

# Parallel Framework for Evolutionary Black-box Optimization with Application to Algebraic Cryptanalysis

presenter: Stepan Kochemazov

A. Pavlenko, A. Semenov, V. Ulyantsev, O. Zaikin

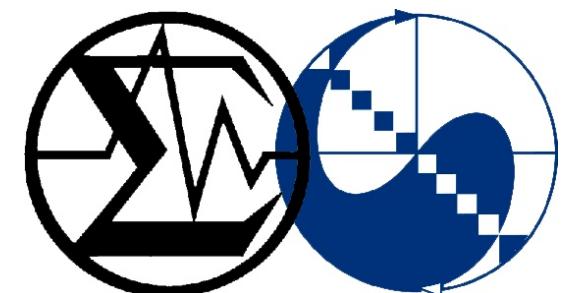
{alpavlenko,ulyantsev}@corp.ifmo.ru

biclop.rambler@yandex.ru

zaikin.icc@gmail.com

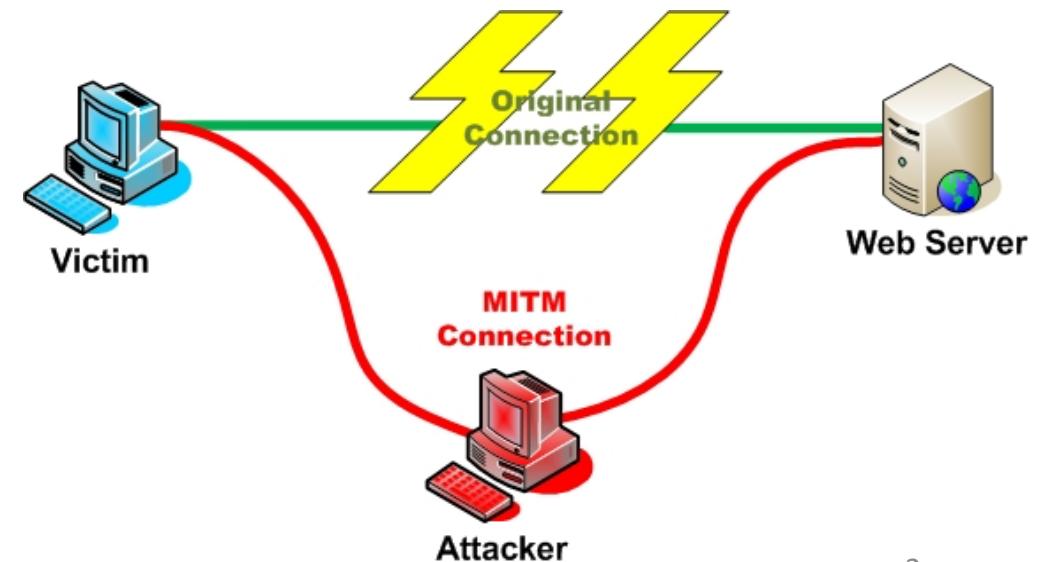
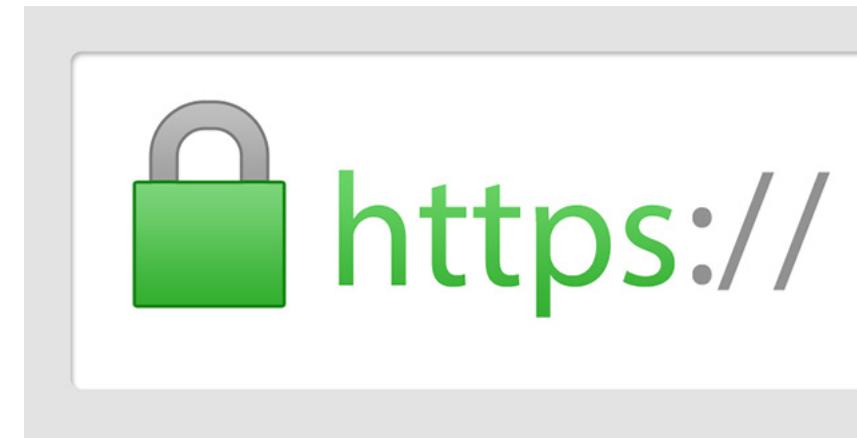
ITMO UNIVERSITY

ITMO University, St. Petersburg, Russia  
ISDCT SB RAS, Irkutsk, Russia



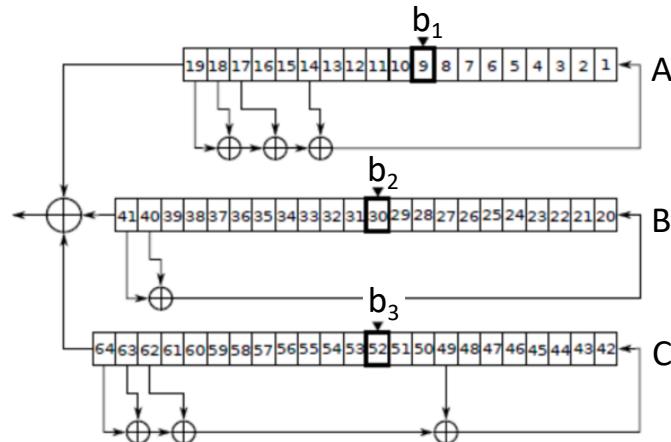
# Cryptanalysis

- There are a lot of ways to encode and to decode information
  - HTTPS, mobile traffic ...
  - man in the middle
- **Algebraic cryptanalysis** is a way of analyzing and breaking ciphers
- Type of attacks:
  - Brute-force attack
  - **Guess-and-determine** attack



# Stream ciphers and cryptanalysis

Cipher A5/1 – used in 2G protocol



$b_1, b_2, b_3$  – clocking bits

$$X = X_A \cup X_B \cup X_C$$

$$X = \{x_1, x_2, \dots, x_{64}\}$$

$$Y = \{y_1, y_2, \dots, y_{128}\}$$

$$\begin{aligned}f: \{0,1\}^{64} &\rightarrow \{0,1\}^{128} \\ f(x) &= y\end{aligned}$$

**Research question:** how practically hard it is to decrypt some encrypted text?

# SAT and SAT-solvers

$(\neg x_1 \vee \neg x_3 \vee \neg x_4)$

$(x_2 \vee x_3 \vee \neg x_4)$

$(x_1 \vee \neg x_2 \vee x_4)$

$(x_1 \vee x_3 \vee x_4)$

$(\neg x_1 \vee x_2 \vee \neg x_3)$

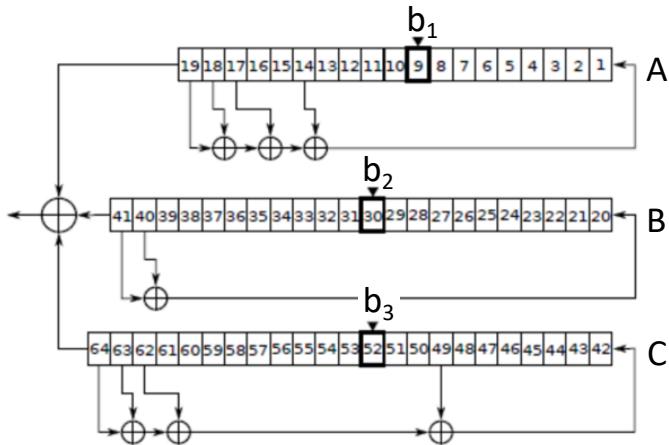
- Boolean SATisfiability – first known NP-complete problem
- A dozen of **applicable** SAT-solvers
  - minisat, lingeling, ROKK ...
  - SAT, **UNSAT**
- Annular competitions in solving SAT!



