PROOFS: Security Proofs for Embedded Systems

29th September, 2017 | Taipei, Taiwan

An Automated Framework for Exploitable Fault Identification in Block Ciphers – A Data Mining Approach

Sayandeep Saha, Ujjawal Kumar, Debdeep Mukhopadhyay, and Pallab Dasgupta

Outline

- Introduction
- Motivation
- Fault Attack Automation – State-of-the-art
- Proposed Approach
- Case Studies
- Conclusion

Introduction

Fault Analysis

Adversary injects faults into cryptosystems and analyzes the faulty output to recover the key

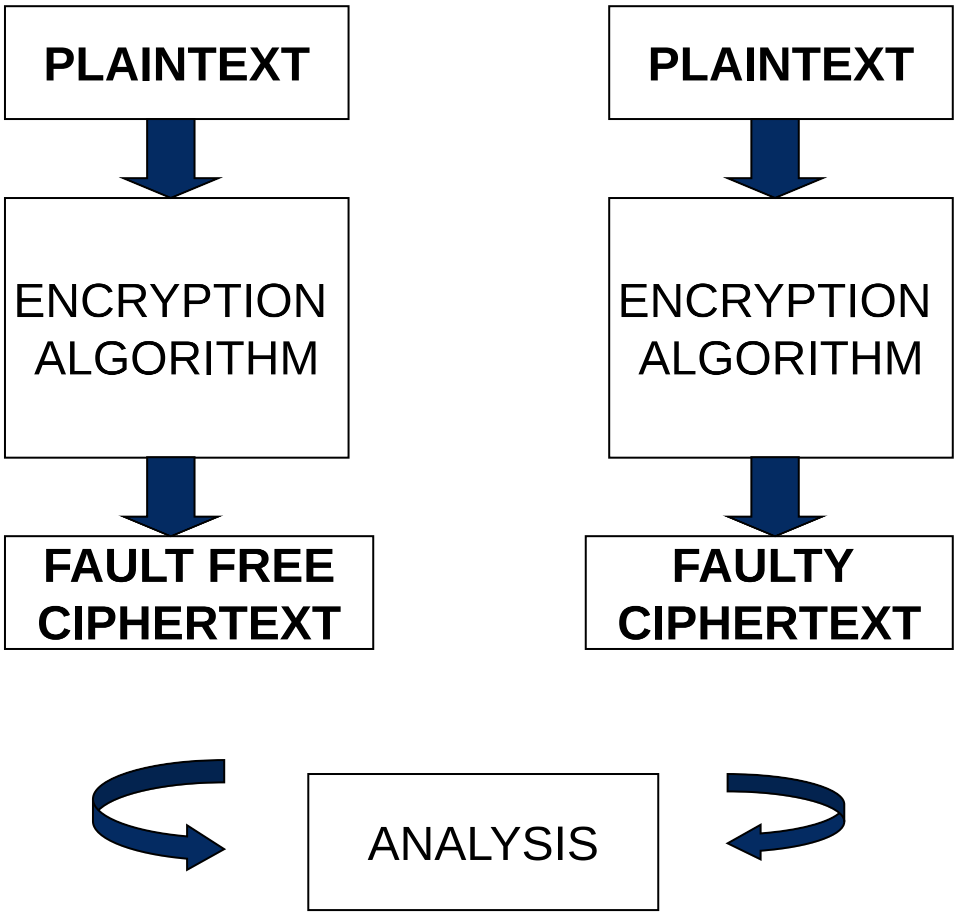
Easy to perform, does not require high end equipment

Must design efficient countermeasures against fault attacks

Weakens even mathematically robust cryptosystems

Introduction

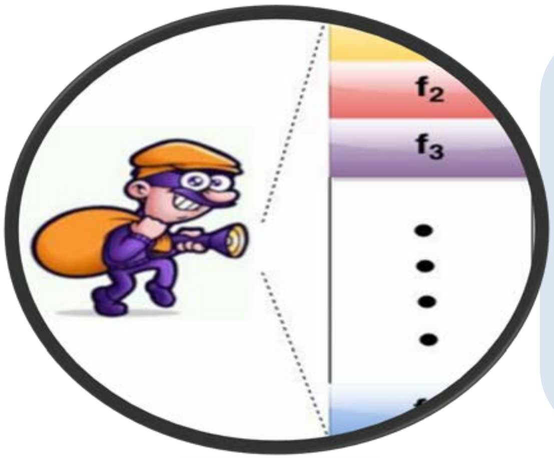
Differential Fault Analysis (DFA)



- Most widely explored
- Low fault complexity
- Complex analysis
- Fault Locations
  - Datapath
  - Key-schedule
- Fault models
  - Bit based
  - Nibble based
  - Byte based
  - Multiple byte based

Motivation (1/2)

Testing Block Ciphers for Fault Attacks



- Knowledge of all possible attacks
  - **Exploitable Faults**
- Not every fault is exploitable
  - Filter out the space of exploitable faults

"Exploits" of Exploitable Faults

- Designing precise countermeasures.
- Testing countermeasures
  - On "non-random" exploitable fault space.
- Cipher evaluation

Motivation (2/2)

Challenges

- Fault space for a block cipher:
  - Prohibitively large.
  - Manual fault analysis methodologies -- Impractical !!!
  - Demand: A fast automation to characterize each individual fault

- Several block ciphers in use
  - Trend of designing application-specific lightweight ciphers
  - Demand: A generic automation

Fault Attack Automation

State-of-the-art

Algebraic Fault Attack (AFA)

- Generic representation.
- Use of SAT solvers.
- Not so fast !!!
- Lack of interpretability.

F. Zhang et.al. , "A Framework for the Analysis and Evaluation of Algebraic Fault Attacks on Lightweight Block Ciphers", *IEEE Transactions on Information Forensics and Security*, *11*(5), 1039-1054., 2016

Synthesis of Fault attacks

- Program synthesis based
- Demonstrated on Public key systems

Gilles Barthe, et al. "Synthesis of fault attacks on cryptographic implementations." *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*., 2014.

Proposed Approach

- What DFA does?
  - Reduces the key search space with faults
  - Exhaustive search within practical limits

- What we suggest …
  - Do not perform the exhaustive search
  - Automatically compute the search complexity

- Advantage
  - Fast characterization of each individual faults
  - Challenge: Generic algorithm.

Proposed Approach (1/17)

- Formalization of DFA:
  - A DFA algorithm can be represented as follows:

$$\mathscr{A} = \langle \mathscr{D}, \mathscr{T}, \mathscr{R} \rangle$$

$\mathscr{D}$ A fault distinguisher, Constructed over the XOR differentials of a cipher state

$\mathscr{T}$ An algorithm to evaluate the distinguisher over key guesses

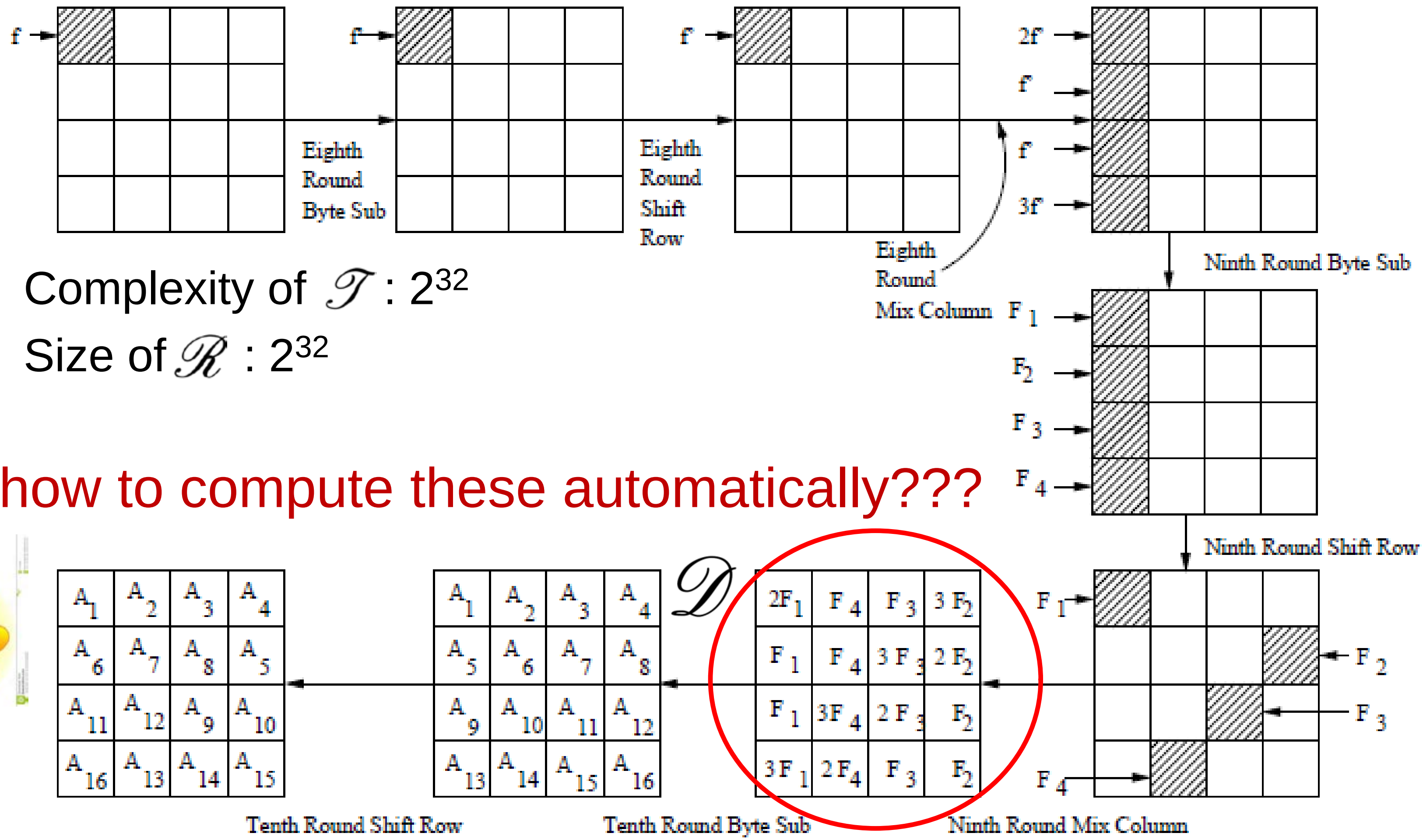
$\mathscr{R}$ Remaining key space, filtered with the distinguisher

