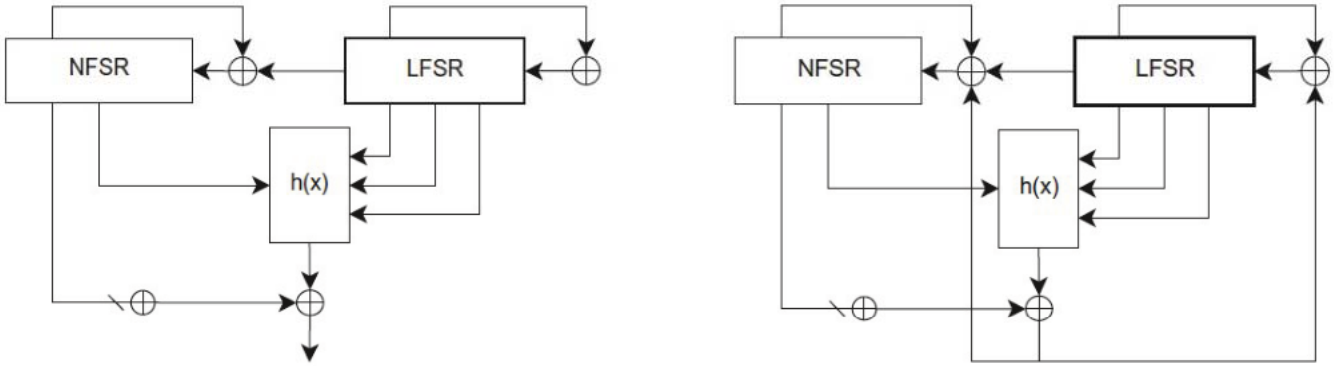


Fig. 1. Grain-v1: Keystream generation mode and Key initialization mode

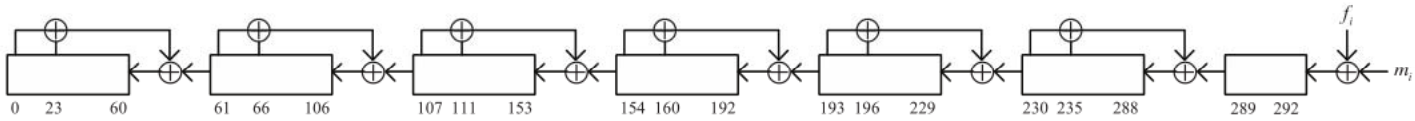


Fig. 2. ACORN-128

Stream cipher

TCA

Outline

- Introduction
- Time-Memory-Data Tradeoff Attack against Grain-v1
- Security Evaluation of ACORN
- Conclusion

TCA

TMD Tradeoff Attack against Grain-v1

Introduction of Time-Memory-Data Tradeoff Attack

- Procedure

  offline step: build large tables relating to the function in question.

  online step: obtain several actual data points and try to find a preimage of at least one value using precomputed tables.

- Parameters

  the size of the search space N, the time of precomputation P,
  the memory for precomputed tables M, the time of online phase T,
  the data required D.

- Development

  TM: Hellman $P = N, TM^2 = N^2, 1 \le T \le N$

  TMD: Babbage and Golic $P = M,\ N = TM$ and $T = D$

  Biryukov, Shamir and Wagner $P = N / D, N^2 = TM^2D^2, D^2 \le T$

TMD Tradeoff Attack against Grain-v1

Description of Grain-v1

- Grain-v1 is a bit-oriented stream cipher taking an 80-bit key and a 64-bit IV.
- The internal state of Grain-v1 has 160 bits, and it consists of an 80-bit LFSR $(s_0,\cdots,s_{79})$ and an 80-bit NFSR $(b_0,\cdots,b_{79})$.
- The update functions of the LFSR and NFSR are

$$s_{t+80} = s_{t+62} \oplus s_{t+51} \oplus s_{t+38} \oplus s_{t+23} \oplus s_{t+13} \oplus s_t$$