good idea to **translate** hard problem to SAT

# Encode to SAT using Transalg\*

Cipher A5/1



$b_1, b_2, b_3$  – clocking bits

$$X = X_A \cup X_B \cup X_C$$

$$X = \{x_1, x_2, \dots, x_{64}\}$$

$$Y = \{y_1, y_2, \dots, y_{128}\}$$

manually  
⇒

Transalg program

```

1 __in bit XA[19];
2 __in bit XB[22];
3 __in bit XC[23];
4 __out bit Y[128];

5
6 bit shift_rslosA() {
7     bit x0 = XA[18]^XA[17]^XA[16]^XA[13];
8     for(int j = 18; j > 0; j=j-1) {
9         XA[j] = XA[j-1];
10    }
11    XA[0] = x0;
12 }
13 ...
14
15 bit majority(bit A, bit B, bit C) {
16     return A&B|A&C|B&C;
17 }
18
19 void main() {
20     int b1 = 8;
21     int b2 = 10;
22     int b3 = 10;
23     bit maj;
24     for(int i = 0; i < 128; i=i+1) {
25         maj = majority(XA[b1], XB[b2], XC[b3]);
26         if(!(maj^XA[b1])) shift_rslosA();
27         if(!(maj^XB[b2])) shift_rslosB();
28         if(!(maj^XC[b3])) shift_rslosC();
29         Y[i] = XA[18]^XB[21]^XC[22];
30     }
31 }
```

automatically  
⇒

SAT-formula

```

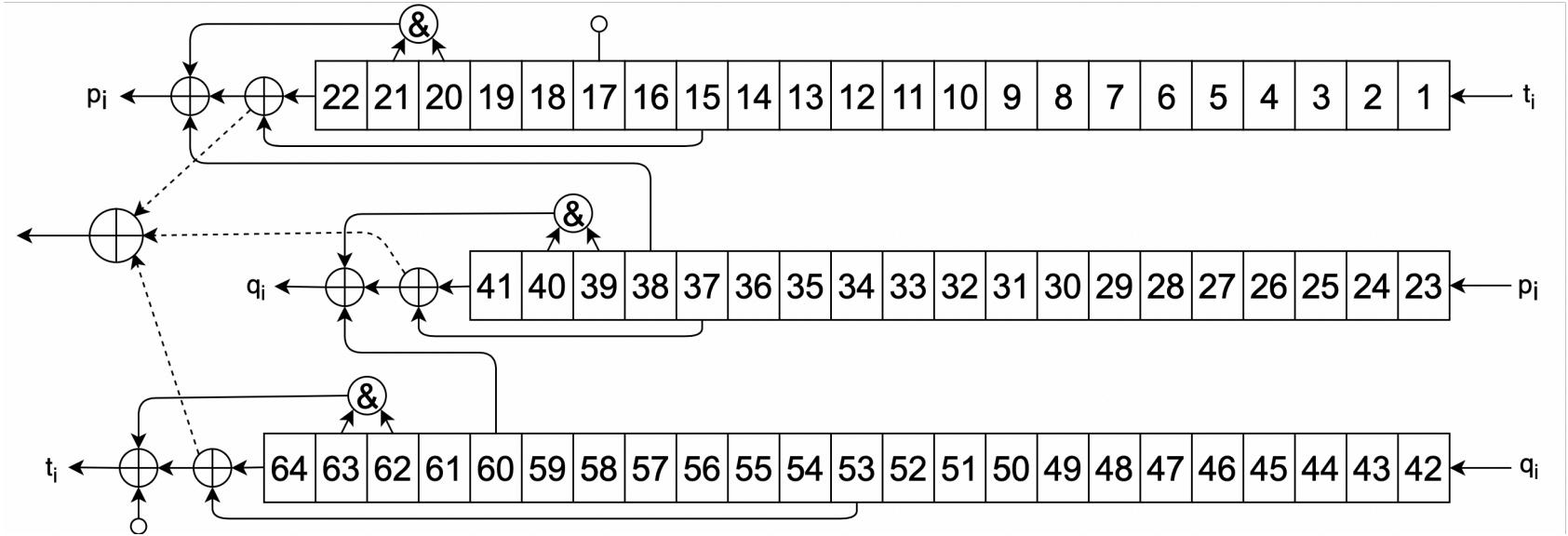
1 p cnf 8425 38262
2 c input variables 64
3 c literals count 128374
4 65 9 30 0
5 65 9 52 0
6 -65 9 -30 -52 0
7 65 -9 -52 0
8 -65 -9 30 52 0
9 65 -9 -30 0
10 66 -19 65 0
11 66 -18 -65 0
12 -66 19 65 0
13 -66 18 -65 0
14 67 -18 65 0
15 67 -17 -65 0
16 -67 18 65 0
17 -67 17 -65 0
18 68 -17 65 0
19 68 -16 -65 0
20 -68 17 65 0
21 -68 16 -65 0
22 69 -16 65 0
23 69 -15 -65 0
24 -69 16 65 0
...
...
38264 -8425 -8293 8295 -8297 0
38265 -8425 -8293 -8295 8297 0
```

\*Transalg: [Otpuschennikov, I., Semenov, A., Gribanova, I., Zaikin, O., Kochemazov, S.: Encoding Cryptographic Functions to SAT Using TRANSALG System. In: ECAI 2016. FAIA, vol. 285, pp. 1594–1595 (2016)]

