Evaluating CNF/SMT Encodings for SAT-Based Differential Cryptanalysis of Lightweight Block Ciphers

Marzieh Vahid Dastjerdi[1,2], Majid Rahimi[2], Iman Mirzaali Mazandarani[2,3] and Sadegh Sadeghi[a,1]

[1] Department of Mathematics, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran

[2] Computer Science Group, Research Center for Development of Advanced Technologies, Tehran, Iran

[3] CPS2 Lab, Department of Communication, Faculty of Electrical Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

Abstract

This study evaluates three encoding methods for automated differential cryptanalysis: (1) SMT formulations (using `CVC`), (2) standard `CNF`, and (3) size-optimized `CNF` (via Logic Friday). We assess these using four SAT/SMT solver types: single-core (`CryptoMiniSat-v5`, `CaDiCaL`), multi-core (`Treengeling`), and massively parallel `Mallob`—novel to cryptanalysis. Encoding-solver combinations are tested on seven lightweight block ciphers representing distinct design philosophies: `SPECK-32` and `CHAM-64` (ARX structure), `SIMON-32` (AND-RX structure), `PRESENT`, `GIFT-128`, and `MIDORI-64` (4-bit S-box in SPN structure), and `LBLOCK` (Feistel structure). For each cipher, SAT/SMT instances targeting specific rounds/differential weights were generated, with wall-clock solving time, parallel efficiency, and modeling effort measured. Our results establish criteria for optimal encoding-solver pairings that strike a balance between modeling simplicity and computational performance. Crucially, `Mallob` emerges as the state-of-the-art framework for large-scale automated differential cryptanalysis.

Keywords: Automated Cryptanalysis, CNF, SMT, CVC, Differential Cryptanalysis, Lightweight Block Cipher

1 Introduction

Differential cryptanalysis [BS91] remains one of the most fundamental techniques for evaluating the security of symmetric block ciphers. It analyzes how differences in the input of a cipher affect the resulting differences at the output, aiming to identify high-probability propagation paths that can lead to practical attacks. Over the past two decades, the cryptographic community has invested significant effort in automating the search for such differential characteristics. Declarative frameworks such as Mixed-Integer Linear Programming (MILP) [MWGP11, MAA24], Boolean Satisfiability (SAT), Satisfiability Modulo Theories (SMT) [MP13, SWW21, SPGS20]. More recently, Constraint Programming (CP) [GMS16, HSE23, HGSE24] has emerged as a powerful tool for automatic trail

Emails: dastjerdi@rcdat.ac.ir (Marzieh Vahid Dastjerdi), rahimi@rcdat.ac.ir (Majid Rahimi), iman.mirzaali@sru.ac.ir (Iman Mirzaali Mazandarani), s.sadeghi@iasbs.ac.ir (Sadegh Sadeghi)

[a]Corresponding author.

Date of this document: 2026-11-12.

discovery. These techniques provide a generic modeling approach, eliminating the need for handcrafted search algorithms. However, each comes with unique trade-offs in terms of modeling complexity, runtime, and scalability.

While MILP-based tools remain widely used for small and mid-sized primitives due to their straightforward support for weight optimization, SAT and SMT-based approaches have demonstrated superior performance in bit-oriented designs such as `SPECK`, particularly when solving differential characteristics with tight probabilistic constraints [SWW21]. The work of Bellini *et al.* [BDPF+23] presents the most comprehensive cross-formalism comparison to date, covering SAT, SMT, MILP, and CP models across more than 20 cryptographic primitives. Their study provides valuable breadth, but it does not deeply investigate the implementation-level aspects of SAT/SMT approaches—such as encoding strategies, modeling compactness, or parallel scalability—which are critical for practical large-scale cryptanalysis.

Contributions

As SAT/SMT has emerged as the most effective available formalism for automated bit-oriented cryptanalysis, it becomes essential to dissect this approach in detail and analyze its various dimensions, including encoding strategies, solver backends, and scalability aspects. This paper complements and extends prior work by narrowing the focus to SAT/SMT-based differential cryptanalysis and providing a detailed, experimental study at the encoding and solver level. Our contributions are threefold:

- **Encoding-level evaluation.** Unlike [BDPF+23], which primarily compared different formalisms, we systematically evaluate three concrete encoding strategies—Satisfiability Modulo Theories (`SMT-LIB`) in the `CVC` (Cooperating Validity Checker) dialect using bit-vector theories, standard Conjunctive Normal Form (`CNF`) in DIMACS format, and size-minimised `CNF` generated by Logic Friday ($\texttt{CNF}_{\texttt{LF}}$, derived from SMT-level encoding)—under identical high-level models. This reveals how modeling choices translate into structural differences in the resulting `CNF`s and directly affect solver runtime.
- **Solver-level evaluation with parallel scalability.** We assess the performance of these encodings across a diverse solver suite that includes single-core solvers (`CryptoMiniSat-v5` [SNC09], `CaDiCaL` [BFF+24]), a multi-core portfolio solver (`Treengeling` [Bie17]), and the massively parallel `Mallob` platform [SS24]. Our experiments quantify the impact of solver design (portfolio vs. decomposition vs. single-core) and demonstrate significant scalability gains, with `Mallob` achieving an average $\approx 2.9\times$ speedup over the best single-core baselines.
- **Practical guidelines.** By mapping cipher structures (ARX, SPN, `SIMON-like`, Feistel) to the most efficient encoding–solver combinations, we extract concrete lessons that serve as a practical guide for future SAT/SMT-based cryptanalysis. This implementation-level perspective provides actionable insights for researchers and designers.

Figure 1 illustrates the overall pipeline, including modeling, encoding evaluation, solver assessment, and results. All modeling scripts, encoding generators, and benchmark configurations are publicly available at: https://github.com/sat-diff-analysis/differential-sat-encoding.git

Outline

The remainder of this paper is organised as follows. Section 2 reviews prior work in automatic differential cryptanalysis, focusing in particular on comparative evaluations of