$$\begin{aligned} b_{t+80} =& s_t \oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \oplus b_{t+37} \oplus b_{t+33} \oplus b_{t+28} \oplus b_{t+21} \\ & \oplus b_{t+14} \oplus b_{t+9} \oplus b_t \oplus b_{t+63}b_{t+60} \oplus b_{t+37}b_{t+33} \oplus b_{t+15}b_{t+9} \\ & \oplus b_{t+60}b_{t+52}b_{t+45} \oplus b_{t+33}b_{t+28}b_{t+21} \oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \\ & \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \oplus b_{t+63}b_{t+60}b_{t+21}b_{t+15} \\ & \oplus b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \oplus b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \\ & \oplus b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}. \end{aligned}$$

TMD Tradeoff Attack against Grain-v1

Description of Grain-v1

- During the keystream generation, Grain-v1 outputs a single bit at each clock cycle . The output function is defined as

$$z_t = h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}) \oplus \sum_{j \in \mathrm{A}} b_{t+j},$$

where $\mathrm{A}=\{1,2,4,10,31,43,56\}$ and the filter function $h(x)$ is given by

$$\begin{aligned} h(x) =& x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \\ & \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4, \end{aligned}$$

where the variables x_0, x_1, x_2, x_3 and x_4 of $h(x)$ correspond to the tap positions $s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}$ and b_{t+63}

TMD Tradeoff Attack against Grain-v1

Description of Grain-v1

- Grain-v1 is initialized with a 64-bit IV injected directly into the LFSR (the remaining bits of the LFSR are assigned value one), and an 80-bit key that is loaded into the NFSR.
- Then the cipher is clocked 160 times without producing any keystream, but feeding the output bits back into both the LFSR and the NFSR.
- The state update function of Grain-v1 is invertible both during keystream generation and key initialization.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

- A Boolean function is said to be *k*-normal if it is constant on a *k*-dimensional flat.
- Further generalization: a Boolean function is said to be *k*-linear-normal if its restriction is a linear function on a *k*-dimensional flat.
- For the filter function $h(x)$ of Grain-v1, we deduce all the linear operation modes by setting certain constraints on the state.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

- Consider a stream cipher with n-bit state. If given a value of $n-l$ special state bits of the cipher and the first l bits of the keystream sequence generated from that internal state, the remaining l bits of the internal state can be recovered directly, then the sampling resistance is defined as $R = 2^{-l}$.
- We can enumerate the special states and associate with each special state a short name of $n-l$ bits, and a short output of $n-l$ bits.
- The sampling resistance of Grain-v1 is at most 2^{-18}, which is obviously not enough.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

- How to make l longer?
- Our idea is associating the technique of k-linear-normality with sampling resistance by fixing some state bits.
- Under the constraints of state bits, the filter function of the remaining bits is linear. Thus we can substitute more state bits with keystream bits, i.e., extend the sampling resistance.
- Also, we can reduce the space of guessed bits both impacted by the constraints and sampling resistance.
- We call it conditional sampling resistance, which supports tradeoff parameters in larger range.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

- We present the specific conditional sampling resistance for Grain-v1, based on the guess-and-determine strategy
- We choose two of those linear operation modes according to the *k*-linear-normality of the filter function *h*(x):
  - Let $x_2 = 0, x_3 = 1$, i.e., $s_{t+46} = 0, s_{t+64} = 1$, then $h(x) = x_0 \oplus x_1 = s_{t+3} \oplus s_{t+25}$,
  - Let $x_0 = 1, x_1 = 0, x_2 = 1$, i.e., $s_{t+3} = 1, s_{t+25} = 0, s_{t+46} = 1$, then $h(x) = x_3 = s_{t+64}$.
- Combining the linear modes with the output function and the update functions of both LFSR and NFSR, we show the guessing path.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

