A Complete Tolerant Algebraic Side-Channel Attack for AES with CP

Fanghui Liu, Waldemar Cruz, and Laurent Michel(✉)

Computer Science and Engineering Department, School of Engineering, University of Connecticut, Storrs, CT 06269-4155, USA
{fanghui.liu,waldemar.cruz,laurent.michel}@uconn.edu

Abstract. Tolerant Algebraic Side-Channel Attack (TASCA) is a combination of algebraic and side-channel analysis with error tolerance. Oren et al., used mathematical programming to implement TASCA over a round-limited version of AES. In [7], Liu et al. revisited their results and introduced a TASCA-CP model that delivers solutions to this 1-round relaxation with orders of magnitude improvement in both solving time and memory consumption.

This paper extends the result and considers TASCA for the full 10-rounds AES algorithm. Two approaches are introduced: staged and integrated. The staged approach uses TASCA-CP as a spring board to enumerate and check its candidate solutions against the requirements of subsequent rounds. The integrated model formulates all the rounds of AES together with side-channel constraints on all rounds within a single unified optimization model. Empirical results shows both approaches are suitable to find the correct key of AES while the integrated model dominates the staged both in simplicity and solving time.

Keywords: Algebraic side-channel attack · AES
Cryptography · Block cipher · Constraint programming · Optimization

1 Introduction

Side-Channel Analysis (SCA) is a type of attack that exploits the physical properties of a device performing a cryptographic operation, the goal of which is to obtain secret information (e.g., a secret key) from a cryptographic system. A side-channel attack typically needs hundreds to thousands of power traces to reduce the sensitivity to noise from measurement or decoding. *Algebraic* Side-Channel Attack (ASCA) was introduced in [19] to combine algebraic cryptanalysis with side-channel attack. It was applied to AES in [20]. Compared to SCA, ASCA requires much less power traces ([20] showed that as little as a single power trace is enough to recover the secret key of AES). However, the success of ASCA heavily depends on the accuracy of the side-channel information because it does not tolerate power trace measurement errors. Improved ASCA (IASCA) [12] improved the performance of ASCA on AES by optimizing AES and algebraic

J. Hooker (Ed.): CP 2018, LNCS 11008, pp. 259–275, 2018.
https://doi.org/10.1007/978-3-319-98334-9_18

representation, it also introduced a method for error handling. Tolerant Algebraic Side-Channel Attack (TASCA) [14] transforms a side-channel analysis problem to a pseudo-boolean optimization problem where the objective is the minimization of the total deviation from the measured side-channel signal.

The TASCA for AES, as presented by Oren et al. [16], models the encryption algorithm as a series of pseudo-boolean equations and side-channel constraints (i.e., the Hamming weights representing the side-channel signal). The formulation allows for providing either the plaintext alone, or both the plaintext and the ciphertext. The key was recovered with up to 20% of error rate, which refers to the probability that the measured Hamming weights are incorrect. The key is counted as correct if four or less bytes of the sixteen bytes of the key are incorrect.

In 2017, Liu et al. [7] revisited the TASCA attack on AES and offered a CP formulation based on bit-vectors [8]. Their model, like that of Oren et al. [16], focused on the first round of AES and provided an approach that attempts to recover the key with low complexity. By using CP over bit-vectors with a customized search, it was possible to recover candidate keys for the one round problem with orders of magnitude improvements both in runtime and memory usage over the pseudo-boolean optimization and Integer Programming (IP) approaches. Yet, relying on only one round of AES and side-channel information is a relaxation of the true problem that mandates a brute-force post-processing to zero-in on the true key and rule out candidate solutions that are not conforming to the requirements of the full AES algorithm and side-channel information.

Several approaches were proposed to apply Constraint Programming to crypt-analysis. [18] used a CSP framework to design substitution functions for substitution permutation (SP) networks. [4] introduced a chosen key differential crypt-analysis using Constraint Programming models, and it was performed against AES-128 in [5], AES-192 and AES-256 in [3]. Based on [4,21] apply CP to search for differential/linear characteristics, integral distinguishers and performed the analysis on AES, PRESENT [2] and SKINNY [1].

This paper explores extensions of the CP approach to natively handle the full 10 rounds of AES and side-channel information. The paper presents two approaches for doing so and evaluates them empirically alongside the natural extensions that could be offered for the earlier pseudo-boolean formulation. The *staged* approach uses TASCA-CP as a "black box" subroutine to produce solutions in the full model. The *integrated* approach extends the constraint programming model to directly handle all 10 rounds of AES alongside the side-channel information for those rounds.

The remainder of the paper is organized as follows. Section 3 recalls the 1-round TASCA approach and how it is meant to be used. Section 4 articulates the limitations of the restricted formulation, the source of relaxations and the contributions of the paper. Section 5 is the core of the paper and presents two extensions needed to consider AES in full. Section 6 discusses the empirical results while Sect. 7 concludes.