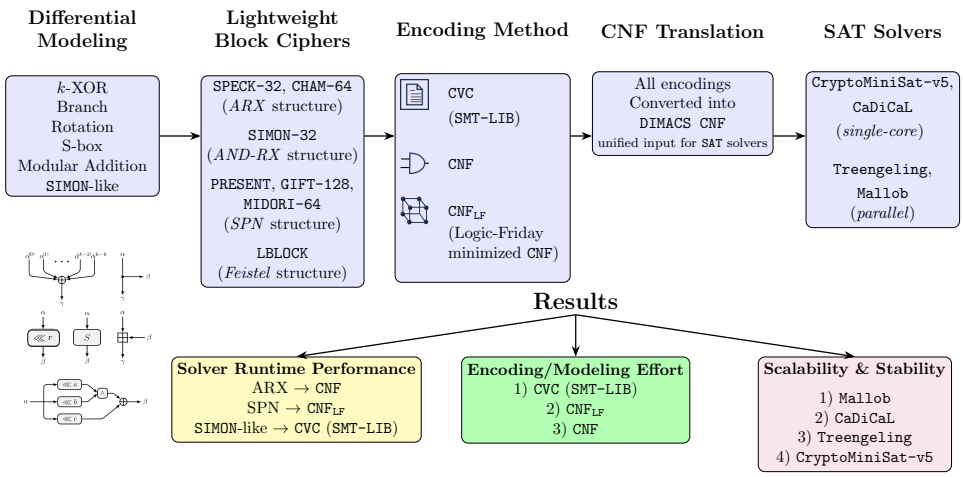

Figure 1: Experimental pipeline for encoding and solver evaluation.

SAT- and SMT-based approaches. In Section 3, we detail the high-level modelling strategy and describe three families of propositional encodings: `SMT-LIB` (`CVC` dialect), conventional `CNF`, and size-minimised `CNF` produced by Logic Friday ($\texttt{CNF}_{\texttt{LF}}$). Section 4 presents comprehensive experimental results across four solver platforms and seven lightweight block ciphers, comparing performance, parallel scalability, and modelling effort. Finally, Section 5 summarises our key findings and outlines directions for future research.

2 Related Work

Sahu *et al.* [SPGS20] develop an SMT-based cryptanalysis framework demonstrated on the `IDEA` cipher. By encoding Boolean representations of the cipher into `SMT-LIB`, they enable key recovery from a few plaintext-ciphertext pairs. Among several evaluated solvers, Boolector performs best, especially after custom optimization for modular multiplication.

Delaune *et al.* [DDH+20] compare MILP, SAT, CP, and ad-hoc tools for finding differential characteristics in `SKINNY`. While ad-hoc methods are most effective for truncated trails, CP models outperform others in probability maximization due to their efficient handling of constraints. Their results improve known bounds for up to 14 rounds.

Sun *et al.* [SWW21] enhance SAT-based differential and linear trail search by encoding Matsui's bounding conditions directly into SAT without dummy variables. This significantly reduces problem size and improves efficiency, yielding optimal trails for multiple ciphers.

Gundaram *et al.* [GTM21] assess SMT solver performance in key-recovery attacks on `SIMON`. Although SAT solvers are better for Boolean logic, SMT solvers, particularly Boolector, handle ARX-style constructions more effectively due to their support for word-level operations.

Sun and Wang [SW23] study modeling strategies for large S-boxes in SAT-based differential cryptanalysis. They introduce logic minimization techniques that simplify 8-bit S-box encodings, allowing tighter bounds and improved efficiency in analyzing `SKINNY-128` and `AES`.

Bellini *et al.* [BDPF+23] provide the most comprehensive comparison to date of four declarative frameworks—SAT, SMT, MILP, and CP—for automatic differential characteristic search in over 20 cryptographic primitives. Their study encompasses three primary tasks: identifying the optimal differential trail, enumerating all optimal trails, and estimating differential probabilities. Employing a diverse suite of 16 solvers across these frameworks, they analyze performance on both easy and challenging instances. Results

demonstrate that SAT solvers generally outperform others for ARX and SPN ciphers, with `Kissat` leading in SAT, `Yices2` in SMT, `Gurobi` in MILP, and `Chuffed` in CP.

3 Modeling Primitives

Traditional cryptanalysis has relied on expert intuition and manual inspection, often resulting in limited scalability and inconsistent reproducibility, particularly when analyzing ciphers with many rounds. In contrast, automated approaches based on logical reasoning and constraint solving have emerged as powerful alternatives, offering a principled and systematic framework for cryptographic evaluation.

These methods encode cipher components and difference propagation rules into logical formulas that can be processed by SAT and SMT solvers. In SAT-based models, constraints are typically transformed into propositional logic expressed in Conjunctive Normal Form (`CNF`). SMT-based approaches support richer representations using theories such as bit-vectors, modular arithmetic, and uninterpreted functions. Encodings in `SMT-LIB` or `CVC` formats are commonly used by SMT solvers to express operations such as XOR, modular addition, rotations, and S-box transformations more compactly.

Tools like `STP` act as a bridge between high-level SMT encodings and SAT solvers by translating bit-vector constraints into `CNF`. This facilitates the use of optimized SAT engines in analyzing algebraic structures of block ciphers. SAT solvers such as `CryptoMiniSAT v5` and `CaDiCaL` are widely adopted for their high single-core efficiency and built-in support for features like native XOR clauses and incremental solving. SMT solvers such as `Z3` and `CVC5` [BFH+20] offer enhanced expressiveness at the expense of performance, making them suitable for modeling complex cipher components, like modular addition, in ARX designs.

In multicore settings, parallel solvers like `Treengeling`, which employ portfolio-based strategies, and distributed systems such as `Mallob`, which can scale across thousands of workers using message-passing architectures, facilitate the scalable exploration of large search spaces. These solver backends, combined with expressive encoding strategies, form the computational backbone of modern automated differential and linear cryptanalysis.

In this section, the exclusive OR (XOR) is denoted by $\oplus$. Logical OR and AND are represented by $\vee$ and $\wedge$, respectively. Moreover, $\overline{x}$ denotes the negation of the Boolean variable x. For rotation amount $r > 0$, left and right bitwise rotations are expressed as $\lll r$ and $\ggg r$, respectively.

3.1 Logical Encodings of Cryptographic Operations

In this paper, we apply three different logical encodings of `CNF` expressions to model cryptographic primitives under differential propagation. Each encoding has distinct characteristics and is suited to specific use cases.

1. `CVC` (High-Level SMT Approach)

The `CVC` format is widely used in SMT solvers such as `CVC5` and Boolector. It supports rich data types including bit-vectors, arrays, and integer arithmetic, making it suitable for expressing cryptographic operations directly in a high-level, human-readable form. For instance, an n-bit XOR operation ($\gamma = \alpha \oplus \beta$) can be encoded as follows:

```
"ASSERT(gamma=BVXOR(alpha,beta))";
```

This encoding abstracts away the underlying `CNF` representation, allowing for rapid prototyping. Integrating with tools like CryptoSMT is beneficial for automating differential and linear cryptanalysis.