Step	Constraint conditions	Concerned keystream bits	Guessed NFSR and LFSR bits	Recovered bit
0	$s_{46}=0, s_{64}=1$	z_0	$b_1, b_2, b_4, b_{31}, b_{43}, b_{56}, s_3$	b_{10}
1	$s_{47}=0, s_{65}=1$	z_1	$b_3, b_5, b_{32}, b_{44}, b_{57}, s_4$	b_{11}
2	$s_{48}=0, s_{66}=1$	z_2	$b_6, b_{33}, b_{45}, b_{58}, s_5$	b_{12}
3	$s_{49}=0, s_{67}=1$	z_3	$b_7, b_{34}, b_{46}, b_{59}, s_6$	b_{13}
4	$s_{50}=0, s_{68}=1$	z_4	$b_8, b_{35}, b_{47}, b_{60}, s_7$	b_{14}
5	$s_{51}=0, s_{69}=1$	z_5	$b_9, b_{36}, b_{48}, b_{61}, s_8$	b_{15}
6	$s_{52}=0, s_{70}=1$	z_6	$b_{37}, b_{49}, b_{62}, s_9, s_{31}$	b_{16}
7	$s_{53}=0, s_{71}=1$	z_7	$b_{38}, b_{50}, b_{63}, s_{10}, s_{32}$	b_{17}
8	$s_{54}=0, s_{72}=1$	z_8	$b_{39}, b_{51}, b_{64}, s_{11}, s_{33}$	b_{18}
9	$s_{55}=0, s_{73}=1$	z_9	$b_{40}, b_{52}, b_{65}, s_{12}, s_{34}$	b_{19}
10	$s_{56}=0, s_{74}=1$	z_{10}	$b_{41}, b_{53}, b_{66}, s_{13}, s_{35}$	b_{20}
11	$s_{57}=0, s_{75}=1$	z_{11}	$b_{42}, b_{54}, b_{67}, s_{14}, s_{36}$	b_{21}
12	$s_{58}=0, s_{76}=1$	z_{12}	$b_{55}, b_{68}, s_{15}, s_{37}$	b_{22}
13	$s_{59}=0, s_{77}=1$	z_{13}	b_{69}, s_{16}, s_{38}	b_{23}
14	$s_{60}=0, s_{78}=1$	z_{14}	b_{70}, s_{17}, s_{39}	b_{24}
15	$s_{61}=0, s_{79}=1$	z_{15}	b_{71}, s_{18}, s_{40}	b_{25}
16	$s_{22}=1, s_{44}=0, \overline{s_{65}}=1$	z_{19}	b_{75}	b_{29}
17	$s_{23}=1, s_{45}=0, \overline{s_{66}}=1$	z_{20}	b_{76}	b_{30}
18	$s_{24}=1, \overline{s_{46}}=0, \overline{s_{67}}=1$	z_{21}	–	b_{77}
19	$s_{19}=s_{20}=s_{28}=1$	$z_{16}=0$	b_{72}, b_{73}	s_0
	$s_{41}=s_{42}=\overline{s_{50}}=0$	$z_{17}=0$		
	$s_{62}=s_{63}=\overline{s_{71}}=1$	$z_{25}=0$		
20	$\overline{s_{20}}=s_{21}=s_{29}=1$	$z_{17}=0$	b_{74}	s_1
	$\overline{s_{42}}=s_{43}=\overline{s_{51}}=0$	$z_{18}=0$		
	$\overline{s_{63}}=\overline{s_{64}}=\overline{s_{72}}=1$	$z_{26}=0$		
21	$s_{30}=1, \overline{s_{52}}=0, \overline{s_{73}}=1$	z_{27}	–	b_{28}
22	$\overline{s_{21}}=1, \overline{s_{43}}=0, \overline{s_{64}}=1$	z_{18}	–	s_2
23	$\overline{s_{29}}=1, \overline{s_{51}}=0, \overline{s_{72}}=1$	z_{26}	–	b_{27}
24	$\overline{s_{28}}=1, \overline{s_{50}}=0, \overline{s_{71}}=1$	z_{25}	–	b_{26}
25	$s_{25}=1, \overline{s_{47}}=0, \overline{s_{68}}=1$	z_{22}	–	b_{78}
26	$s_{26}=1, \overline{s_{48}}=0, \overline{s_{69}}=1$	z_{23}	–	b_{79}
26	$s_{27}=1, \overline{s_{49}}=0, \overline{s_{70}}=1$	z_{24}	–	b_0

– Overline denotes that this condition has already been assigned.