Proposed Approach (2/17)

Formalization of DFA – Example of a Distinguisher

- Complexity of $\mathscr{T}$: 2^{32}
- Size of $\mathscr{R}$: 2^{32}

But how to compute these automatically???



Proposed Approach (3/17)

The Automatic Flow

Automated DFA Framework

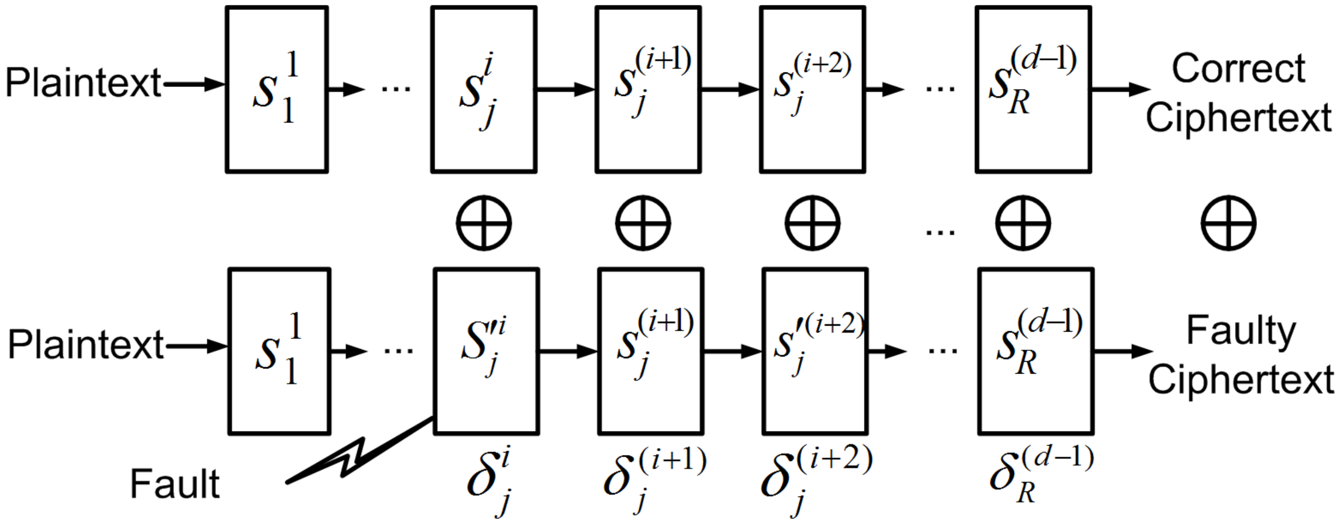
Phase 1	Phase 2	Phase 3
Automatic Identification of Distinguisher	Automatic Calculation of Distinguisher Evaluation Complexity	Automatic Calculation of Remaining Key Space Size

- Checks one fault at a time
- Fault models: Bit, Byte, Nibble, Multiple

Proposed Approach (4/17)

Phase 1: Automatic Distinguisher Identification

- Key Idea:
  - Identify the distinguishers from fault simulation data.



We analyze the dataset for each “state differential” δ_j^i

Proposed Approach (5/17)

Phase 1: How to Identify a Distinguisher from Data?

Calculate entropy *H* for each δ_j^i -- State Differential Entropy

Representation of δ_j^i: Set of nibbles/bytes

$$\delta_j^i = \langle w_1^{ij}, w_2^{ij}, \ldots, w_l^{ij} \rangle$$

Two Possible Cases

Case 1

Each w_z^{ij} is statistically independent

Case 2

Some of the w_z^{ij} s are statistically dependent

Proposed Approach (6/17)

Phase 1: Case 1

- State Differential Entropy:
  - State differential entropy is the sum of individual entropies of each w_z^{ij}
- Maximum Entropy of individual w_z^{ij}
  - 2^m, m is bit width of each w_z^{ij}

How to Calculate Entropy?

- Check the value ranges of each w_z^{ij}
  - If some values are always missing, it causes entropy reduction.
  - Save each occurring value for future use.

Proposed Approach (7/17)

Phase 1: Case 2

- Check dependency among w_z^{ij} s
  - Tricky, no general method to identify and enumerate the relations.
  - Solution: **Frequent Itemset Mining**
    - Itemsets exists ➡ statistical dependency

TID	v_1	v_2	v_3	v_4	v_5
1	1	5	7	8	11
2	2	4	6	9	13
3	1	5	7	10	2
4	2	4	6	11	4
5	3	9	8	6	5
6	1	10	11	9	8

Itemsets and Variable Sets

- <v_1, v_2, v_3> -- Variable set
  - Support: 2/6 = 0.33
  - Corresponding itemsets (1,5,7); (2,4,6)

Proposed Approach (8/17)

Phase 1: Case 2

- Collect variable sets and corresponding Itemsets:

$$\left(VS_{\delta_j^i},\ \{IS^{v}_{\delta_j^i}\}_{v=1}^{|\bar{V}S_{\delta_j^i}|}\right)$$

- State differential entropy is the sum of the entropies of the variable sets

$$
\begin{array}{l}
\textbf{for each } v \in VS_{\delta_j^i} \textbf{ do} \\
\quad tot := \texttt{VarCount}(v) \times m \\
\quad p'^{v}_{q} := \frac{1}{|IS^{v}_{\delta_j^i}|}, \forall q \in IS^{v}_{\delta_j^i} \\
\quad p'^{v}_{q} := 0, \forall q \notin IS^{v}_{\delta_j^i} \\
\quad H_{Assn}(v) := -\sum_{q=0}^{2^{tot}-1} p'^{v}_{q} \log_2(p'^{v}_{q}) \\
\quad H_{Assn}(\delta_j^i) := H_{Assn}(\delta_j^i) + H_{Assn}(v) \\
\textbf{end for}
\end{array}
$$

Proposed Approach (9/17)

Phase 1: Overall Flow

- Calculate state differential entropy
  - assuming independence of w_z^{ij} s: $H_{Ind}(\delta_j^i)$
  - if variable sets exist: $H_{Assn}(\delta_j^i)$
- Set State differential entropy, $H_j^i = \min(H_{Ind}, H_{Assn})$
- If $H_j^i < H_{max}(\delta_j^i)$: State differential is a key-distinguisher.