2 AES Overview

AES is an iterated block cipher that supports a fixed block size of 128 bits, and key size of 128, 192 and 256 bits. The version used in this paper is AES with 128-bit cipher key (AES-128). The cipher key is first expanded into 10 round keys via key expansion [13]. The plaintext is separated into blocks of 16 bytes, denoted as $p_0 \dots p_{15}$ (where $p_i \in \{0,1\}^8, i = 0, \dots, 15$). Each block is represented by a 4×4 matrix of bytes.

There are four elementary operations that build the whole function of AES:

- **SubBytes:** Apply a permutation on the input bytes using an "S-Box" [11].
- **ShiftRows:** Apply a byte-wise left-rotate operation for each of the four rows by 0, 1, 2, and 3 bytes.
- **MixColumns:** Multiply each column in the state with a matrix of constants.
- **AddRoundKey:** XOR the state with the round key.

During the encryption, the plaintext is first combined with the initial round key (cipher key) and then goes through 9 rounds of iteration consisting of 4 subrounds (SubBytes, ShiftRows, MixColumns, AddRoundKey), and the last iteration consist of 3 subrounds (without MixColumns) to produce the ciphertext $c_0 \dots c_{15}$. The AES encryption process is outlined in Algorithm 1. More details of AES can be found in [13].

Algorithm 1. Pseudocode for AES Encryption Algorithm

```
function AES(byte in[16], byte out[16], key_array round_key[Nr+1])
    byte state[16]
    state = in
    AddRoundKey(state,round_key[0])
    for i = 1 to Nr-1 do
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, round_key[i])
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, round_key[Nr])
    out = state
return out
```

3 TASCA over Restricted AES

TACSA-CP presented a tolerant algebraic side-channel attack using constraint programming. The TASCA-CP model includes one round AES structural constraints, one round side-channel constraints (which contains 10% errors) and

auxiliary constraint (e.g. plaintext). The goal is to minimize the errors in the side-channel information. With a customized search, the TASCA-CP is able to find the global optima with an overwhelmingly better performance compared to pseudo-boolean model with SCIP [16] and IP model with Gurobi [7].

The global optima is then given to a CP solver to search for all candidate solutions. An attacker needs to enumerate over the candidate solutions to discover the true key used in the captured AES encryption process.

4 TASCA over Full AES

The previous section showed that it is possible to carry out a Tolerant Algebraic Side-Channel Attack using CP on one round AES encryption with one round side-channel information. The empirical results established that the performance of TASCA-CP is significantly better than the IP models making CP the technology of choice to conduct this type of side-channel attacks. Nonetheless, it is *essential* to remember that the TASCA approach considered earlier remains a resolution technique for a *relaxation* of the true problem. Indeed, with only one round of the AES computation being modeled, one can only find *candidate solutions*. Those candidate solutions may not satisfy the structural constraints imposed by the subsequent rounds of AES and they may deliver a ciphertext that is different from the sought after solution. Specifically, it is worth noting that there are two distinct relaxations.

Round. The 1-round relaxation was essential in the case of the integer programming approach to sidestep difficulties arising from the sheer size of the model. Yet, the relaxed problem is still challenging and requires non-trivial efforts. While the relaxation will produce all the bits of the key and fix all the bits of the output state of round 1, those output bits should still be subjected to three classes of constraints: (1) the structural constraints of state derivation dictated by AES; (2) the side-channel constraints based on the measurements done in rounds 1–10; and (3) the equality to the target ciphertext when the attack is carried out with a known pair cleartext, ciphertext. This limitation can still be acceptable if the number of candidate solutions remains small, in which case an exhaustive check can verify each candidate solution. Yet, observe that since this is a relaxation, it is quite possible that *all candidate solutions* associated to the global optima of the relaxation are infeasible in the full problem. This indicates that the global optimum of the full problem has an objective value worse than what was reported by the relaxation. The instances considered in [7] were engineered to have a tight relaxation, i.e., the global optimum of the relaxation was also the global optimum of the true problem.

Tolerance. Both the MIP and the CP models from [7] consider that side-channel measurements for a state byte b are hamming weights estimates ($\tilde{H}(b)$) and may be off from their true value $H(b)$ by ± 1, i.e., for any state byte b, $\tilde{H}(b) - 1 \leq H(b) \leq \tilde{H}(b) + 1$. With both technologies, the optimization models see

a byte b as a decision variable that must satisfy these inequalities. If the measurement estimates are off by a wider margin, the true solution can never be found since the model will exclude it.