TMD Tradeoff Attack against Grain-v1

Preliminary Analysis

In summary, given the guess-and-determine strategy, we derive that by fixing 51 bits of state constraint conditions and guessing 81 bits more of the internal state, the remaining 28 bits of the state can be recovered directly using the first 28 keystream output bits generated from the state.

TMD Tradeoff Attack against Grain-v1

Time-Memory-Data Tradeoff Attack

Preprocessing Phase:

1. Choose a fixed string $s \in \{0,1\}^{28}$ as a segment of keystream.
2. Form a $m \times t$ matrix that tries to cover the whole search space which is composed of all the possible guessed 81 bits of NFSR and LFSR states as follows.
   (a) Randomly choose m startpoints of the chains, each point formed by a vector of 81 bits which is to be an injection into the guessed positions of NFSR and LFSR.
   (b) Under the constraint conditions of 51 bits, the remaining 28 bits of the state can be recovered using the segment of keystream s according to the guessing path. Thus, the overall system is obtained. Perform a backward computation of the system and generate the former 81 keystream bits from this moment on. Make it the next point in the chain, i.e., update the injection into NFSR and LFSR with this point.
   (c) Iterate Step (b) t times on each startpoint respectively.
   (d) Store the pairs of startpoints and endpoints $(SP_j, EP_j), j = 1, \ldots, m$ in a table.

TMD Tradeoff Attack against Grain-v1

Time-Memory-Data Tradeoff Attack

Realtime Phase:

1. Observe the keystream and find 2^{51} number of 28-bit strings matching with string s. For one such string, let its former 81 bits in the keystream be y.
2. For each y, check if there is EP_j, $j = 1,\ldots,m$ matching with y first. If not, iterate Step (b) w times on y until it matches with one of EP_j, $j = 1,\ldots,m$, where $w = 1,\ldots,t$. When there is a match, jump to the corresponding startpoint, and repeatedly apply Step (b) to the startpoint until the 81-bit keystream vector reaches y again. Then the previous point visited is the 81 guessed state bits of NFSR and LFSR, and the whole initial state of the cipher is recovered by jointing it with the 51 constraint state bits and 28 derived state bits.

TMD Tradeoff Attack against Grain-v1

Complexity Analysis and Comparison

- The whole search space is composed of all the possible guessed NFSR and LFSR bits, whose cardinality is $2^{81} = m \cdot t = P$
- The memory complexity is $M = m$ for the storage of the startpoints and endpoints table.
- Since the matrix is built under the constraint conditions of 51 bits, we expect to encounter a state among the matrix given 2^{51} selected data. Moreover, we need to sample $D = 2^{51} \cdot 2^{28}$ consecutive keystream bits to collect the required strings, since the string *s* of length 28 bits occurs on average once in 2^{28} keystream bits.
- For each selected data, we need to calculate Step(b) at most *t* times, and the time complexity T equals $2^{51} \cdot t$.
- Here, we choose $m=2^{71}$, $t=2^{10}$, then we get a group of tradeoff parameters as follows $T = 2^{61}, M = 2^{71}, D = 2^{79}, P = 2^{81}.$

TMD Tradeoff Attack against Grain-v1

Complexity Analysis and Comparison

- We compare our attack with previously reported TMD trade-off state recovery attacks against Grain-v1 in the single key and IV pair setting.