End of Phase 1:

Gives us distinguishers

$$\mathscr{D}_j^i := \langle \{w_z^{ij}\}_{z=1}^{l},\ \{Rng_{w_z^{ij}}\}_{z=1}^{l},\ VS_{\delta_j^i},\ \{IS_{\delta_j^i}^{v}\}_{v=1}^{|VS_{\delta_j^i}|} \rangle$$

Proposed Approach (10/17)

Phase 2: Calculate Distinguisher Evaluation Complexity

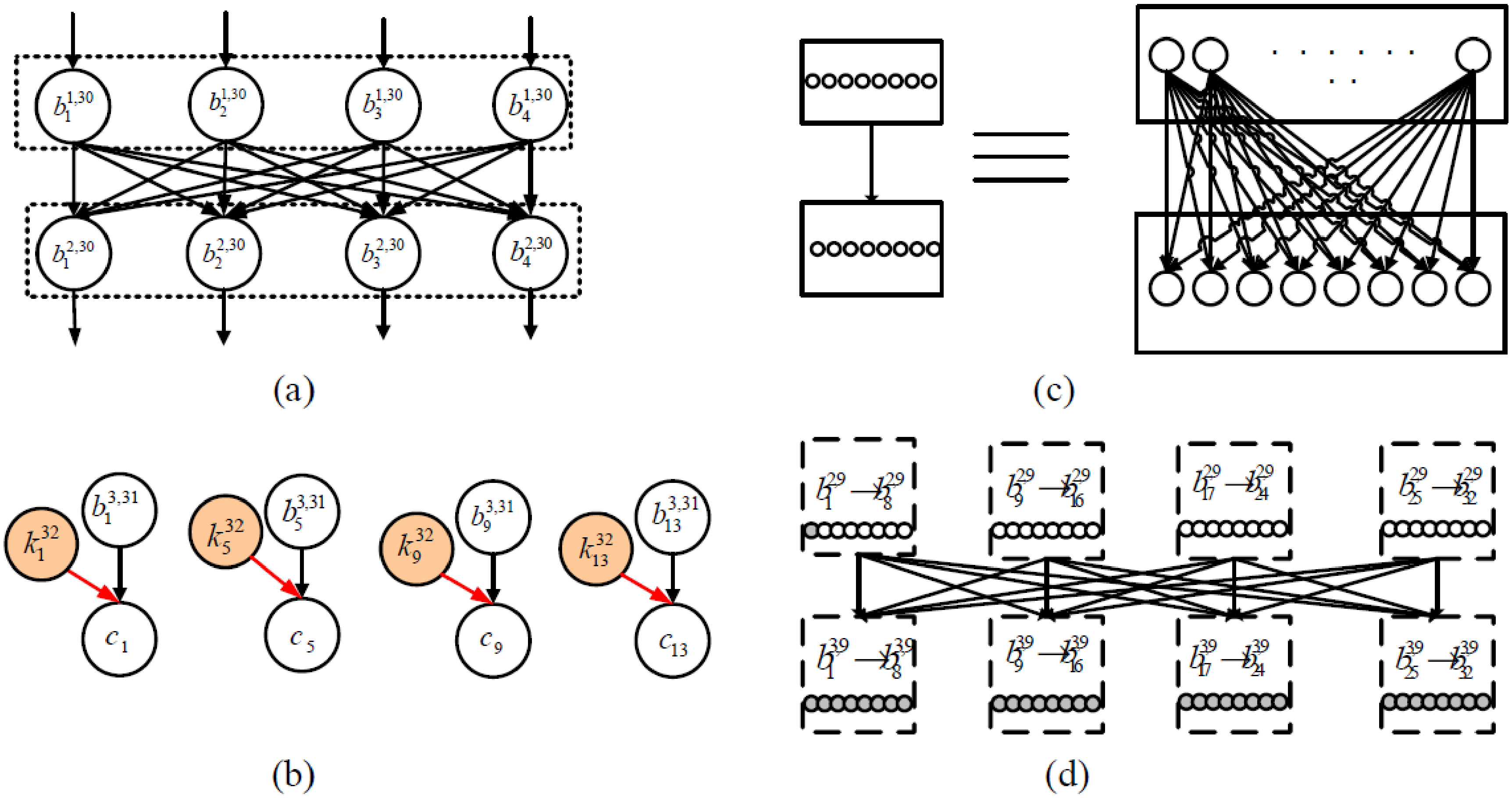
- Target: Evaluate a distinguisher
  - Identify the key bits to guess:
    - To evaluate each w_z^{ij}
    - To evaluate each variable set if exists.

A Graph Based Abstraction of the Cipher

- Cipher Dependency Graph (CDG)
  - Each node is a bit from any cipher state.
  - Directed links: Causal dependencies among bits

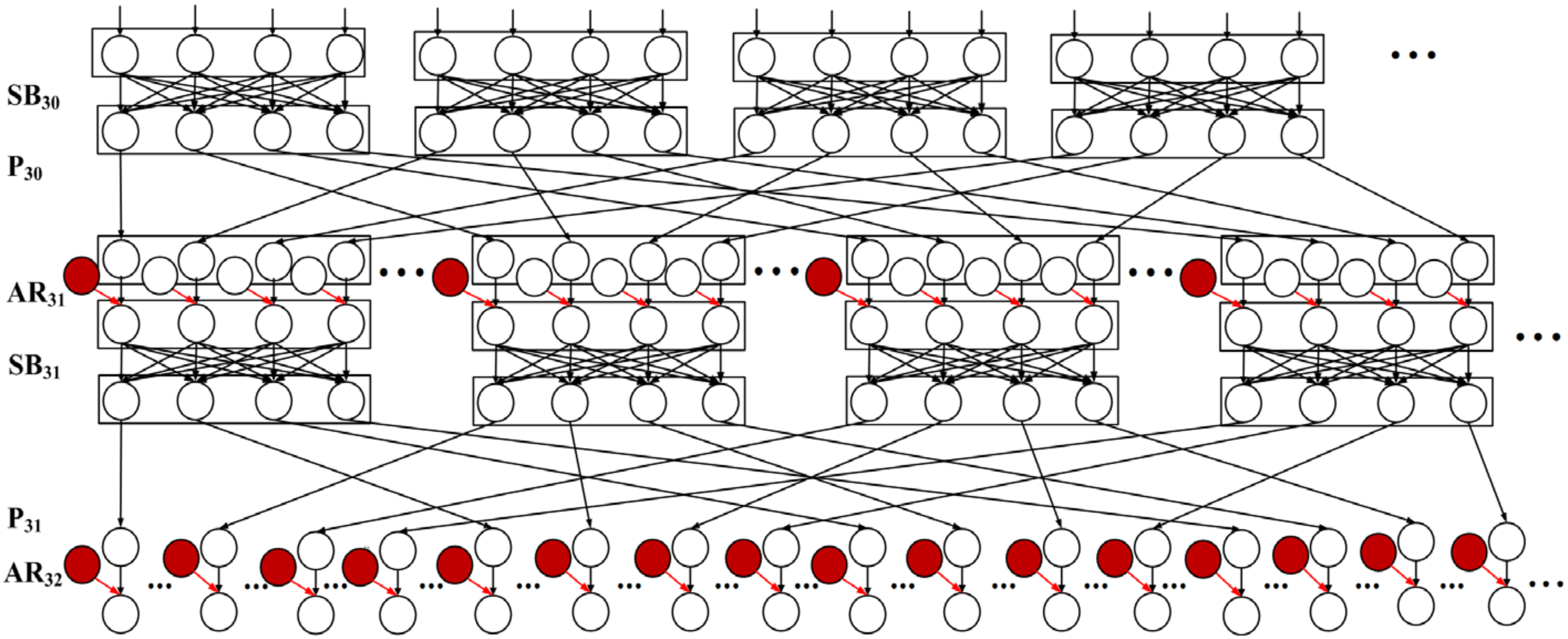
Proposed Approach (11/17)