# Example of breaking for Trivium 64

**CPU:**  
AMD Opteron 6276  
@ 2.3 GHz x32

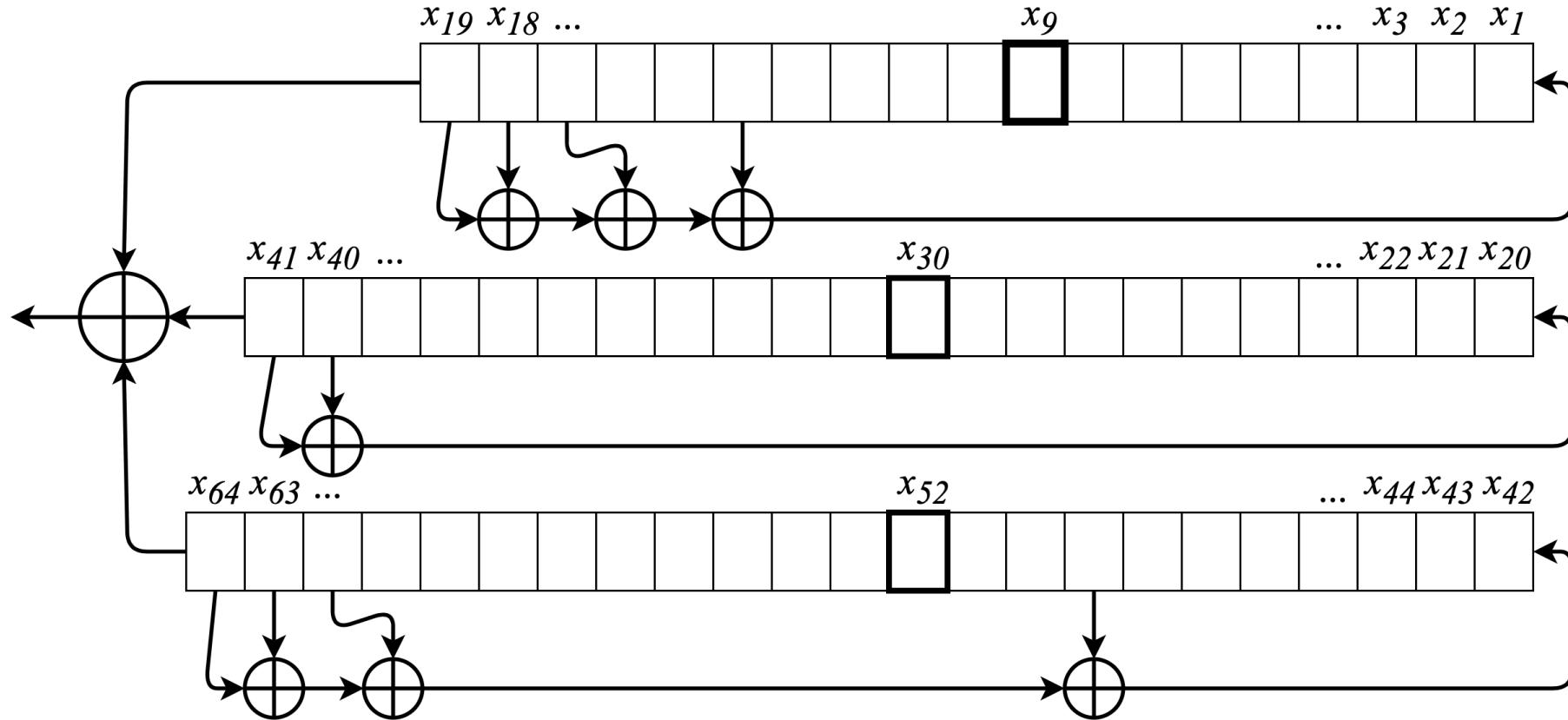
**Timelimit:**  
7 days



	PLingeling	Treengeling	Guess-and-determine attack
task 1	interrupted	interrupted	2d 6h
task 2	interrupted	3d 2h	3d 19h
task 3	interrupted	4d 10h	15h
task 4	interrupted	interrupted	1d 21h
task 5	interrupted	interrupted	4d 3h

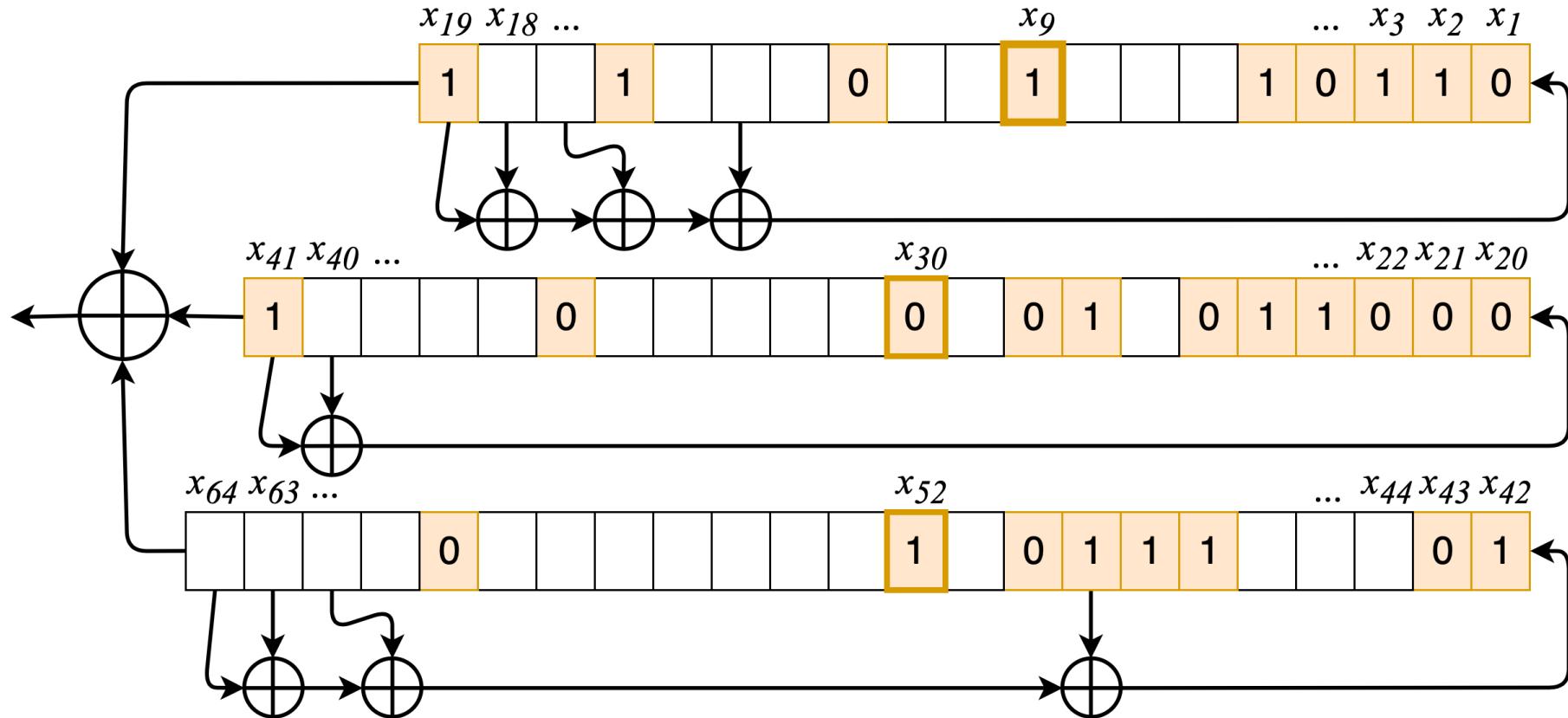
## 2. Guess-and-determine attacks

# Guess-and-Determine. Backdoor



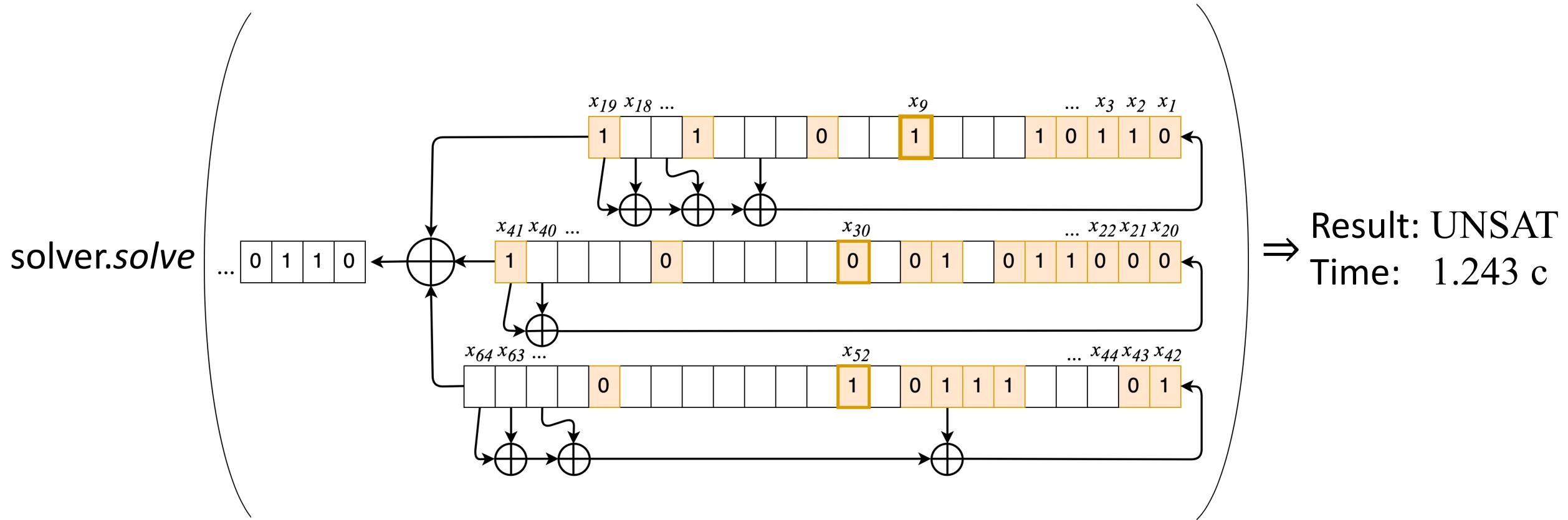
$$B = \{ x_1, x_2, x_3, x_4, x_5, x_9, x_{12}, x_{16}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, \\ x_{25}, x_{27}, x_{28}, x_{30}, x_{36}, x_{41}, x_{42}, x_{43}, x_{47}, x_{48}, x_{49}, x_{50}, x_{52}, x_{60} \}$$