Resource	Time online (T)	keystream (D)	Memory (M)	Preprocessing time (P)
BSW TMD	2^{80}	2^{40}	2^{80}	2^{120}
[8]	2^{71}	$2^{53.5}$	2^{71}	$2^{106.5}$
new	2^{61}	2^{79}	2^{71}	2^{81}

- Our figures appear as significantly better than the previously reported ones, since the preprocessing time can be controlled much lower.

- There are still several TMD attacks against Grain-v1 in the case of multi key and IV pairs, or different initial values.

TMD Tradeoff Attack against Grain-v1

Complexity Analysis and Comparison

- We transform the tradeoff parameters into cipher ticks.
- For the preprocessing time, Step (b) needs to run backwards 81 cipher ticks. Thus we need the precomputation of $P = 2^{81} \cdot 81 = 2^{87.3}$ ticks.
- The memory is $M = m \cdot 81 \cdot 2$ bits for storing the pairs of 81-bit length points.
- The time online taken is $T = 2^{51} \cdot t \cdot 81$.
- We choose $m = 2^{71}$, $t = 2^{10}$, the memory is $M = 2^{78.3}$ bits and the time online is $T = 2^{67.3}$ cipher ticks.
- As a baseline, we analyzed the time complexity of the brute force attack against Grain-v1. Actually, the complexity of brute force attack is $2^{87.4}$, higher than 2^{80} ticks.
- Our complexities are lower than that of brute force attack. The brute force attack can only be mounted for each fixed IV, while our attack can be applied to any IV.

TMD Tradeoff Attack against Grain-v1

Complexity Analysis and Comparison

- It is a generic approach to analyze stream ciphers.
- We summarize the tradeoff curve of the new TMD attack.
- Given r state bits of constraint conditions and g guessed state bits, the remaining /state bits can be recovered by the first /keystream bits generated from the state.

$$D = 2^{r+l} = N/2^g = N/P,$$
$$TMD = t2^r m2^{r+l} = 2^r 2^{g+r+l} = 2^r N.$$

Outline

- Introduction
- Time-Memory-Data Tradeoff Attack against Grain-v1
- Security Evaluation of ACORN
- Conclusion

TCA

Security Evaluation of ACORN

Description of ACORN

- ACORN-128 uses a 128-bit key and a 128-bit IV.
- The state size is 293 bits, and there are six LFSRs being concatenated in ACORN-128.
- Two Boolean functions are used in ACORN:

$$maj(x, y, z) = (x\&y) \oplus (x\&z) \oplus (y\&z); \quad ch(x, y, z) = (x\&y) \oplus ((\sim x)\&z),$$

Security Evaluation of ACORN

Description of ACORN

- One step of ACORN is done as follows:

$$S_{i,289} = S_{i,289} \oplus S_{i,235} \oplus S_{i,230};$$
$$S_{i,230} = S_{i,230} \oplus S_{i,196} \oplus S_{i,193};$$
$$S_{i,193} = S_{i,193} \oplus S_{i,160} \oplus S_{i,154};$$
$$S_{i,154} = S_{i,154} \oplus S_{i,111} \oplus S_{i,107};$$
$$S_{i,107} = S_{i,107} \oplus S_{i,66} \oplus S_{i,61};$$
$$S_{i,61} = S_{i,61} \oplus S_{i,23} \oplus S_{i,0};$$
$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193});$$
$$f_i = S_{i,0} \oplus (\sim S_{i,107}) \oplus maj(S_{i,244}, S_{i,23}, S_{i,160}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66}) \oplus (ca_i \& S_{i,196}) \oplus (cb_i \& ks_i);$$
$$\text{for } j := 0 \text{ to } 291 \text{ do } S_{i+1,j} = S_{i,j+1};$$
$$S_{i+1,292} = f_i \oplus m_i; ca_i = 1, cb_i = 0$$

and $c_i = p_i \oplus ks_i$

Security Evaluation of ACORN

Security Evaluation of ACORN

- One observation is that the maj function can be linearly approximated with a big probability.
- At each step, the probability that any one of these three variables x,y,z equals the value of the maj function is 3/4; any two of these three variables is 1/2; all these three is 1/4.
- Since $ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193})$, it is easy to get several linear equations before that the nonlinear feedback bits shift into the register and become one tap going into the keystream generating function used for linear approximation.
- Then there can be 139, 100 and 58 steps for linear equations when using the approximation to $S_{i,61}$ $S_{i,193}$, $S_{i,235}$