CDG: Basic Blocks



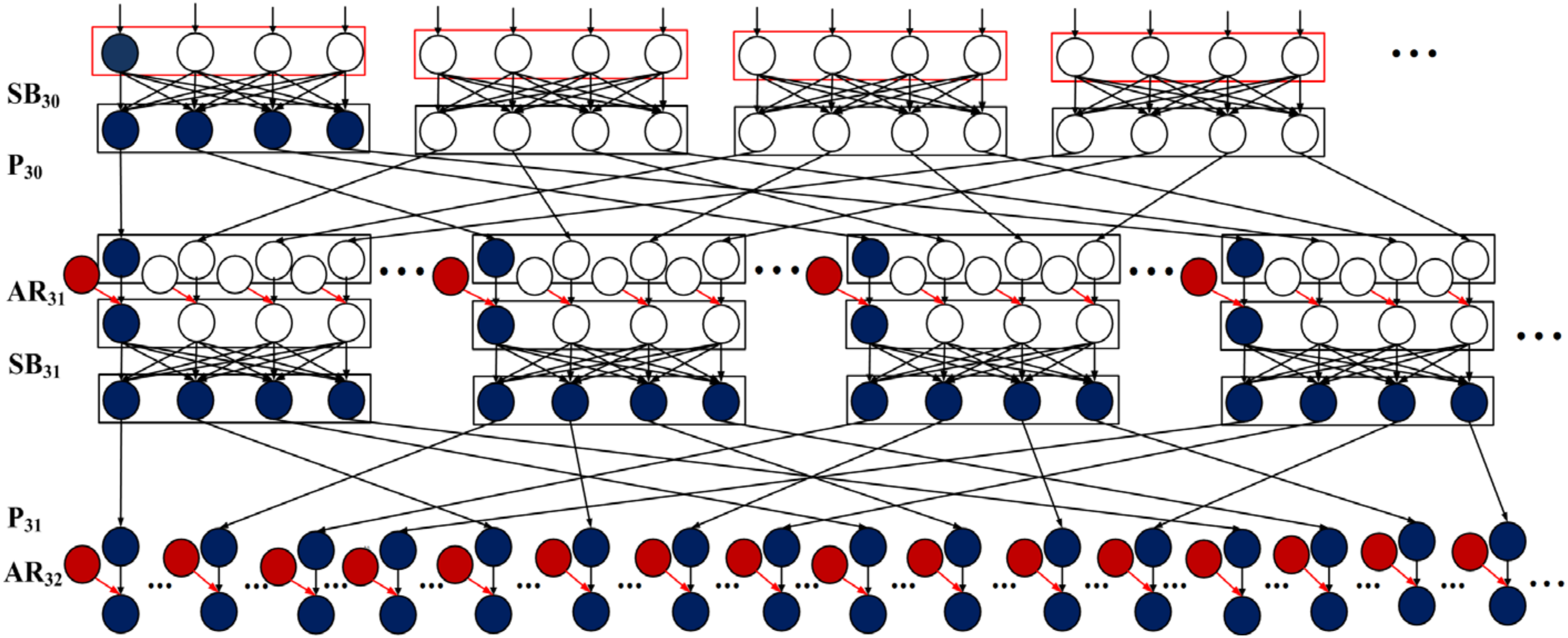
Proposed Approach (12/17)

CDG: An Example



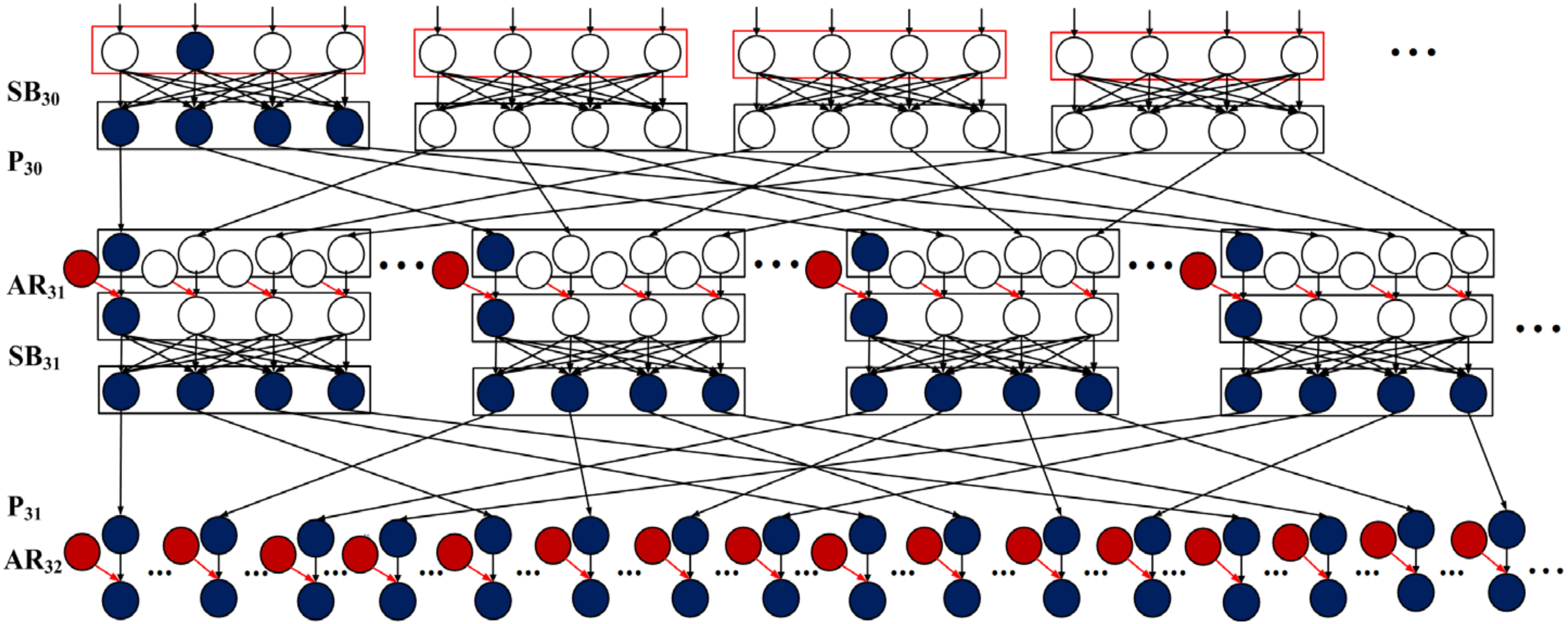
Proposed Approach (13/17)

CDG: How Does it Work?



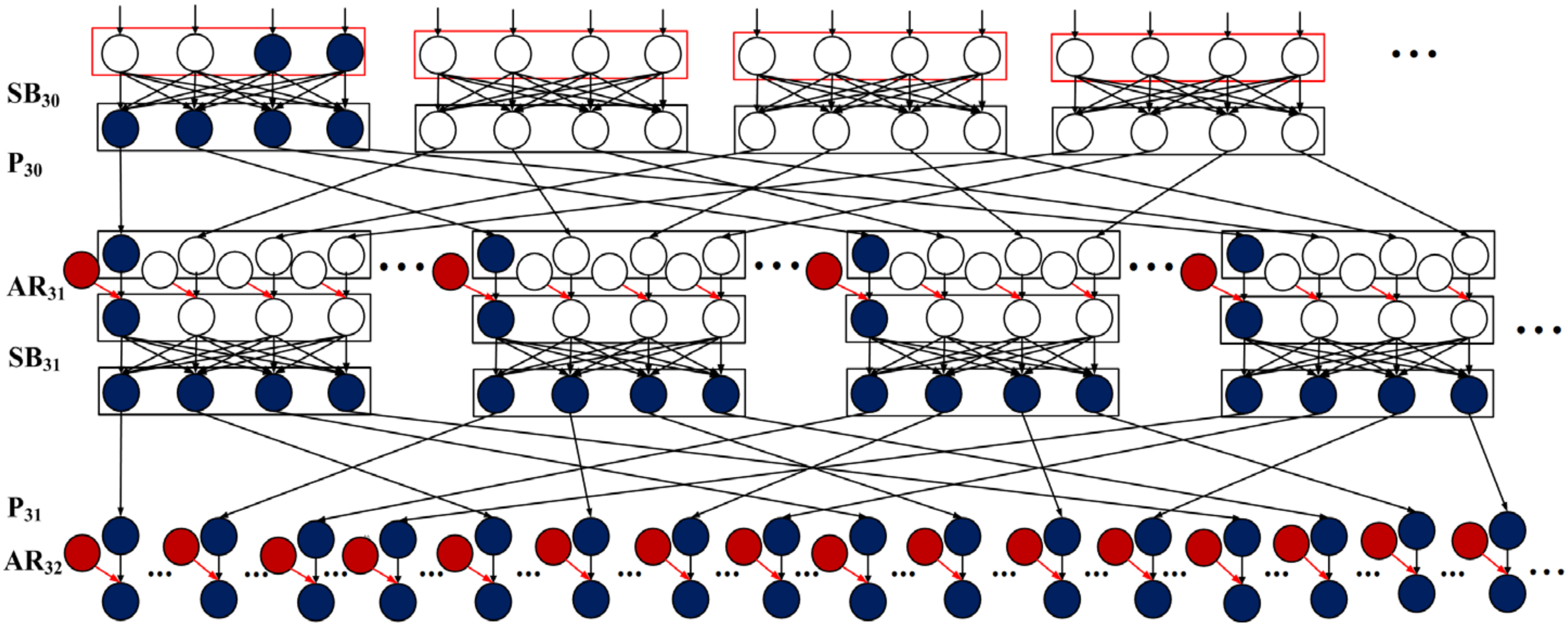
Proposed Approach (13/17)

CDG: How Does it Work?