2. DIMACS `CNF` (Low-Level SAT Format)

DIMACS `CNF` is the standard format used by SAT solvers, where variables are represented by positive integers (e.g., 1, 2, 3,...), and negation is denoted by a minus sign (e.g., -1). A DIMACS file starts with a header line of the form:

```
p cnf <num_vars> <num_clauses>
```

This specifies the number of variables and clauses. Each subsequent line represents a clause consisting of space-separated literals, ending with a zero to indicate termination. For example, the XOR operation can be modeled using four clauses per bit position:

```
p cnf 3 4
1 2 -3 0
1 -2 3 0
-1 2 3 0
-1 -2 -3 0
```

3. $\texttt{CNF}_{\texttt{LF}}$ (Hybrid Logic Friday-Inspired Encoding)

The $\texttt{CNF}_{\texttt{LF}}$ method is inspired by the output of the logic minimization tool `Logic Friday`. This approach translates each clause extracted from Logic Friday into individual `ASSERT` statements in `SMT-LIB` format, making it compatible with solvers such as `Z3` or Boolector.

This method enables seamless integration of `CNF`-based logical constraints into automated reasoning frameworks, facilitating tasks such as differential cryptanalysis. An example of generating bitwise XOR constraints ($\gamma = \alpha \oplus \beta$) programmatically is shown below:

```
for i in range(n):
    "ASSERT((({0}[{3}:{3}])|({1}[{3}:{3}])|(~{2}[{3}:{3}]))=0bin1);".format(alpha, beta
        , gamma, i)
    "ASSERT((({0}[{3}:{3}])|(~{1}[{3}:{3}])|({2}[{3}:{3}]))=0bin1);".format(alpha, beta
        , gamma, i)
    "ASSERT(((~{0}[{3}:{3}])|({1}[{3}:{3}])|({2}[{3}:{3}]))=0bin1);".format(alpha, beta
        , gamma, i)
    "ASSERT(((~{0}[{3}:{3}])|(~{1}[{3}:{3}])|(~{2}[{3}:{3}]))=0bin1);".format(alpha,
        beta, gamma, i)
```

Each line corresponds to one clause from the `CNF` representation of XOR. These clauses enforce the deterministic behavior of XOR under differential propagation: $\gamma_i = \alpha_i \oplus \beta_i$.

3.2 Modeling Differential Propagation of Operations

This section presents the logical models for linear and nonlinear cryptographic operations under differential propagation. We use x_i to denote the i-th bit of an n-bit vector x, indexed from 0 (MSB).

k-XOR Operation

The XOR of $k+1$ binary variables refers to an *k-XOR* operation. Our approach to modeling k-XOR avoids the use of dummy variables. This modeling employs integer programming constraints derived from the convex hull of XOR outcomes, as applied in [IS24].

For a k-XOR operation applied bitwise to n-bit words, let $\alpha^{(0)}, \dots \alpha^{(k)}$ be input differences γ be the output difference. A differential $(\alpha^{(0)}, \dots, \alpha^{(k)}, \gamma)$ is valid if and only if the following `CNF` formula is satisfied for all bit positions $0 \leq i < n$:

$$\bigwedge_{S \subseteq \{0,\dots,k\}:|S| \text{ is odd}} \left((\bigvee_{j \in S} \overline{\alpha_i^{(j)}}) \vee (\bigvee_{j \notin S} \alpha_i^{(j)}) \vee \gamma_i \right) \wedge$$

$$\bigwedge_{S \subseteq \{0,\dots,k\}:|S| \text{ is even}} \left((\bigvee_{j \in S} \overline{\alpha_i^{(j)}}) \vee (\bigvee_{j \notin S} \alpha_i^{(j)}) \vee \overline{\gamma_i} \right)$$

In particular, the 2-XOR modeling approach is applied to the `SIMON` algorithm and the matrix multiplication layer of the `MIDORI-64` algorithm.

Branching Operation

For an n-bit branching operation, let α denote the input difference and β, γ represent the output differences on each branch. A differential (α, β, γ) is valid if and only if the following `CNF` formula is satisfied for all bit positions $0 \leq i < n$:

$$(\overline{\alpha}_i \vee \beta_i) \wedge (\alpha_i \vee \overline{\beta}_i) \wedge (\overline{\alpha}_i \vee \gamma_i) \wedge (\alpha_i \vee \overline{\gamma}_i).$$

This enforces the deterministic behavior of the branching operation: $\alpha_i = \beta_i = \gamma_i$.

Rotation Operation

For an n-bit rotation operation, let α denote the input difference and β represent the output difference after rotating by r bits. A differential (α, β) is valid if and only if the following `CNF` constraints are satisfied for all bit positions $0 \leq i < n$:
- Left rotation by r:

$$(\overline{\alpha}_{(i-r) \bmod n} \vee \beta_i) \wedge (\alpha_{(i-r) \bmod n} \vee \overline{\beta}_i)$$

- Right rotation by r:

$$(\overline{\alpha}_{(i+r) \bmod n} \vee \beta_i) \wedge (\alpha_{(i+r) \bmod n} \vee \overline{\beta}_i)$$

S-box Layer

We first generate the *Difference Distribution Table* (DDT) of the S-box. The DDT of an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a $2^n \times 2^n$ matrix where each entry $\mathrm{DDT}(\alpha, \beta)$ for $\alpha \in \mathbb{F}_2^n$ (input difference) and $\beta \in \mathbb{F}_2^n$ (output difference) is defined as:

$$\mathrm{DDT}(\alpha, \beta) = \left|\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus \alpha) = \beta\}\right|$$

This denotes the number of inputs x for which an input difference α results in an output difference β after application of the S-box S. From the DDT, two main representations are extracted:

- $\star$-DDT: A binary representation indicating whether a differential transition (α, β) is possible or impossible [AST+17].

- w-DDT: A weighted version where each entry represents either the number of occurrences or the probability of a given differential transition [BDPF+23].

Let F be a flag indicating whether the transition is valid. For $\star$-DDT-based modeling, the format of the truth table is (α, β, w, F), where $w = 0$ if and only if it corresponds to impossible or deterministic transitions. For w-DDT-based modeling, the format extends to $(\alpha, \beta, p_1, \ldots, p_k, F)$, where $p_1, \ldots, p_k$ are dummy Boolean variables used to encode the differential weight $w = -\log_2(p)$, where p denotes the probability of a given differential (α, β). Let P contain all probabilities of valid differentials. The weight w is decomposed into two components [SW23]:
• An integer part, encoded using μ Boolean variables $u = (u_0, ..., u_{\mu-1})$, where $\mu = \max_{p \in P} (\lfloor -\log_2(p) \rfloor)$,
• A fractional part, represented by k Boolean variables $v = (v_0, ..., v_{k-1})$, corresponding to the following set of size k:

$$D = \{\lceil -\log_2(p) \rceil - (-\log_2(p)) \mid -\log_2(p) \notin \mathbb{Z}, p \in P\}$$

If $\alpha \to \beta$ is a possible differential propagation with a probability of p, then the corresponding assignments to $\alpha \parallel \beta \parallel u \parallel v$ must satisfy the equation:

$$\sum_{i=0}^{\mu-1} u_i + \sum_{i=0}^{k-1} d_i \cdot v_i = -\log_2(p),$$

where d_i is the i-th member of D. The resulting truth tables are then fed into `LogicFriday`, a logic minimization tool that automatically translates them into (`CNF`) expressions.