Security Evaluation of ACORN

Security Evaluation of ACORN

- To receive more linear equations, we consider the feedback bits.

$$\begin{aligned} f_i =& S_{i,0} \oplus (\sim S_{i,107}) \oplus maj(S_{i,244}, S_{i,23}, S_{i,160}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66}) \oplus S_{i,196} \\ =& S_{i,0} \oplus S_{i,23} \oplus S_{i,66} \oplus (\sim S_{i,107}) \oplus S_{i,196} \oplus (S_{i,23} \oplus S_{i,160})\&(S_{i,23} \oplus S_{i,244}) \\ & \oplus (S_{i,66} \oplus S_{i,111})\&S_{i,230}. \end{aligned}$$

- Thus we only need to guess two bits combination information, then the feedback bit becomes liner. Hence, the feedback bit can be used in the approximation accordingly.
- Moreover, we get another two linear equations.
- It is easy to transform the state variables at each step into the initial ones linearly.

Security Evaluation of ACORN

Security Evaluation of ACORN

- Let the number of steps using just one, two and three variables approximations be *a*, *b* and *c*, respectively. Let the number of feedback steps be *f*.
- We transform the balancing problem into an integer linear programming problem (ILP) as follows.

$$\text{maximum} \quad Pr := (\frac{3}{4})^a \cdot (\frac{1}{2})^b \cdot (\frac{1}{4})^c \cdot (\frac{1}{2})^{2f}$$

$$\begin{cases} a + 2b + 3c + 2f \geq 293 \\ t := a + b + c = 139 + f \\ b + c \leq 100 + f \\ c \leq 58 + f \\ a, b, c, f \geq 0, \text{ are integers.} \end{cases}$$

- We use Maple to solve the optimization problem, and the result is a=41, b=112, c=0, f=14. Here the goal $Pr = 2^{-157}$, and t=153.

Security Evaluation of ACORN

Security Evaluation of ACORN

- Algorithm

1. Guess 28 bits information to make the feedback bits at 14 steps linear and get 28 linear equations in the initial state variables.
2. For every guess, collect 265 linear approximating equations at optional steps according to the results of the ILP problem.
3. Given 153 keystream bits of ACORN, recover the state of ACORN, and verify the solution.
4. Repeat the loops from Step 1 to 3 until the right initial state is found.

Security Evaluation of ACORN

Security Evaluation of ACORN

- Analysis
  - If the solution is not the real state, both the error of linear approximation and the guessed information can lead to the inaccuracy
  - We expect a right solution among 2^{157} tests according to the success probability.

- We can view this method a generic way to evaluate the security of stream ciphers, which works as firstly finding the linear approximations of the output function and the efficient guessed combination information of the upstate function, then transforming the bounding problem into an integer linear programming problem for searching the optimal solution.