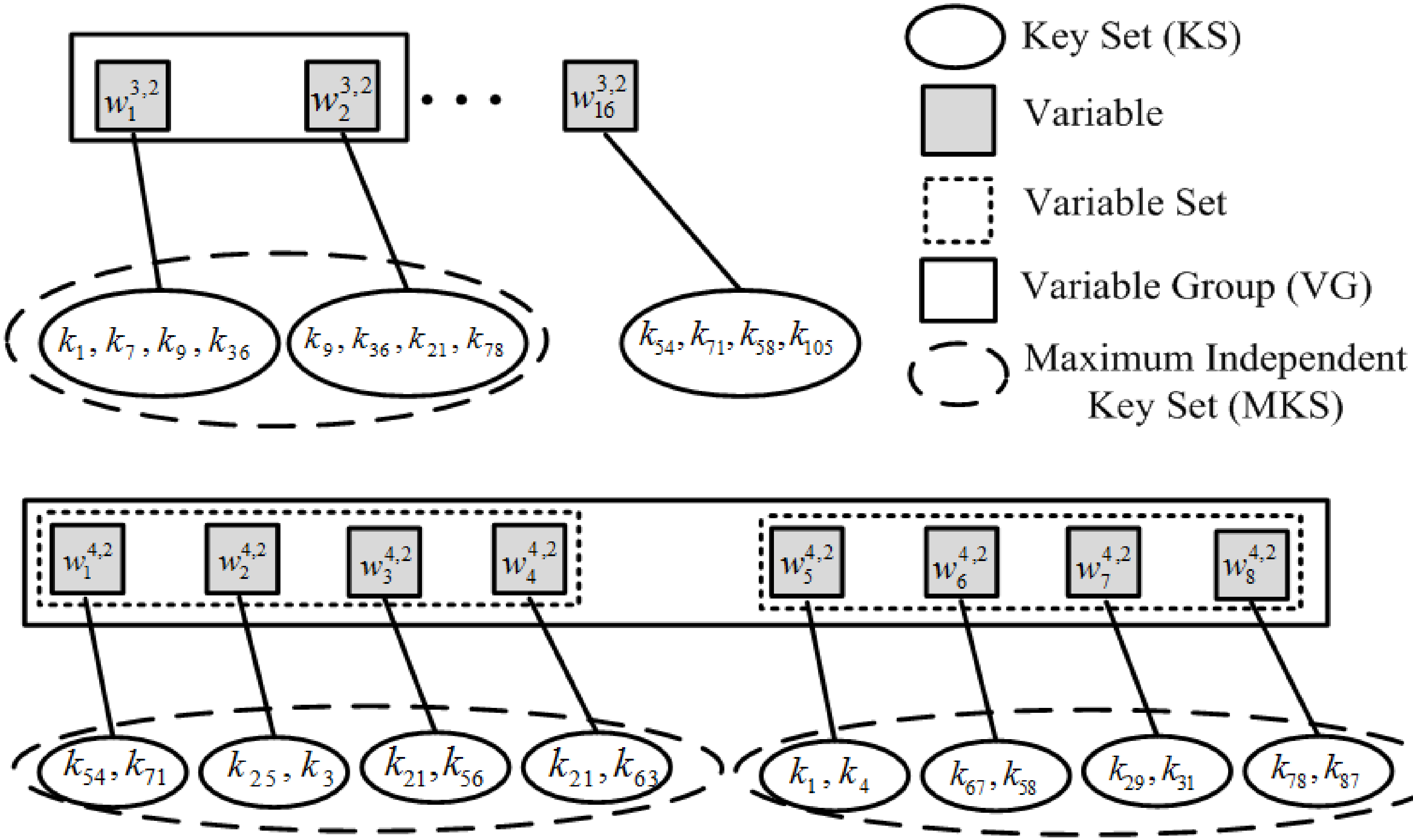
Proposed Approach (13/17)

CDG: How Does it Work?



Proposed Approach (14/17)

CDG: Maximum Independent Key Set and Variable Group



Proposed Approach (16/17)

CDG: Maximum Independent Key Set and Variable Group

- Each (MKS_h, VG_h) pair represents an independent subpart of the distinguisher evaluation.
- Each Subpart can be evaluated in parallel.

$$\mathbb{T}(h) = 2^{|MKS_h|}$$

$$max_h(\mathbb{T}(1), \mathbb{T}(2), ..., \mathbb{T}(M))$$

Proposed Approach (17/17)

Phase 3: Size of the Remaining Key Space

- Distinguisher evaluation returns a set of candidate keys.
- Searched exhaustively.
- Attack complexity:
  - max(distinguisher evaluation complexity, exhaustive search complexity)
- Calculated with ($MKS_{h,}$ VG_h) pairs

Proposed Approach (17/17)

Phase 3: Size of the Remaining Key Space

- Calculate the "probability of occurrence of the distinguishing property" with each VG_h

$$\mathscr{D}_j^i := \langle \{w_z^{ij}\}_{z=1}^{l}, \{Rng_{w_z^{ij}}\}_{z=1}^{l}, VS_{\delta_j^i}, \{IS_{\delta_j^i}^{v}\}_{v=1}^{|VS_{\delta_j^i}|} \rangle$$

for each VG_h

$$k_{size} := \texttt{BitCount}(MKS_h)$$

$$|\mathscr{R}|_{VG_h} := 2^{k_{size}} \times \mathbb{P}[VG_h]$$

$$|\mathscr{R}| := |\mathscr{R}| \times |\mathscr{R}|_{VG_h}$$

Calculated using these information

Case Study I

AES: Byte Fault at the Beginning of 8th Round

- 9 distinguishers identified
- First 4 rejected
  - Excessive evaluation cost
- Last 2 rejected
  - No nonlinear layer
- Distinguisher:
  - Output of 9th round MixColumn

$$\delta_9^4 = \langle w_1^{49}, w_2^{49}, \ldots, w_l^{49} \rangle$$

How?

- 4 Variable sets

$$(w_1^{49}, w_2^{49}, w_3^{49}, w_4^{49})$$
$$(w_5^{49}, w_6^{49}, w_7^{49}, w_8^{49})$$
$$(w_9^{49}, w_{10}^{49}, w_{11}^{49}, w_{12}^{49})$$
$$(w_{13}^{49}, w_{14}^{49}, w_{15}^{49}, w_{16}^{49})$$

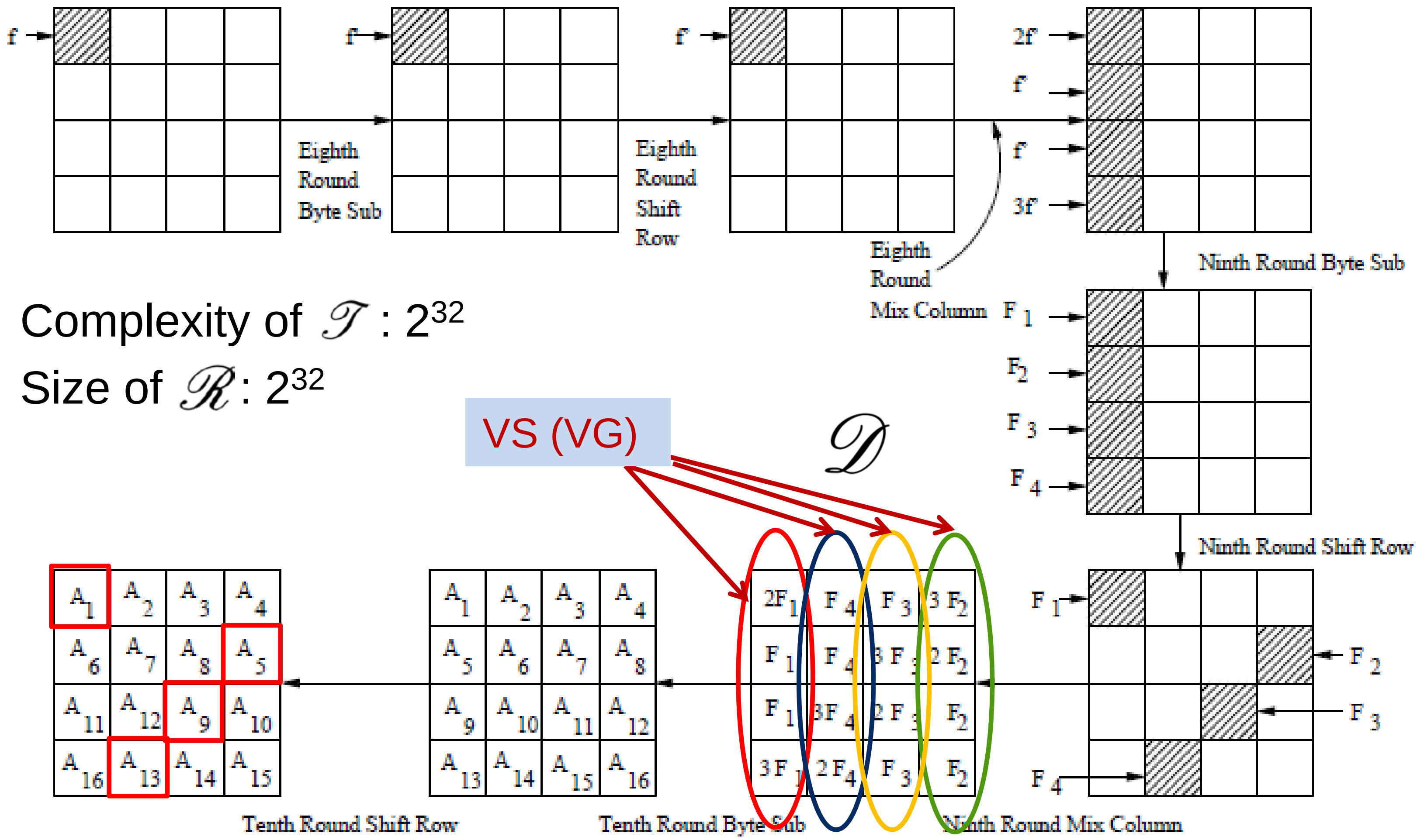
- Each have 255 itemsets
- Entropy of each variable set :
- State entropy:

$$\sum_{q=1}^{255} \frac{1}{255} \log_2(255) = 7.99$$

$$H_{Assn}(\delta_9^4) = 4 \times 7.99 = 31.96.$$

Case Study I (cont…)