Modular Addition

For an n-bit modular addition operation $z = x + y \mod 2^n$, let α and β denote the input differences and γ represent the output difference. A differential (α, β, γ) is valid if it satisfies the differential constraints for modular addition defined in [LM01], including:

$$\alpha_{n-1} \oplus \beta_{n-1} = \gamma_{n-1}$$

and for $0 \leq i < n-1$,

$$\begin{aligned}
&(\alpha_i \vee \beta_i \vee \overline{\gamma_i} \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1}) \\
&\wedge (\alpha_i \vee \overline{\beta_i} \vee \gamma_i \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1}) \\
&\wedge (\overline{\alpha_i} \vee \beta_i \vee \gamma_i \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1}) \\
&\wedge (\overline{\alpha_i} \vee \overline{\beta_i} \vee \overline{\gamma_i} \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1}) \\
&\wedge (\alpha_i \vee \beta_i \vee \gamma_i \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}}) \\
&\wedge (\alpha_i \vee \overline{\beta_i} \vee \overline{\gamma_i} \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}}) \\
&\wedge (\overline{\alpha_i} \vee \beta_i \vee \overline{\gamma_i} \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}})
\end{aligned}$$

The logarithmic weight w of a differential is computed using dummy variables $w_0, w_1, \ldots, w_{n-2}$, where:

$$w = \sum_{i=0}^{n-2} w_i$$

with $w_i = 1$ if and only if α_i, β_i, and γ_i are not all equal. For $0 \leq i \leq n-2$, each w_i must satisfy:

$$\begin{aligned}
&(\alpha_{i+1} \vee \gamma_{i+1} \vee w_i) \wedge (\beta_{i+1} \vee \gamma_{i+1} \vee w_i) \\
&\wedge (\alpha_{i+1} \vee \beta_{i+1} \vee w_i) \wedge (\alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} \vee w_i) \\
&\wedge (\alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} \vee w_i) \wedge (\alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} \vee w_i)
\end{aligned}$$

`SIMON`-like Round Function

The round function of `SIMON` is defined over n-bit words and takes the form

$$f(x) = ((x \lll a) \wedge (x \lll b)) \oplus (x \lll c),$$

where $a > b$, n is even, and $gcd(n, a-b) = 1$. We apply the differential model from [KLT15], which is adapted for automated analysis using SAT/SMT solvers [SWW21].

Let α and β denote the input and output differences, respectively. To capture the propagation patterns through the AND operation, three dummy n-bit variables z, *varbits*, *doublebits* are used. When α is not the all-ones vector (a special case requiring separate analysis), the differential transition (α, β) is considered valid if and only if all of the

following CNF constraints are satisfied across every bit position $0 \leq i \leq n-1$ (with all indices computed modulo n):

$$
\begin{aligned}
&(\overline{\alpha_{i+a}} \vee varibits_i) \wedge (\overline{\alpha_{i+b}} \vee varibits_i) \\
&\wedge (\alpha_{i+a} \vee \alpha_{i+b} \vee \overline{varibits_i}) \wedge (varibits_i \vee \overline{z_i}) \\
&\wedge (\overline{\alpha_{i+a}} \vee \overline{\alpha_{i+b}} \vee \overline{doublebits_i}) \\
&\wedge (\overline{\alpha_{i+b}} \vee \alpha_{i+2a-b} \vee \overline{doublebits_i}) \\
&\wedge (\alpha_{i+a} \vee \overline{\alpha_{i+b}} \vee \overline{\alpha_{i+2a-b}} \vee doublebits_i) \\
&\wedge (\alpha_{i+b} \vee \overline{doublebits_i}) \wedge (\overline{doublebits_i} \vee z_i \vee \overline{z_{i+a-b}}) \\
&\wedge (\overline{doublebits_i} \vee \overline{z_i} \vee z_{i+a-b}).
\end{aligned}
$$

Furthermore, the necessary 2-XOR differential constraints for $\alpha_{i+c} \oplus \beta_i \oplus z_i$.

3.3 Objective Function

In automated differential cryptanalysis using SAT or SMT solvers, a crucial step is defining an objective function that guides the search toward trails with higher probabilities. This is typically achieved by introducing dummy binary variables $w_i \in \{0, 1\}$, where each variable encodes the contribution of a specific operation to the overall weight of a differential trail. Let u be the number of weight variables. The goal is to minimize the total weight

$$\sum_{i=0}^{u-1} w_i.$$

Since standard SAT solvers do not directly support optimization objectives, we instead impose a constraint on the total weight:

$$\sum_{i=0}^{u-1} w_i \leq k,$$

where k is a user-defined upper bound on the total weight. This constraint can be efficiently encoded into `CNF` using the *sequential encoding method* [Sin05], which introduces dummy variables $u_{i,j}$ (with $0 \leq i \leq u-2$, $0 \leq j \leq k-1$) to enforce the cardinality limit. The following clauses ensure that any assignment violating the weight limit results in an unsatisfiable formula:

$$
\begin{aligned}
&(w_0 \vee u_{0,0}) \wedge (u_{0,j}) \wedge (w_i \vee u_{i,0}) \wedge (u_{i-1,0} \vee u_{i,0}) \\
&\wedge (w_i \vee u_{i-1,j-1} \vee u_{i,j}) \wedge (u_{i-1,j} \vee u_{i,j}) \\
&\wedge (w_i \vee u_{i-1,w-1}) \wedge (w_{u-1} \vee u_{u-2,w-1})
\end{aligned}
$$

for $1 \leq i \leq u-2$, $1 \leq j \leq k-1$. These clauses effectively restrict the number of active weight variables to at most k, ensuring that the solver explores only promising trails.