# Guess-and-Determine. Guess



$$B = \{x_1, x_2, x_3, x_4, x_5, x_9, x_{12}, x_{16}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{27}, x_{28}, x_{30}, x_{36}, x_{41}, x_{42}, x_{43}, x_{47}, x_{48}, x_{49}, x_{50}, x_{52}, x_{60}\}$$

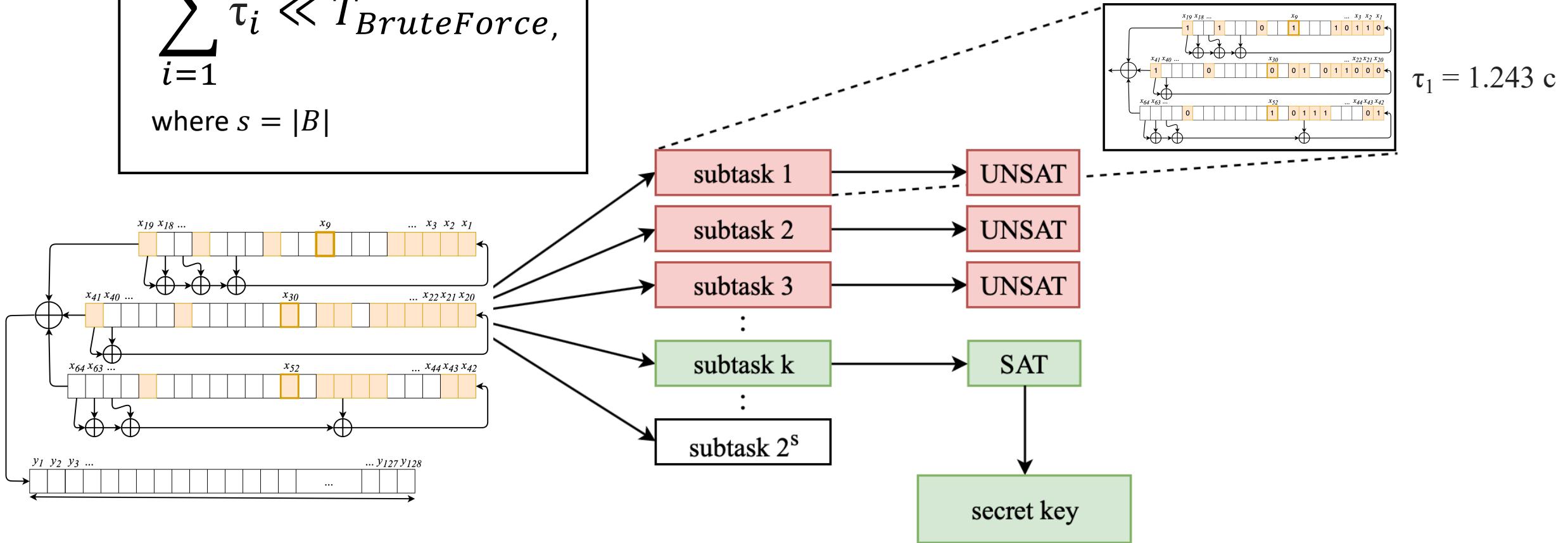
# Guess-and-Determine. Determine



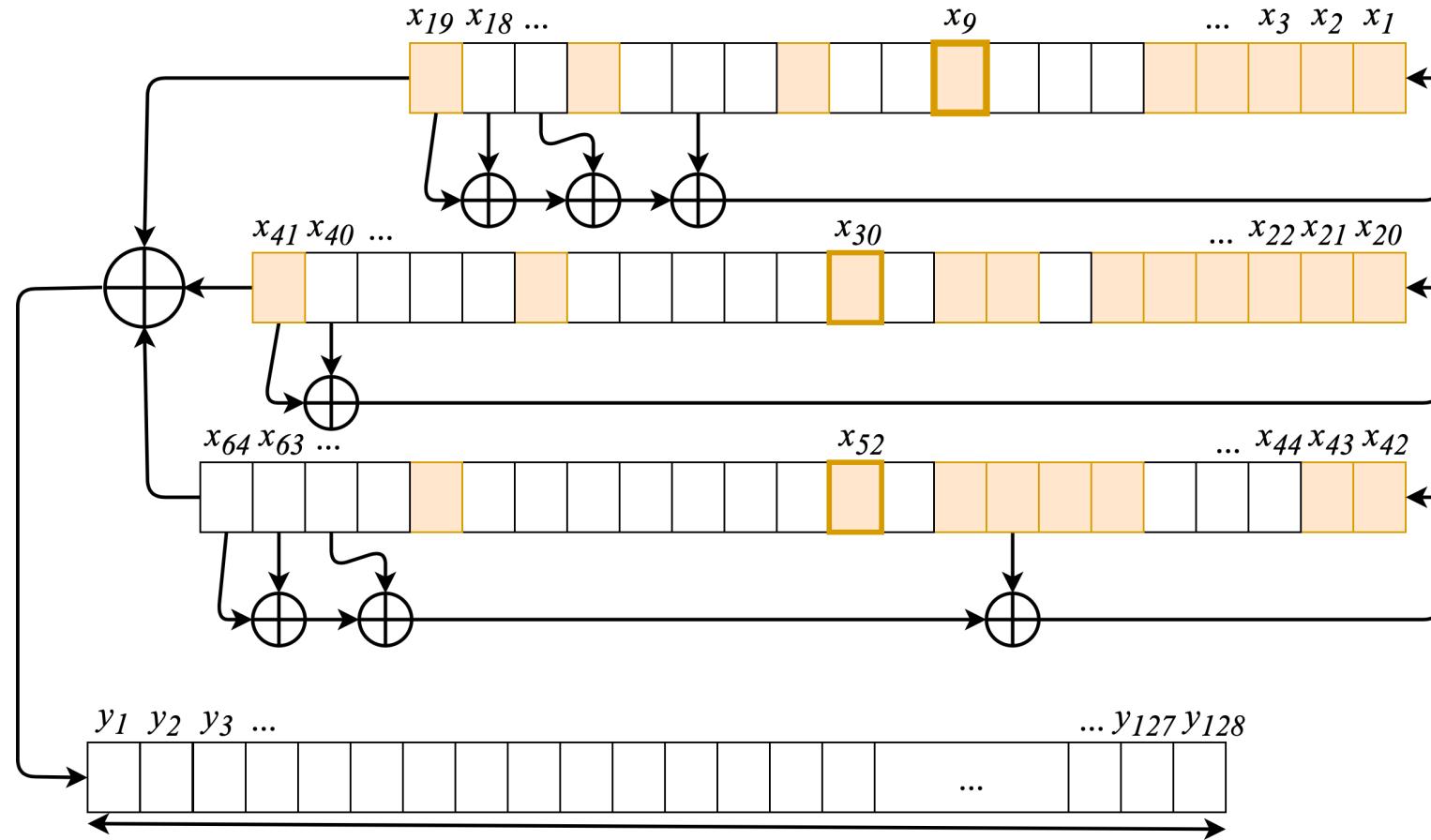