- Complexity of $\mathscr{T}$: 2^{32}
- Size of $\mathscr{R}$: 2^{32}



Case Study I (continued..)

Distinguisher Evaluation

- Each VG contains a single variable set (VS)
- Each MKS contain 32 key bits.
- There are 4 (MKS, VG) pairs.
- Evaluation complexity: 2^{32}

Remaining Key Space

- 4 variable sets
  - Each with 255 itemsets
  - $\mathbb{P}[VG_h] = \frac{255}{2^{32}}$
- Remaining key space for each VS (aka. VG)
  - $2^{32} \times 2^{-24}$
- Total remaining key space
  - $(2^8)^4 = 2^{32}$

So, with a single fault the attack complexity reduces to 2^{32}

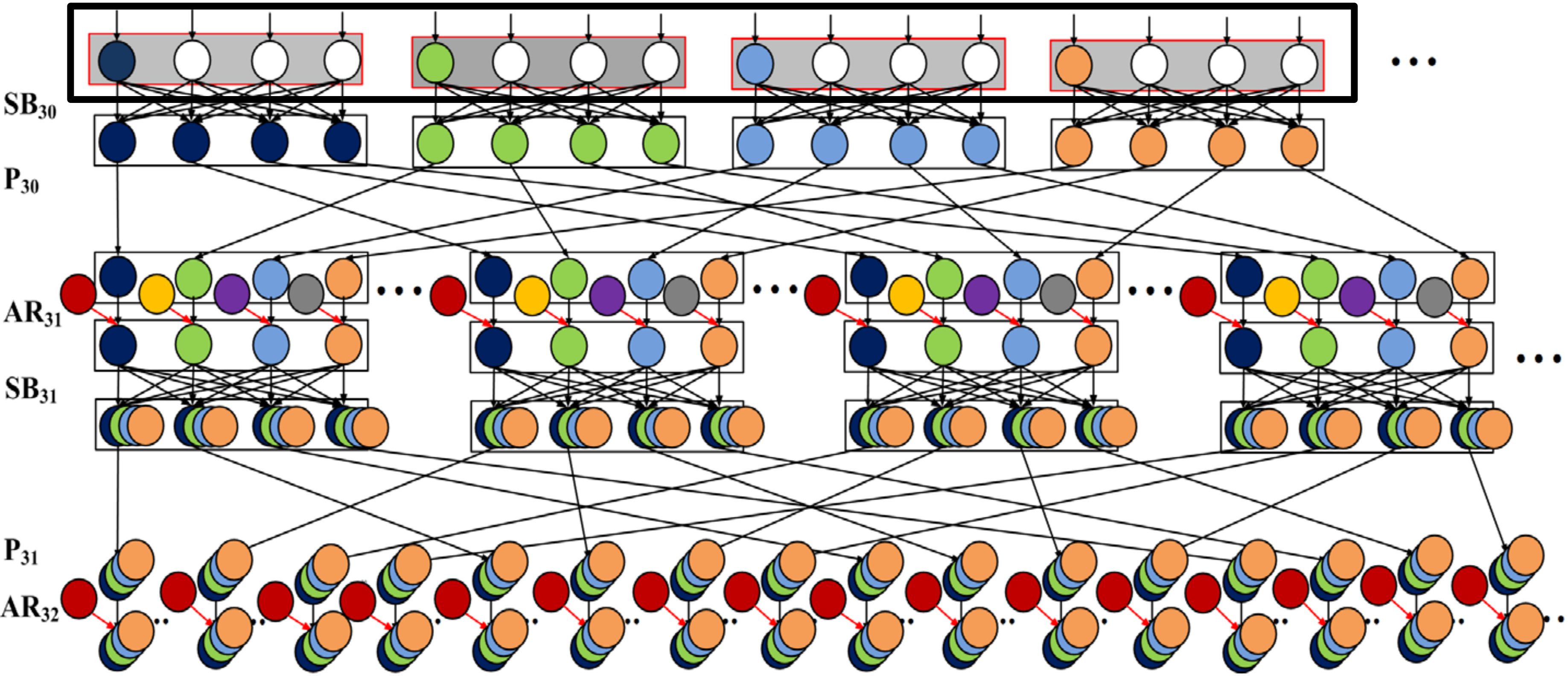
Case Study II

PRESENT: 2-Byte Fault at the Beginning of 28th Round

- Distinguisher at the 30th round
  - Each nibble can take only two values among 16
  - $H_{Ind}(\delta_{30}^{1}) = 16$

Case Study II (cont…)

CDG for PRESENT



$$\mathbb{P}[VG_h] = (2^{-3})^4$$

$$|MKS| = 32$$

Case Study II

PROOFS

PRESENT: 2-Byte Fault at the Beginning of 28^{th} Round

- Distinguisher at the 30^{th} round
  - Each nibble can take only two values among 16
  - $H_{Ind}(\delta_{30}^{1}) = 16$

- 4 (MKS, VG) pairs.
  - Each have 32 key bits
- Evaluation Complexity: 2^{32}
- Key space complexity: 2^{80}
- After another injection
  - $(2^{32} \times (2^{-12})^2)^4 = 2^{32}$

Conclusion

- Characterization of the exploitable fault space for a block cipher is a problem of immense practical value.
- Exploitable fault space characterization demands fast, generic and automated mechanism for the characterization of individual fault instances.
- A fast automation is proposed
  - Need not to do the attack; just calculate the complexity
  - Automatic distinguisher identification from fault simulation data.
  - Automated identification of divide-and-conquer strategy for key guess
  - Automated complexity evaluation.
- Future works:
  - Further generalization – Key schedule attacks, DFIA, MitM attacks
  - Automatic generation of attack equations.

Thank You

Questions?

Backup Slides

Proposed Approach (10/18)

Phase 1: Overall Flow

- Return:
  - Range of each w_z^{ij} : $\{Rng_{w_z^{ij}}\}_{z=1}^{l}$
  - Variable Set, Itemsets: $(VS_{\delta_j^i},\ \{IS_{\delta_j^i}^{v}\}_{v=1}^{|\bar{V}S_{\delta_j^i}|})$
  - State differential entropy: H_j^i

- End of Phase 1:
  - Gives us distinguishers

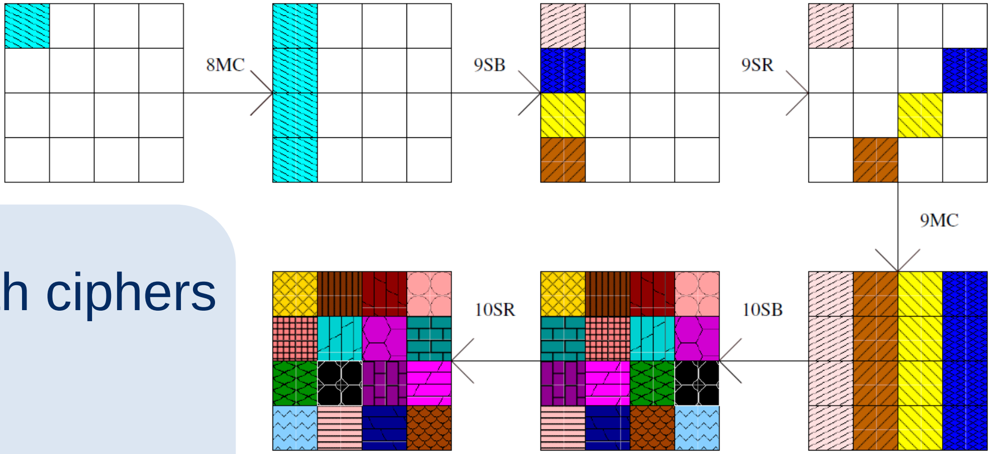
$$\mathscr{D}_j^i := \langle \{w_z^{ij}\}_{z=1}^{l},\ \{Rng_{w_z^{ij}}\}_{z=1}^{l},\ VS_{\delta_j^i},\ \{IS_{\delta_j^i}^{v}\}_{v=1}^{|VS_{\delta_j^i}|} \rangle$$

A Similar Suggestion: XFC

XFC Framework

- Characterizes the fault propagation path with colors
- Assumes simple fault equations
- Calculates the complexity using the equations and the colors

Punit Khanna, Chester Rebeiro, and Aritra Hazra. "XFC: A Framework for eXploitable Fault Characterization in Block Ciphers." *Proceedings of the 54th Annual Design Automation Conference 2017*., 2017.