Unlike SAT solvers, modern SMT solvers, such as `CVC5` and `Z3`, support native optimization objectives. This allows us to minimize the total weight directly without manually imposing bounds via cardinality constraints. For example, in the `CVC` and $\texttt{CNF}_{\texttt{LF}}$ format, we can define bit-vector variables representing the weight contributions and then minimize their sum:

```
VAR
    k : BITVECTOR(u);
    w_0,w_1,...,w_(u-1): BITVECTOR(1)
"ASSERT(k=BVPLUS(u,w_0,w_1,...,w_(u-1)))";
MINIMIZE k
```

4 Experimental Results

In this section, we present and compare experimental results for seven lightweight block ciphers. Each cipher is analyzed separately over selected round ranges and differential weight bounds using a unified modeling and solving framework. The full results are summarized in Table 1, where each row corresponds to a specific cipher, round range, weight range, and encoding. For each such configuration, the table reports the average runtime of four different solvers as separate columns.

Three encoding strategies are considered: `SMT-LIB`-style `CVC`, standard conjunctive normal form (`CNF`), and size-optimized $\texttt{CNF}_{\texttt{LF}}$ generated by Logic Friday, all of which are ultimately converted to `CNF` before being passed to the SAT solver. These encodings are evaluated using four solver backends: `CryptoMiniSat v5`, `CaDiCaL`, `Treengeling`, and the massively parallel `Mallob` system. For each encoding-solver combination, we report the average wall-clock runtime over five independent executions to reduce stochastic variance. In the case of `CryptoMiniSat-v5` and `Treengeling`, only one run is reported for some ciphers due to excessive runtime. For others, even a single execution did not complete within a reasonable time limit, as indicated by the "–" symbol in Table 1. To ensure the correctness of our differential models, output probabilities and weight computations have been cross-validated against trusted reference works [SWW21, Ste].

The round and weight bounds for each cipher were selected based on two practical considerations: (1) computational feasibility, as full exploration across all round counts and weight combinations would be prohibitively time-consuming, and (2) runtime visibility, since later-round instances typically incur higher solving times, making performance differences between solvers more observable and the comparisons more informative.

All experiments were conducted on a desktop machine running Ubuntu 22.04.5 LTS with the Linux kernel 6.8.0. The system is equipped with a 13th Gen Intel® Core™ i7-13700K processor (24 threads, up to 5.4 GHz turbo boost) and 64 GiB of RAM. Benchmarks were performed in a single-user environment to ensure resource consistency.

Table 1: Unified comparison of encoding strategies and solver run-times for seven lightweight block ciphers across selected rounds and differential weight bounds.

Cipher	Rounds	w	Encoding	$\texttt{T}_{\texttt{CryptoMiniSat}}$(s)	$\texttt{T}_{\texttt{CaDiCaL}}$(s)	$\texttt{T}_{\texttt{Treengeling}}$(s)	$\texttt{T}_{\texttt{Mallob}}$(s)
SPECK-32	15–17	50–63	CVC	–	72688.92	450815.29	28961.95
			CNF	–	26510.69	27827.62	**16773.82**
			$\texttt{CNF}_{\texttt{LF}}$	–	61419.44	132351.23	28002.53
CHAM-64	29–31	39–46	CVC	88191.15	30920.49	6670.23	6813.53
			CNF	40265.00	25930.79	3882.11	**3604.28**
			$\texttt{CNF}_{\texttt{LF}}$	93539.85	30584.30	10981.80	6408.91
SIMON-32	30–32	80–90	CVC	20505.68	2956.85	3914.43	**727.70**
			CNF	20008.64	8722.44	31350.13	1490.26
			$\texttt{CNF}_{\texttt{LF}}$	44615.49	10704.35	30731.87	1547.23
PRESENT	29–31	124–136	CVC	27466.68	10155.67	34933.00	3532.13
			CNF	40042.46	9110.09	92383.47	2448.33
			$\texttt{CNF}_{\texttt{LF}}$	24674.96	6916.66	17925.05	**2383.09**
GIFT-128	10–13	45–67	CVC	502515.59	50215.32	–	21549.09
			CNF	209664.23	54195.14	–	14587.91
			$\texttt{CNF}_{\texttt{LF}}$	435965.99	40737.46	–	**14312.07**
MIDORI-64	14–16	135–168	CVC	–	17664.82	140832.67	10160.83
			CNF	–	16565.82	201050.68	5278.16
			$\texttt{CNF}_{\texttt{LF}}$	–	13686.85	107484.33	**4684.26**
LBLOCK	27–29	121–135	CVC	28784.45	2960.65	9718.11	2427.87
			CNF	21165.68	4684.28	23929.41	1589.65
			$\texttt{CNF}_{\texttt{LF}}$	48227.89	2059.20	7461.84	**1271.40**

4.1 Results on SPECK-32

SPECK-32 is a lightweight ARX-based block cipher introduced by the NSA in 2013 [BSS+15]. It operates on 32-bit blocks using modular addition, XOR, and rotations, and consists of 22 rounds with a 64-bit key.

For SPECK-32, the differential analysis was performed over the range 15–17 rounds, with weight bounds selected from 50 to 63. Within this range, rounds 15, 16, and 17 were found to be SAT for weights 54, 58, and 63, respectively, while all lower weights in each round resulted in UNSAT outcomes. As a representative example, the SMT-LIB CVC model for round 17 at weight 63 resulted in 16,590 variables and 47,174 clauses; the standard CNF encoding yielded 16,833 variables and 36,412 clauses; and the Logic Friday–optimized CNF_{LF} version contained 26,535 variables and 77,009 clauses. Among all tested configurations, the combination of CNF encoding with the parallel Mallob solver yielded the best performance in terms of runtime efficiency.

4.2 Results on CHAM-64

CHAM-64 is a lightweight ARX-based block cipher [KRK+17, RKJ+19], designed for resource-constrained devices. It employs a 4-branch generalized Feistel structure with 88 rounds.

For this cipher, the differential weight range of 39 to 46 was selected. Within this range, rounds 29 to 31 correspond to satisfiable weights 41, 43, and 46, respectively. As a representative case, for the final satisfiable instance at weight 46 and round 31, the CVC encoding leads to 37,332 variables and 102,824 clauses; the CNF encoding results in 22,369 variables and 48,811 clauses; and the CNF_{LF} encoding yields 52,987 variables and 149,789 clauses. The best-performing encoding–solver pair in terms of solving time for this cipher is CNF combined with Mallob.

As illustrated in Figure 2, the solving time per differential weight is shown for each encoding–solver pair. It can be observed that CNF combined with Mallob achieves the lowest solving time across almost all weights. Additionally, SAT solvers tend to perform significantly faster when the problem instance is satisfiable, highlighting the importance of satisfiability in runtime behavior.