# Guess-and-Determine. Definition

$$\sum_{i=1}^{2^s} \tau_i \ll T_{BruteForce},$$

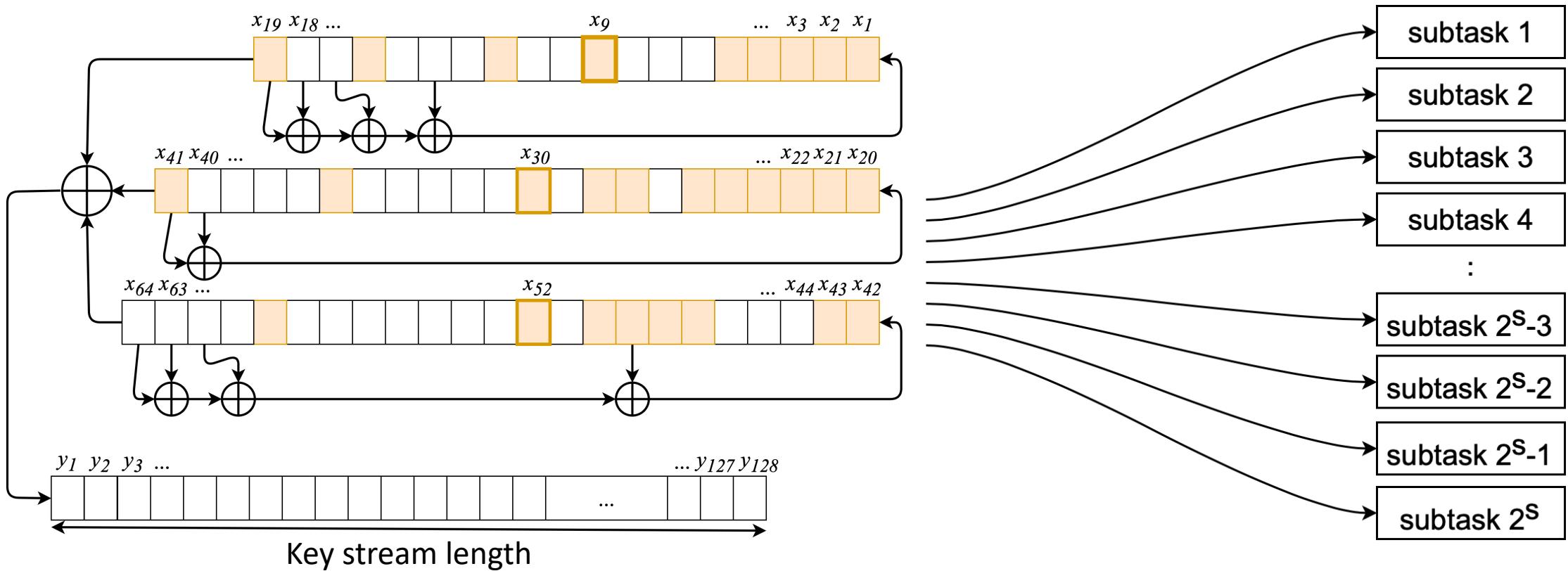
where  $s = |B|$



# How to construct a efficient backdoor?

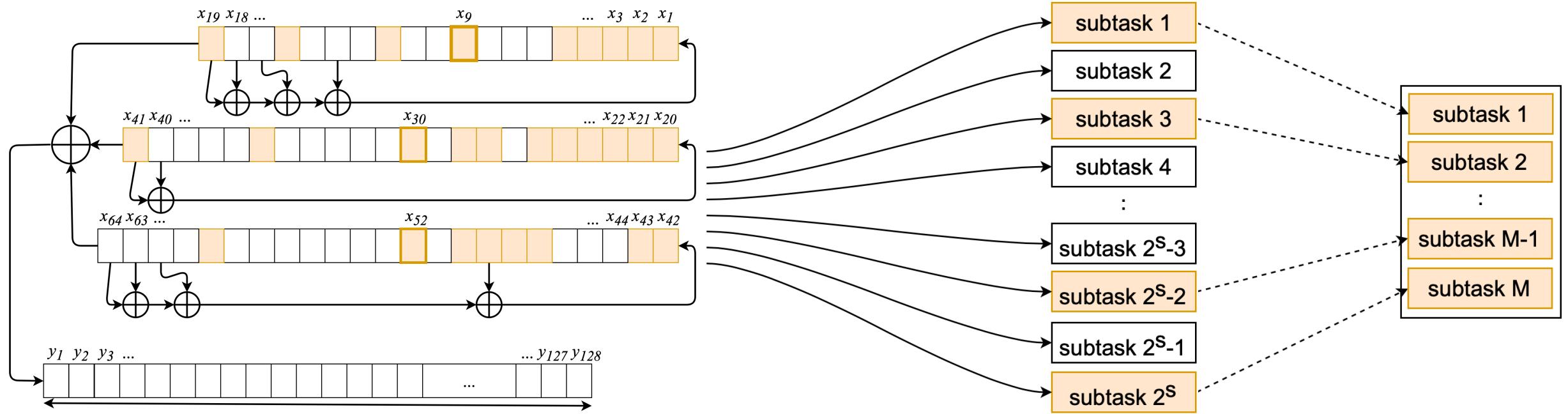


# Backdoor-based Decomposition



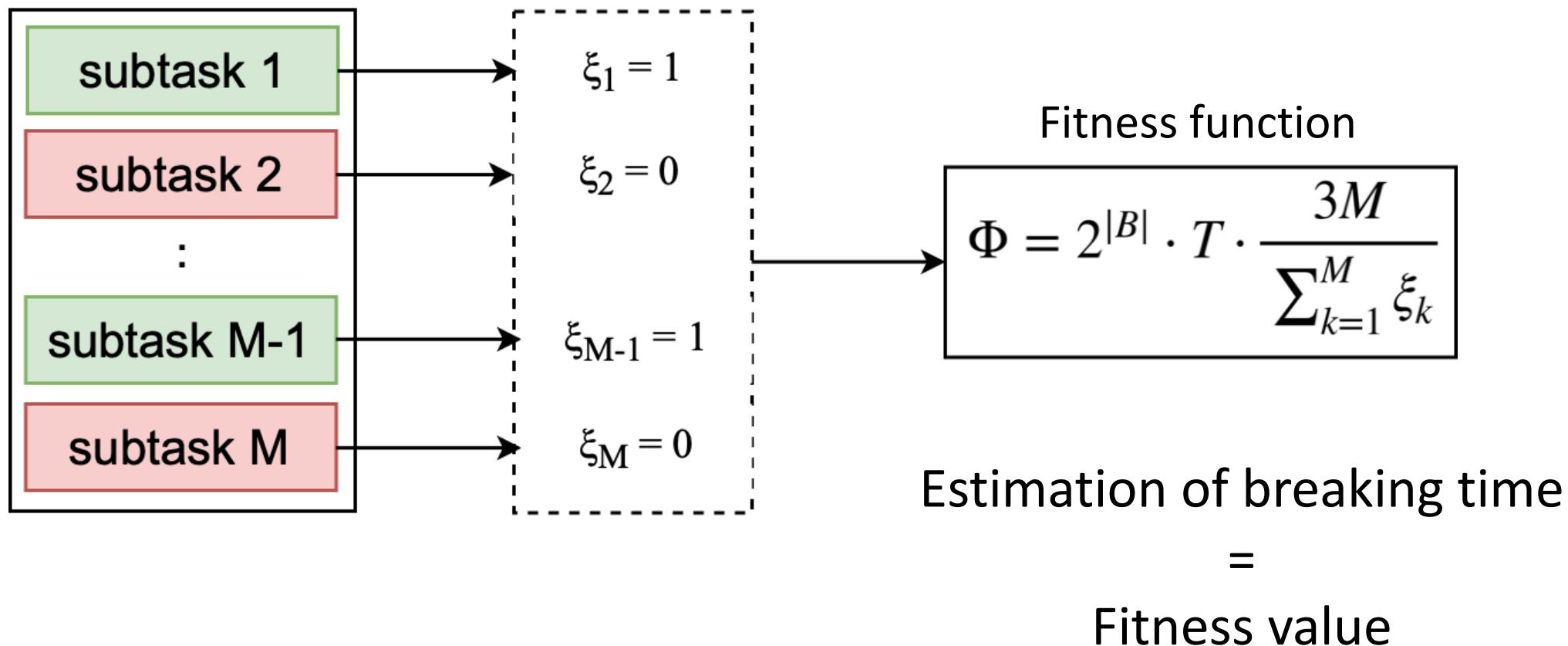
$$s = |B| - \text{power of backdoor set}$$