TCA

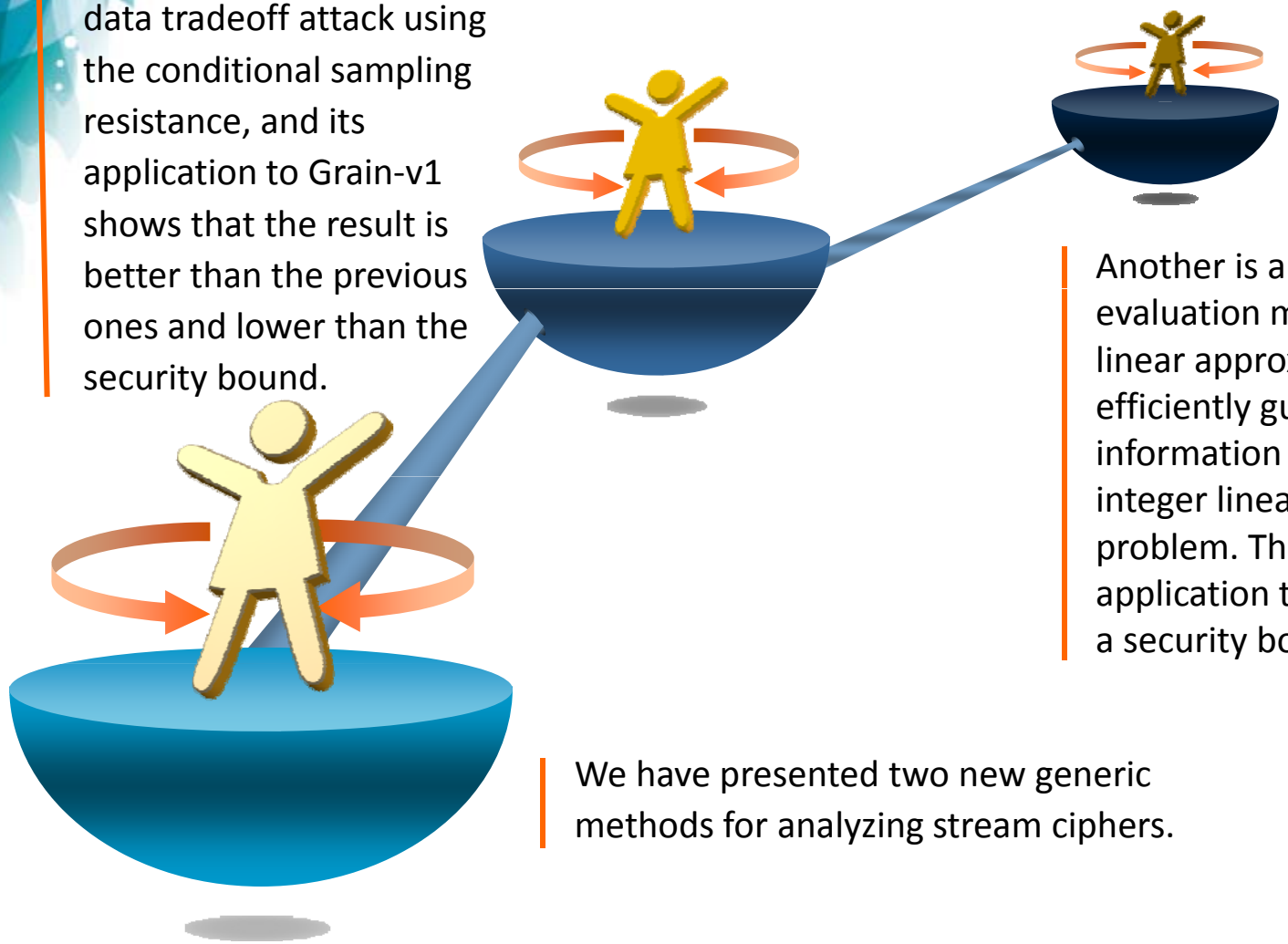
Outline

- Introduction
- Time-Memory-Data Tradeoff Attack against Grain-v1
- Security Evaluation of ACORN
- Conclusion

TCA

Conclusion

One is a time-memory-data tradeoff attack using the conditional sampling resistance, and its application to Grain-v1 shows that the result is better than the previous ones and lower than the security bound.



Another is a security evaluation method using linear approximations, efficiently guessed information and the tool of integer linear programming problem. The result of its application to ACORN gives a security bound of ACORN.

We have presented two new generic methods for analyzing stream ciphers.

TCA

THANK YOU!