4.3 Results on SIMON-32

SIMON-32 is a lightweight block cipher proposed by the NSA [BSS+15]. It uses a Feistel structure over 32 rounds, operating on 32-bit blocks with a 64-bit key. The design employs only simple bitwise operations—AND, XOR, and left circular shifts—optimized for hardware efficiency.

For this cipher, the differential weight range of 80 to 90 was selected. Within this range, rounds 30 to 32 correspond to satisfiable weights 84, 86, and 90, respectively. As a representative case, for the final satisfiable instance at weight 90 and round 32, the CVC encoding results in 60,873 variables and 176,327 clauses; the CNF encoding produces 49,094 variables and 104,690 clauses; and the CNF_{LF} encoding yields 55,465 variables and 157,031 clauses. The best-performing encoding–solver pair in terms of runtime for this cipher is CVC combined with Mallob. Figure 3 clearly illustrates this observation, demonstrating that CVC consistently results in lower solving times across different differential weights.

4.4 Results on PRESENT

PRESENT is a well-known 64-bit lightweight block cipher designed for hardware efficiency, proposed by Bogdanov *et al.* [BKL+07]. It follows an SPN structure with 4-bit S-boxes and a bitwise permutation, supporting 80- and 128-bit keys over 31 rounds.

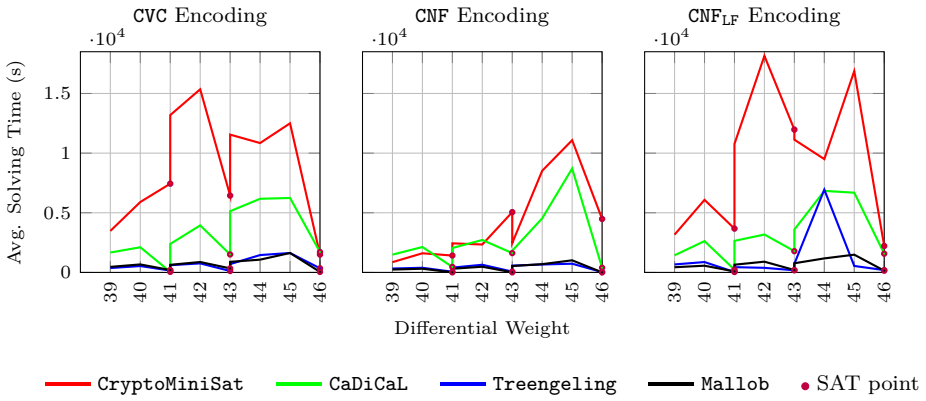

Figure 2: Comparison of average solving time per differential weight for `CHAM-64` using different encoding strategies and SAT solvers. Filled circles highlight SAT results. Each subplot shows one encoding with solvers.

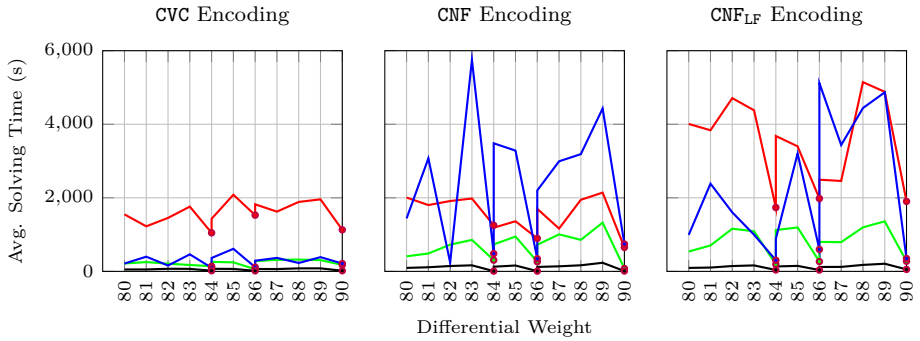

Figure 3: Comparison of average solving time per differential weight for `SIMON-32` using different encoding strategies and SAT solvers. Filled circles highlight SAT results. Each subplot shows one encoding with solvers.

For `PRESENT`, the differential search was carried out over rounds 29–31 with weight bounds ranging from 124 to 136. Within this range, rounds 29, 30, and 31 were found to be SAT at weights 128, 132, and 136, respectively, while all lower weights for each round were UNSAT. As a representative case, the `CVC` SMT representation for round 31 at weight 136 contains 1,552,022 variables and 4,773,255 clauses; the standard `CNF` encoding consists of 205,768 variables and 433,096 clauses; and the `CNF`$_{LF}$ encoding yields 75,577 variables and 303,103 clauses. Among all tested solver–encoding combinations, the fastest execution was achieved using the `CNF`$_{LF}$ encoding paired with the `Mallob` parallel solver.

4.5 Results on `GIFT-128`

`GIFT-128` is a 40-round lightweight block cipher based on an SPN structure [BPP+17]. It operates on 128-bit blocks using 4-bit S-boxes, a fixed bit permutation, and round-dependent key and constant injection.

For `GIFT-128`, the differential characteristic search was conducted over rounds 10–13 within the weight interval of 45 to 67. In this setting, the SAT results were observed for weights 49, 54, 60, and 67 corresponding to rounds 10, 11, 12, and 13, respectively, while lower weights yielded UNSAT outcomes. As an illustrative example, the `CVC` SMT-based formulation for round 13 at weight 67 comprises 656,793 variables and 2,067,862 clauses;

the standard CNF encoding includes 86,589 variables and 190,743 clauses; and the $\texttt{CNF}_{\texttt{LF}}$ representation yields 60,533 variables and 236,943 clauses. The best overall performance in terms of solver runtime was achieved using the $\texttt{CNF}_{\texttt{LF}}$ encoding in conjunction with the parallel Mallob backend.

4.6 Results on MIDORI-64

MIDORI-64 is a lightweight 64-bit block cipher [BBI+15], targeting resource-constrained environments. It follows an SPN structure and supports two key sizes: 128 and 64 bits. MIDORI-64 uses a single 4-bit involutive S-box and a lightweight almost-MDS matrix for diffusion, designed to minimize energy consumption while offering competitive security margins.

For MIDORI-64, differential SAT-based exploration was performed over rounds 14–16 within the weight interval of 135 to 168. SAT results were obtained for weights 144, 150, and 168, corresponding to rounds 14, 15, and 16, respectively, while all lower weights yielded UNSAT verdicts. For the final weight tested—168 at round 16—the CVC representation comprises 3,145,065 variables and 9,422,663 clauses; the standard CNF encoding contains 131,736 variables and 280,584 clauses; and the optimized $\texttt{CNF}_{\texttt{LF}}$ form yields 121,945 variables and 354,071 clauses. The fastest runtime across all solver–encoding combinations was achieved using $\texttt{CNF}_{\texttt{LF}}$ paired with the Mallob solver.