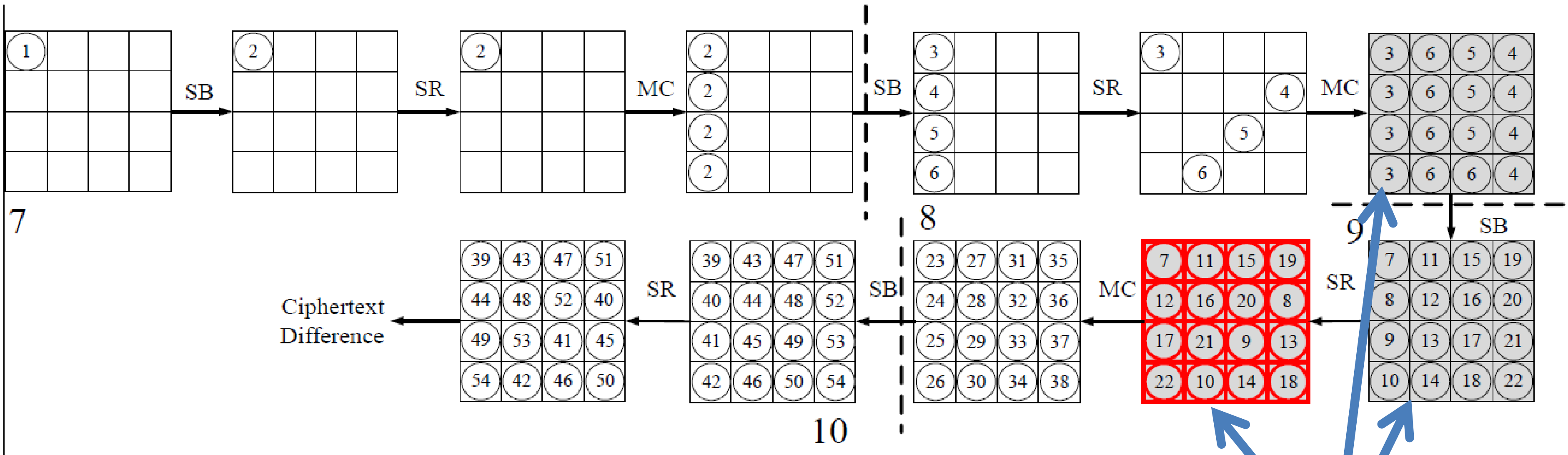


Issues

- Over simplistic – mainly works with ciphers having byte level structures
- Lacks proper automation

Issues with XFC

An Example – Impossible differential fault analysis on AES



All the bytes
are non zero

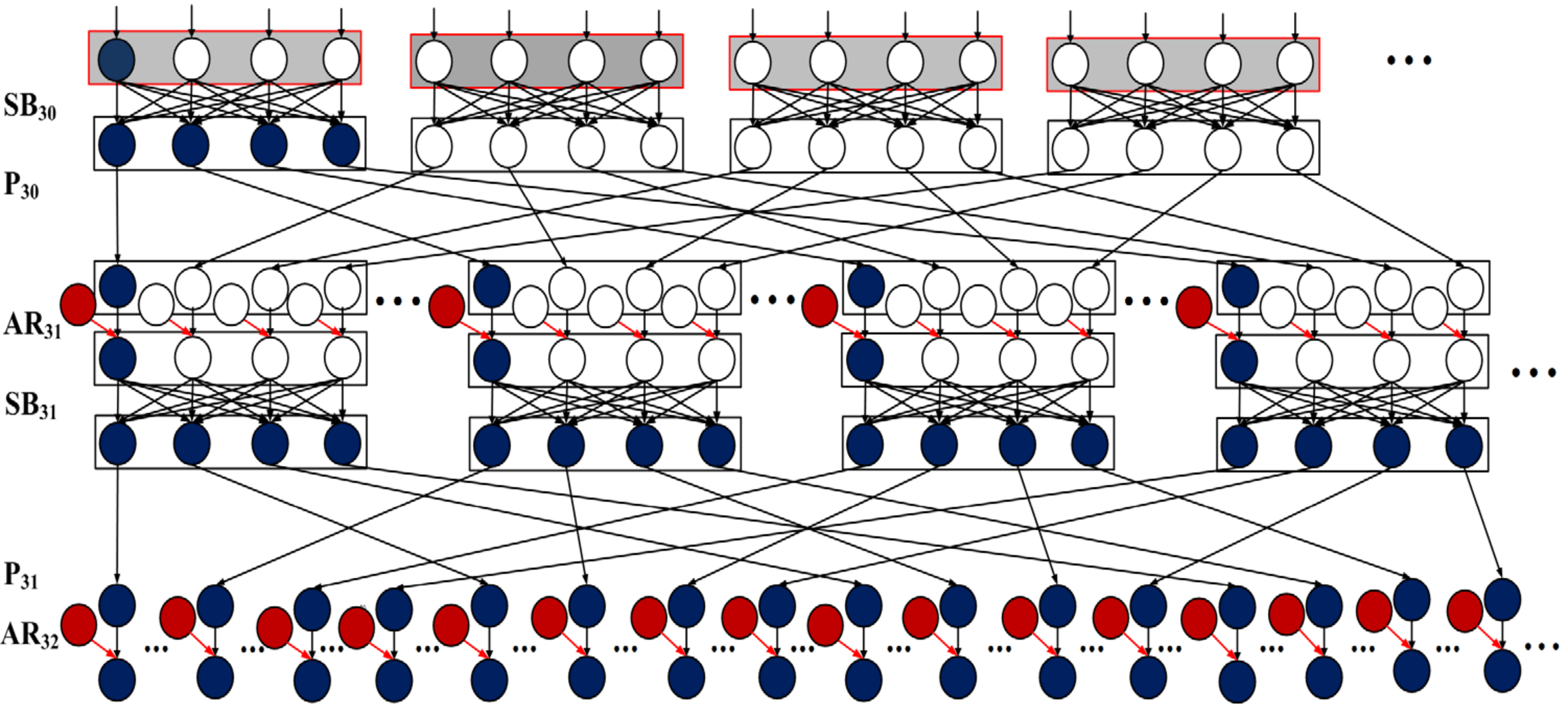
XFC assumes equations of the following form