# Monte-Carlo Sampling



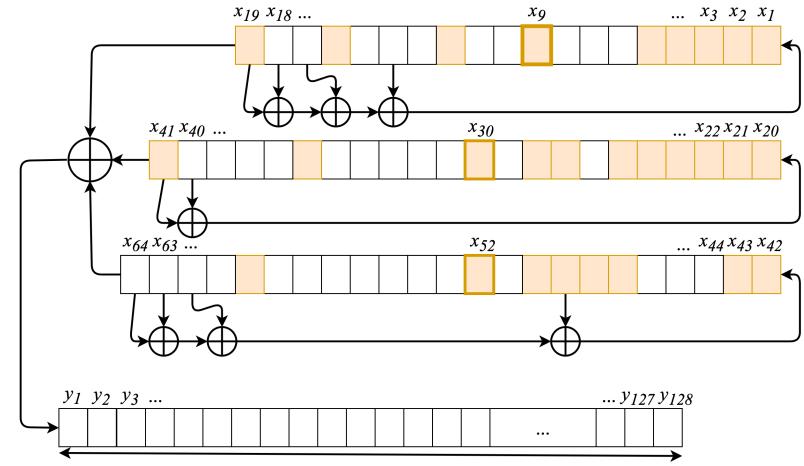
# Evaluating

If the task is solved in time  $T$ ,  
then  $\xi = 1$ , else  $\xi = 0$



# Intermediate sum-up

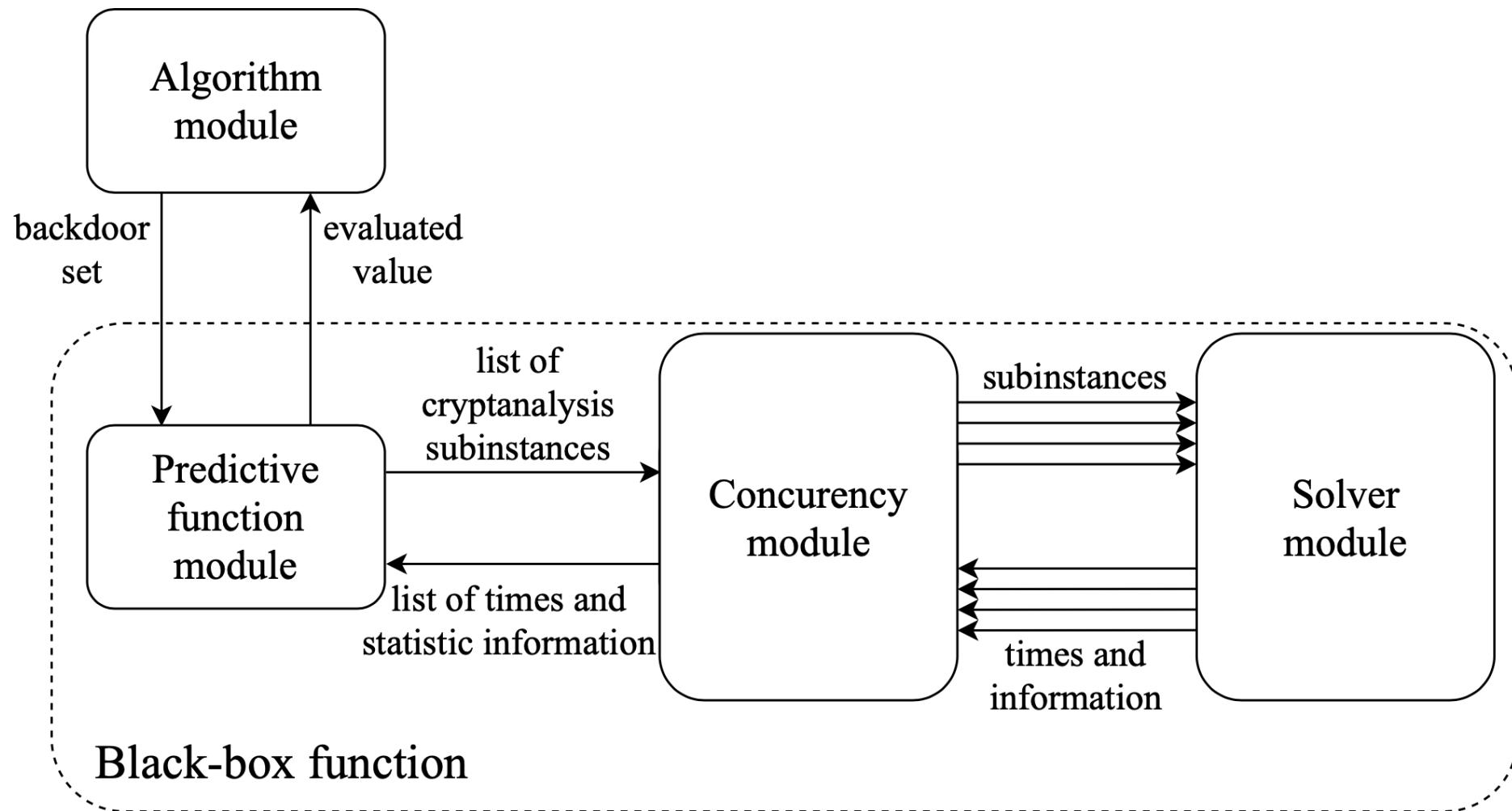
- Analyzing **stream cyphers** is a hard problem
- We can **translate** the attack to **SAT**
- We can **speedup** the SAT-based attack using **backdoor**
- Selecting the **efficient** backdoor is a hard problem
- But there is a way to **estimate** the attack time for a given backdoor