4.7 Results on LBLOCK

LBLOCK is a lightweight 64-bit block cipher proposed in 2011 [WZ11]. It follows a 32-round Feistel structure, using a 4-bit S-box and a simple bit permutation for diffusion. Its low area footprint makes it suitable for constrained devices such as RFID and sensor networks.

For LBLOCK, differential search experiments were conducted across rounds 27–29 over the weight interval 121 to 135. SAT results were obtained for weights 126, 131, and 135 corresponding to rounds 27, 28, and 29, respectively, while other weight–round pairs in this range were UNSAT. For the final weight tested—135 at round 29—the CVC format includes 2,853,345 variables and 8,545,871 clauses; the standard CNF encoding comprises 97,369 variables and 202,624 clauses; and the optimized $\texttt{CNF}_{\texttt{LF}}$ encoding yields 83,178 variables and 236,066 clauses. The best-performing encoding–solver combination in terms of runtime was $\texttt{CNF}_{\texttt{LF}}$ with the Mallob solver. Figure 4 reflects this trend, showing the solving time per differential weight for each encoding–solver pair.

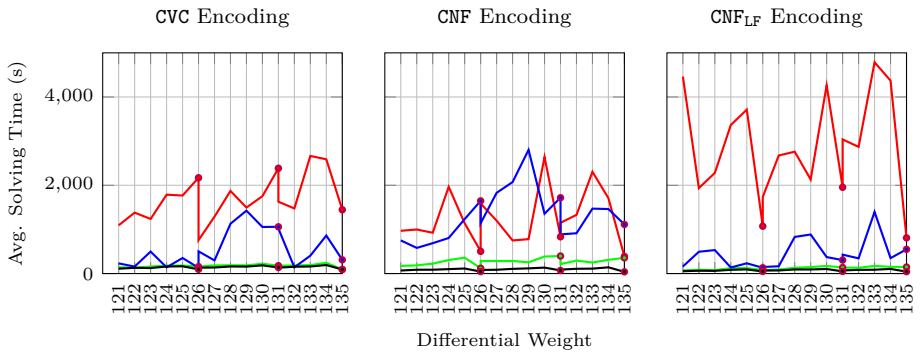

Figure 4: Comparison of average solving time per differential weight for LBLOCK using different encoding strategies and SAT solvers. Filled circles highlight SAT results. Each subplot shows one encoding with solvers.

5 Discussion and Conclusion

This study has presented a comprehensive empirical evaluation of encoding methodologies and solver frameworks for automated differential cryptanalysis. Three distinct encodings - Satisfiability Modulo Theories (SMT) in the `CVC` dialect, standard Conjunctive Normal Form (`CNF`), and size-optimized $\texttt{CNF}_{\texttt{LF}}$ generated via Logic Friday - have been rigorously benchmarked against seven diverse lightweight block ciphers (`SPECK-32`, `CHAM-64`, `SIMON-32`, `PRESENT`, `GIFT-128`, `MIDORI-64`, `LBLOCK`) using four classes of SAT/SMT solvers (`CryptoMiniSat-v5`, `CaDiCaL`, `Treengeling`, and `Mallob`).

Although every model is ultimately solved in `CNF`, the resulting formula depends strongly on the encoding path. Across our benchmarks, we observed clear shifts in variable and clause counts, preservation of XOR/parity relations, gate decomposition and auxiliary signals, and the way the minimum-weight objective is encoded.

These structural differences steer `CDCL` dynamics (unit propagation per decision, conflict frequency, VSIDS/LBD statistics) and thereby runtime. For example, in `PRESENT` and `GIFT-128`, $\texttt{CNF}_{\texttt{LF}}$ reduced the number of variables but produced more, shorter clauses, which improved propagation and led to faster solves; in contrast, for ARX ciphers such as `SPECK-32` and `CHAM-64`, the raw `CNF` often performed better. Detailed counts and timings supporting these observations appear in Section 4.

Our results have established concrete criteria for selecting optimal encoding-solver pairings, balancing modeling simplicity with computational performance. Crucially, the massively parallel framework `Mallob` has emerged as the state-of-the-art solution for large-scale differential cryptanalysis tasks, delivering an average speedup of ≈ 2.9 over the best single-core baseline across our benchmarks.

Furthermore, our solver-specific analysis reveals that multi-core parallelization does not inherently guarantee superior performance over single-threaded approaches. While one might expect multi-threaded solvers to outperform their single-core counterparts consistently, our results in Table 1 contradict this intuition.

Notably, `CaDiCaL`—a single-threaded, general-purpose SAT solver—outperforms the cryptography-focused `CryptoMiniSat-v5` and even surpasses the multi-threaded `Treengeling` in several configurations, underscoring the significance of efficient preprocessing and clause learning techniques over mere parallelism. That said, `Treengeling`'s performance tends to improve when solving structurally larger problems or ciphers with more expansive search spaces, where its decomposition strategy can better exploit available cores.

In the multi-core category, `Treengeling` employs a decomposition-based strategy, partitioning the input formula into multiple subproblems that are solved concurrently using shared-memory threads. However, this approach often fails to scale effectively in our cryptanalytic benchmarks. In contrast, `Mallob` adopts a portfolio-style parallelism model: it launches multiple instances of single-threaded solvers (e.g., `Kissat`, `CaDiCaL`, `Lingeling`), each seeded differently but solving the same problem instance. Clause learning is coordinated via a centralized communication backend, enabling efficient knowledge sharing across workers. Although `Mallob` does not decompose the input structurally, it consistently outperforms both single-threaded and decomposition-based frameworks, making it the most robust solution for differential cryptanalysis workloads in our evaluation.

For all `Mallob`-based experiments, we utilized the `kcl` portfolio configuration, which cyclically assigns each thread to a different backend solver (`Kissat`, `CaDiCaL`, `Lingeling`), further enhancing solver diversity and reducing search space overlap across threads.

Furthermore, the investigation has revealed a significant structural dependency in encoding efficiency: For S-box-based ciphers (e.g., `PRESENT`, `GIFT-128`, `MIDORI-64`, `LBLOCK`), the $\texttt{CNF}_{\texttt{LF}}$ encoding consistently yields the fastest solving times, with the objective modeled as a direct sum of weight contributions. Conversely, for ARX-based constructions (`SPECK-32` and `CHAM-64`), standard `CNF` modeling proves empirically superior, where the weight-minimization objective is enforced through cardinality constraints encoded via the

sequential encoding method.