$$\boxed{a\delta = S^{-1}(x_h \oplus k_h) \oplus S^{-1}(x'_h \oplus k_h)}$$

Fault relations can be more general

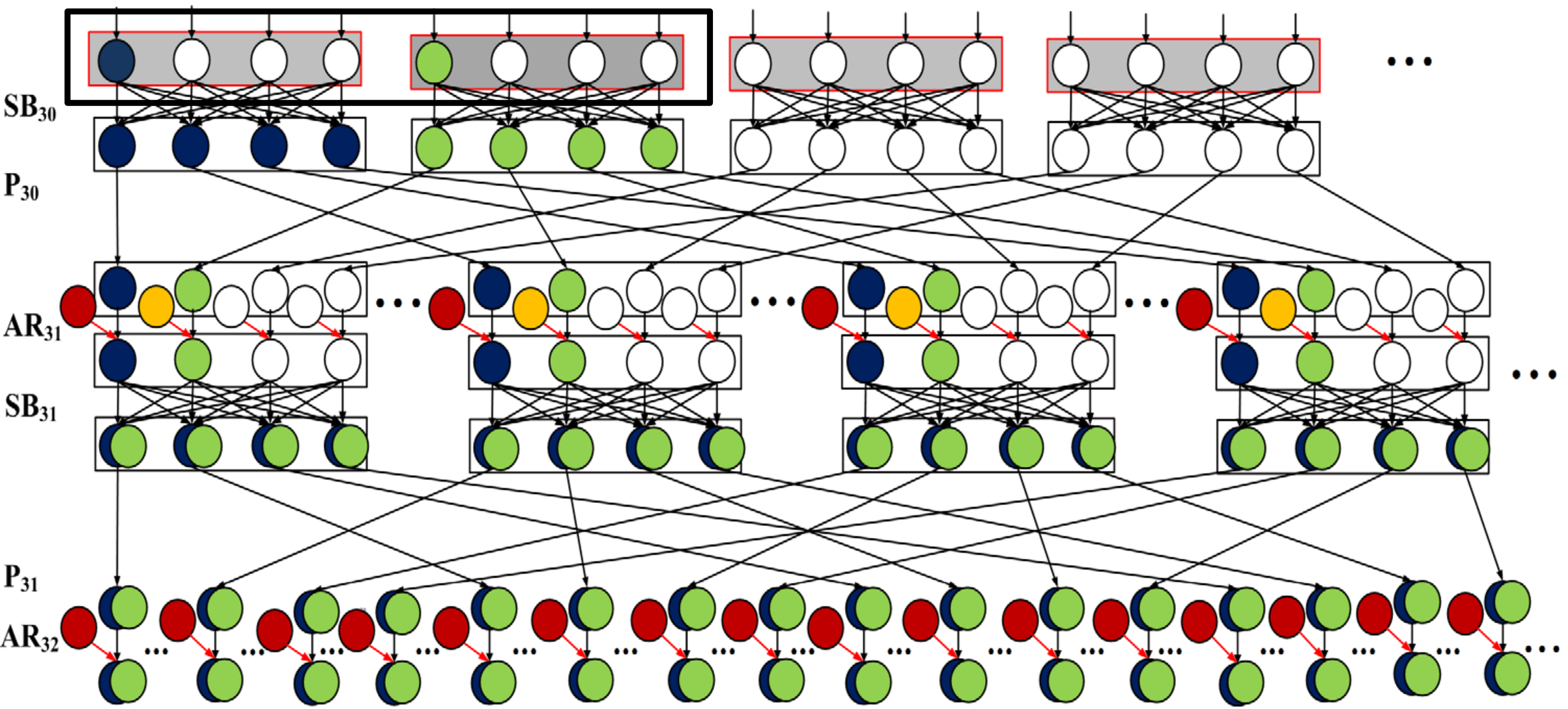
Proposed Approach (15/17)

CDG: Maximum Independent Key Set and Variable Group



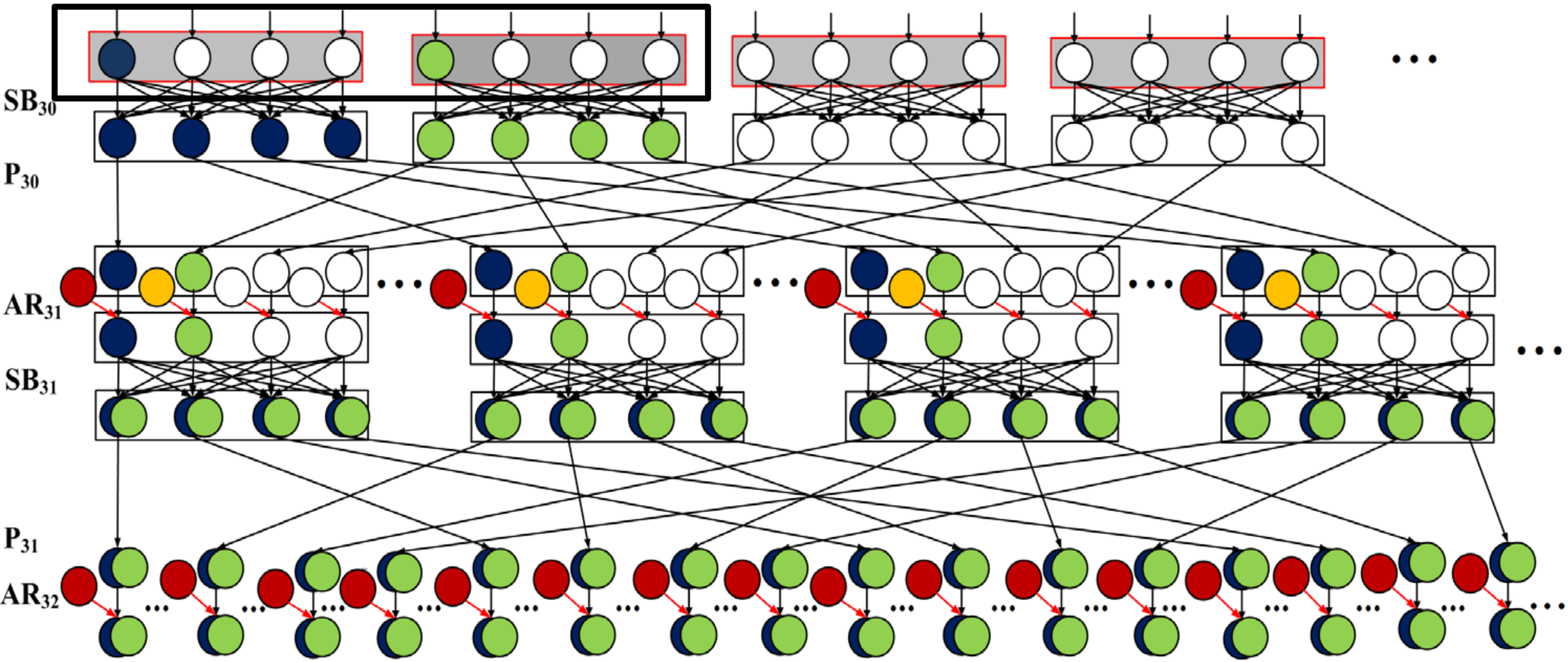
Proposed Approach (15/17)

CDG: Maximum Independent Key Set and Variable Group



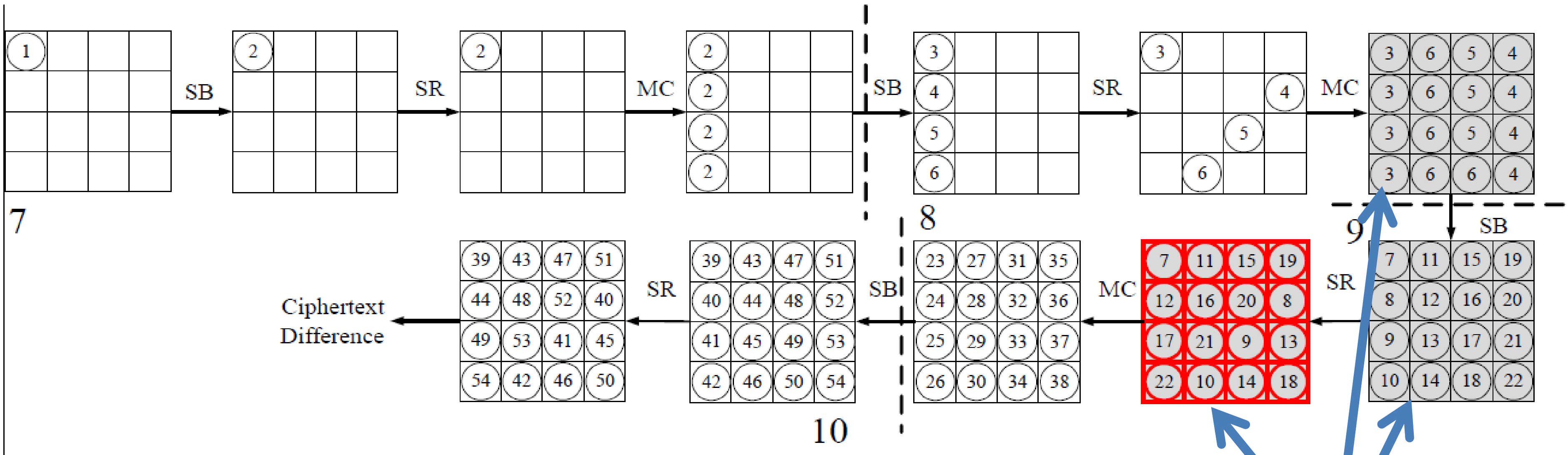
Proposed Approach (15/17)

CDG: Maximum Independent Key Set and Variable Group



Case Study II

An Example – Impossible differential fault analysis on AES



All the bytes are non zero

Case Study II

AES: Byte Fault at the Beginning of 7th Round

- Distinguisher:
  - The state before 9th round MixColumn.
- Impossible Differential Distinguisher:
  - Entropy: $H_{Ind}(\delta_9^4) = 127.90$

- Distinguisher evaluation
  - Complexity: 2^{32}
- Remaining key space:
  - $|\mathscr{R}|_{VG_1} = 2^{32} \times (\frac{255}{2^8})^4 = 2^{32} - 2^{26}$

Need multiple fault injections

Case Study II