magic  
↓  
Estimation of  
breaking time

### 3. Framework for minimizing a fitness function

# Framework scheme

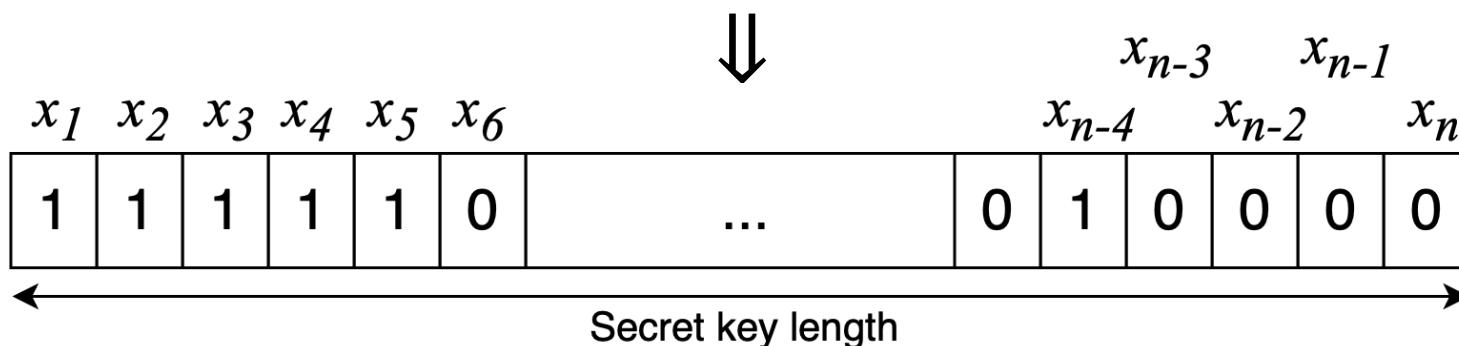


# Algorithm module

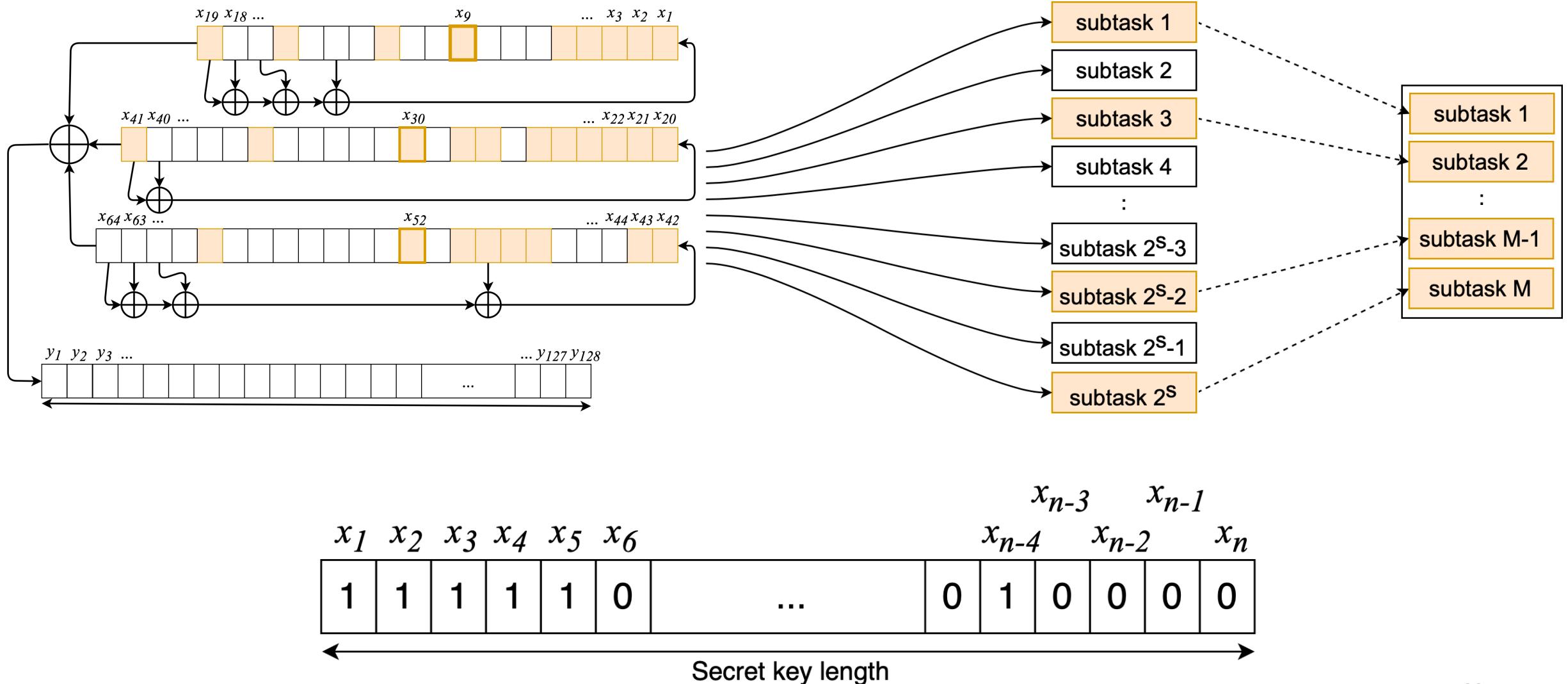
We apply	Framework supports
(1+1)-EA	$(\mu, \lambda)$ -EA
GA (Elitism)	$(\mu+\lambda)$ -EA Tabu Search Simulated Annealing

Individual: bit vector, which presents a backdoor set

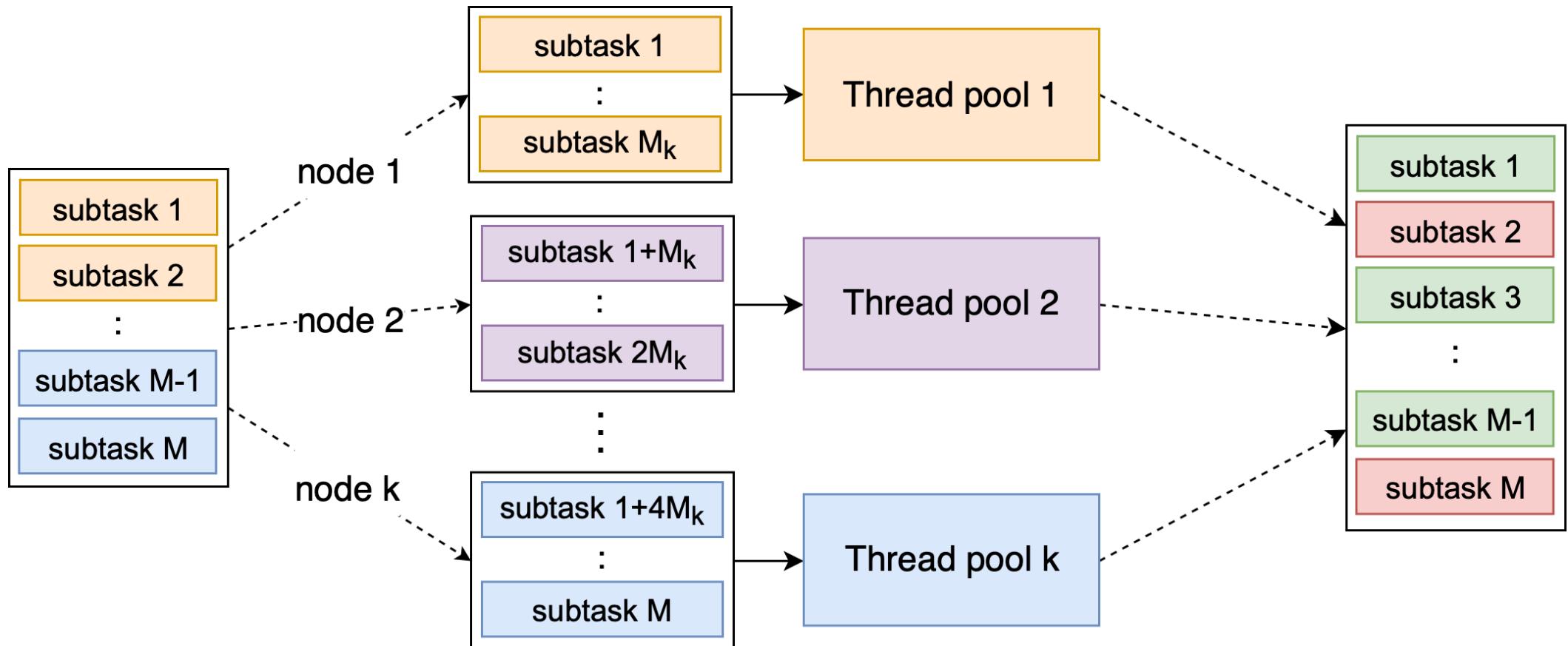
$$B = \{ x_1, x_2, x_3, x_4, x_5, x_9, x_{12}, x_{16}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, \\ x_{25}, x_{27}, x_{28}, x_{30}, x_{36}, x_{41}, x_{42}, x_{43}, x_{47}, x_{48}, x_{49}, x_{50}, x_{52}, x_{60} \}$$