Notably, based on the results presented in Subsection 4.3, for SIMON-like ciphers, the CVC encoding consistently demonstrates superior performance in terms of solving time. This is primarily due to the nature of the SIMON-like round function, which includes an AND operation with dependent inputs. In the CVC encoding, such functions can be efficiently modeled using conditional constructs (e.g., IF statements) without introducing dummy variables. In contrast, both CNF and CNF_{LF} encodings require additional dummy variables to represent the same logic, leading to larger and more complex formulas that degrade solver performance.

As an empirically grounded guideline, use CNF for ARX, CNF_{LF} for SPN, and CVC for SIMON-like block ciphers; pair the model with Mallob by default for time-to-solution and robustness, and fall back to CaDiCaL when constrained to a single core.

In terms of modeling effort, the CVC encoding is the most convenient due to its higher-level abstraction, followed by CNF_{LF} and then standard CNF, which requires more manual formulation.

These findings provide crucial guidance for researchers, enabling more efficient implementation choices in both the analysis and design of cipher structures vulnerable to differential attacks. The classification of efficient encodings based on cipher structure offers the potential for significant time reduction in future cryptanalysis. As a further idea, given the impact of encoding the weight-minimization objective on solver performance, refining and optimizing this encoding in CNF formulations could further improve efficiency and runtime.

Promising avenues for future work include extending this methodology to other cryptanalytic attacks (e.g., linear, integral) and systematically identifying efficient solver-encoding combinations for each specific attack type on broader classes of cryptographic primitives. Moreover, while SAT-based approaches are highly effective for bit-oriented lightweight ciphers, they are less suitable for more complex designs such as AES. In particular, bit-level SAT modeling of 8-bit S-boxes leads to an explosion in variables and clauses, making direct analysis inefficient; in such cases, MILP or CP formulations are generally more practical. Extending our systematic evaluation to word-oriented industrial standards such as AES and ChaCha remains an important direction for future work.

Acknowledgment

This work is based upon research funded by Iran National Science Foundation (INSF) under project No.4026442.

References

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M Youssef. Milp modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Transactions on Symmetric Cryptology*, pages 99–129, 2017.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 411–436. Springer, 2015.

[BDPF+23] Emanuele Bellini, Alessandro De Piccoli, Mattia Formenti, David Gerault, Paul Huynh, Simone Pelizzola, Sergio Polese, and Andrea Visconti. Differential

cryptanalysis with sat, smt, milp, and cp: a detailed comparison for bit-oriented primitives. In *International Conference on Cryptology and Network Security*, pages 268–292. Springer, 2023.

[BFF+24] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024. doi:10.1007/978-3-031-65627-9_7.

[BFH+20] Tomáš Balyo, Nils Froleyks, Marijn JH Heule, Markus Iser, Matti Järvisalo, and Martin Suda. Proceedings of sat competition 2020: Solver and benchmark descriptions. Department of Computer Science, University of Helsinki, 2020.

[Bie17] Armin Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In *Proc. of SAT Competition 2017 – Solver and Benchmark Descriptions*, volume B-2017-1 of *Department of Computer Science Series of Publications B*, pages 14–15. University of Helsinki, 2017.

[BKL+07] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*, pages 450–466. Springer, 2007.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift: A small present: Towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345. Springer, 2017.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4:3–72, 1991.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd annual design automation conference*, pages 1–6, 2015.

[DDH+20] Stéphanie Delaune, Patrick Derbez, Paul Huynh, Marine Minier, Victor Mollimard, and Charles Prud'Homme. Skinny with scalpel-comparing tools for differential analysis. *Cryptology ePrint Archive*, 2020.

[GMS16] David Gerault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings 22*, pages 584–601. Springer, 2016.

[GTM21] Praveen Kumar Gundaram, Appala Naidu Tentu, and Naresh Babu Muppalaneni. Performance of various smt solvers in cryptanalysis. In *2021 international conference on computing, communication, and intelligent systems (ICCCIS)*, pages 298–303. IEEE, 2021.

[HGSE24] Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible differential and zero-correlation attacks: Application to ascon, forkskinny, skinny, mantis, present and qarmav2. *IACR Transactions on Symmetric Cryptology*, 2024(1):234–325, 2024.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 128–157. Springer, 2023.

[IS24] Murat Burhan Ilter and Ali Aydın Selçuk. Milp modeling of matrix multiplication: cryptanalysis of klein and prince. *Turkish Journal of Electrical Engineering and Computer Sciences*, 32(1):183–197, 2024.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the simon block cipher family. In *Annual Cryptology Conference*, pages 161–185. Springer, 2015.

[KRK+17] Bonwook Koo, Dongyoung Roh, Hyeonjin Kim, Younghoon Jung, Dong-Geon Lee, and Daesung Kwon. Cham: A family of lightweight block ciphers for resource-constrained devices. In *International conference on information security and cryptology*, pages 3–25. Springer, 2017.

[LM01] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In *International Workshop on Fast Software Encryption*, pages 336–350. Springer, 2001.

[MAA24] A Mirzaie, S Ahmadi, and MR Aref. Division property-based integral attack on reduced-round sand-128. In *21st international ISC conference on information security & cryptology*, 2024.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. *Cryptology ePrint Archive*, 2013.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.

[RKJ+19] Dongyoung Roh, Bonwook Koo, Younghoon Jung, Il Woong Jeong, Dong-Geon Lee, Daesung Kwon, and Woo-Hwan Kim. Revised version of block cipher cham. In *International Conference on Information Security and Cryptology*, pages 1–19. Springer, 2019.

[Sin05] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009. doi:10.1007/978-3-642-02777-2_24.

[SPGS20] Harish Kumar Sahu, N Rajesh Pillai, Indivar Gupta, and Rajendra Kumar Sharma. Smt solver-based cryptanalysis of block ciphers. *SN Computer Science*, 1:1–12, 2020.

[SS24] Dominik Schreiber and Peter Sanders. Mallobsat:: Scalable sat solving by clause sharing. *Journal of Artificial Intelligence Research*, 80, 2024.

[Ste] Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. https://github.com/kste/cryptosmt.

[SW23] Ling Sun and Meiqin Wang. Sok: modeling for large s-boxes oriented to differential probabilities and linear correlations. *IACR Transactions on Symmetric Cryptology*, pages 111–151, 2023.

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the sat method. *IACR Transactions on Symmetric Cryptology*, pages 269–315, 2021.

[WZ11] Wenling Wu and Lei Zhang. Lblock: a lightweight block cipher. In *International conference on applied cryptography and network security*, pages 327–344. Springer, 2011.