# Predictive function module

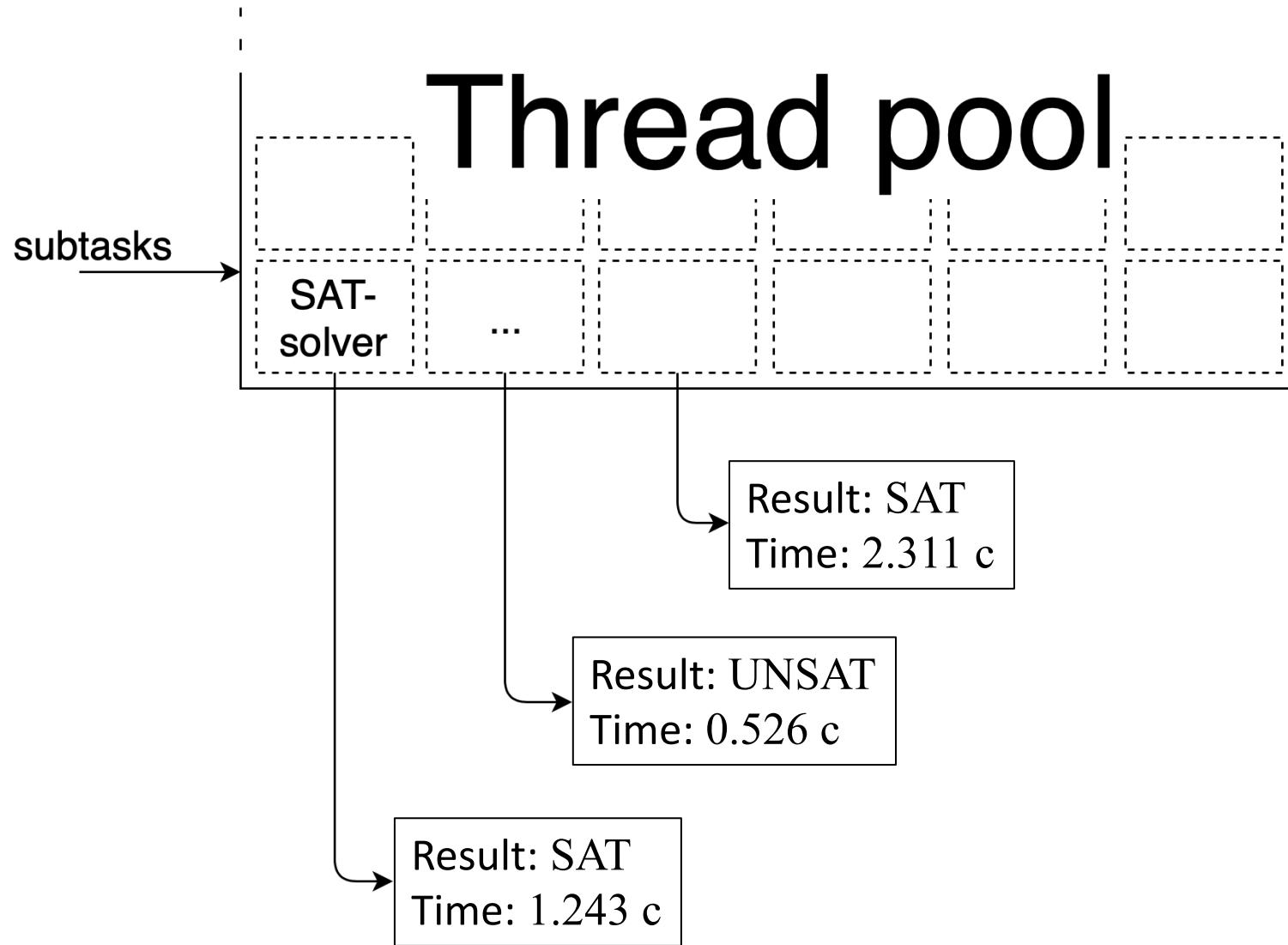


# Concurrency module



$$M_k = \frac{M}{k}, k - \text{count of nodes}$$

# Solver module



Implemented wrappers:

- MiniSat
- Lingeling
- Plingeling
- Treengeling
- ROKK
- CryptoMiniSat
- PalnLeSS

# Predictive function module. Evaluating

$$\Phi_1 = 2^{|B|} \cdot \frac{\sum_{k=1}^M t_k}{M}$$

$t_k$  – time of solving task  $k$

$$\Phi_2 = 2^{|B|} \cdot T \cdot \frac{3M}{\sum_{k=1}^M \xi_k}$$

If the task is solved in time  $T$ ,  
then  $\xi = 1$ , else  $\xi = 0$

# Experimental results. First function

	ALIAS*		EvoGuess (1+1)-EA	
	B	Attack time (s)	B	Attack time (s)
Grain v1 160/160	109	4.04e+30	100	7.51e+30
Trivium 288/300	144	1.40e+41	143	3.51e+43
Mickey 200/250	158	1.56e+48	169	1.77e+51

\*ALIAS: [Zaikin O., Kochemazov S. Pseudo-Boolean Black-Box Optimization Methods in the Context of Divide-and-Conquer Approach to Solving Hard SAT Instances. In DEStech Transactions on Computer Science and Engineering, pp. 76-87 (2018)]

# Experimental results. Second function

	EvoGuess (1+1)-EA		EvoGuess GA	
	B	Attack time (s)	B	Attack time (s)
Grain v1 160/160	104	4.71e+32	103	7.23e+31
Trivium 288/300	136	1.52e+43	146	5.08e+43
Mickey 200/250	159	9.73e+50	152	8.18e+50

# Conclusion

- We propose **new framework** for algebraic cryptanalysis.
- We used **(1+1)-EA** and **GA** to construct SAT-based guess-and-determine attacks on symmetric ciphers.
- We could not outperform ALIAS, so we are planning to significantly extend the framework's spectrum of pseudo-Boolean optimization algorithms and improve the search for guessed bits via tuning parameters of the used SAT solvers.
- Supposed by the Russian Science Foundation (project No 18-71-00150)

# Thank you for attention!

presenter: Stepan Kochemazov

Artem Pavlenko,  
Alexander Semenov,  
Vladimir Ulyantsev,  
Oleg Zaikin

{alpavlenko,ulyantsev}@corp.ifmo.ru  
biclop.rambler@yandex.ru  
zaikin.icc@gmail.com  
[instagram.com/itmo.ctlab](https://instagram.com/itmo.ctlab)