The purpose of this paper is to articulate a solution methodology that mitigates the shortcomings introduced by these relaxations to obtain a direct resolution technique capable of computing the key used by an AES encryption of a known plaintext. Specifically, the methodology must scale to support all 10 rounds of AES, exploit any additional side-channel measurements provided for those rounds, and gracefully handle the tolerance bounds. Note that the bit-vector formulation employed by the CP model holds the promise of a scalable representation as a single round consumes from 20 to 40 times less memory and delivers runtimes that are two order of magnitude better than state-of-the-art mathematical programming solvers.

5 Approaches

The remainder of the paper considers two natural approaches and articulates the rationale behind them and the needed improvements to make them competitive. Before describing the two approaches, a few preliminaries are in order.

5.1 Preliminaries

At a macroscopic level, the optimization model for one AES round uses decision variables to represent the input state of the round, the output state of the round and the round key. Its constraints define the relations connecting the input state and round key through intermediary states to the output state of the round. They also constrain the hamming weights of each state to be within the error tolerance requirements. The objective function collects all the deviation errors, positive or negative, that can be experienced to match the hamming weights with their estimates.

Definition 1 (State Variables). *Let S_k denote the 128-bit wide bit-vector variable denoting the input state to round $k \in 1..10$. Similarly, let O_k denote the 128-bit wide bit-vector variable denoting the output state of round k. Finally, let $I_{k,r}$ denote the 128-bit wide bit-vector variable denoting the internal state r within round k. Namely, this is the state after each stage encountered within a round of AES, e.g., the add round key, S-box and mix column stages ($r \in 1..4$).*

Definition 2 (Key Variable). *Let K_k denote the 128-bit wide bit-vector variable denoting the round key for round k.*

Definition 3 (Byte structure). *Given a state variable S (a 128-bit wide bit-vector), let S^b be a byte corresponding to the 8-bit subsequence of S aligned on an 8-bit boundary.*

Clearly, any state variable S is broken down into its 16 constituent bytes, numbered 0 through 15.

Definition 4 (Error Variable). *$E^{b+}_{k,r}$ and $E^{b-}_{k,r}$ are the non-negative slack variables modeling noise in a side-channel equation. The binary variable $E^{b+}_{k,r}$ models the positive error while $E^{b-}_{k,r}$ models the negative error for byte $b \in 0..15$ of round $k \in 1..10$ and state $r \in 1..4$.*

Definition 5 (Hamming Weight Estimates). *Let $\tilde{H}(S^b)$ denote the estimate of the hamming weight for any state byte S^b ($b \in 0..15$) of a state S.*

Definition 6 (Hamming Weights). *Let $H(S^b)$ denote the actual hamming weight of byte $b \in 0..15$ for a state S.*

AES Constraints

- AddRoundKey $ARK(X, Y, Z)$ is implemented as a bit-wise XOR operation between two 8-bit bit-vector variables, $X \oplus Y = Z$.
- SubBytes $SubByte(X, Z)$ is implemented as an *element constraint* [7] over bit-vectors. The element constraint $Z = c[\mathcal{I}(X)]$ takes in an array c of (constant) bit-vectors and requires Z to be equal to the $\mathcal{I}(X)^{th}$ entry of the array c. An array of 256 constant bit-vectors model the full substitution box.
- ShiftRows $ShiftRows(X, Z)$ is a logical circular shift on input variable and there are no changes in the values. Therefore ShiftRows operation does not leak any side-channel information and is combined with MixColumns [15].
- MixColumns $MixColumns(X, Z)$ is a complex operation that applies to a column of input variable matrix at a time. An 8-bit efficient MixColumns implementation [17] is used in CP encoding. Suppose $[a_0, a_1, a_2, a_3]$ is a column of input X, $[o_0, o_1, o_2, o_3]$ is a column of output Z, the MixColumns operation can be expressed as follows:

$$o_k = \left(\bigoplus_{i=0}^{3} a_i\right) \oplus \texttt{xtime}(a_k \oplus a_{(k+1) \bmod 4}) \oplus a_k \;\; \forall\, k \in 0..3$$

  bit-vector constraints are used to capture XOR as well as `xtime` [13].

Side-Channel Constraint: are created using `count` [7] constraint on bit-vectors. Using count constraint the actual hamming weight $H(I^b_{k,r})$ is encoded as $count(I^b_{k,r})$. The side-channel constraint is formulated as:

$$H(I^b_{k,r}) + E^{b+}_{k,r} - E^{b-}_{k,r} = \tilde{H}(I^b_{k,r})$$

It is now possible to define a COP for round k.

Definition 7 (AES Round Model). *Given a round $k \in 1..10$, the TASCA-CP model for round k is the COP*

$$M_k = \langle X_k, C_k, f_k \rangle$$

Where

$$X_k = S_k \bigcup_{r \in 1..4} I_{k,r} \cup O_k$$

and

$$\begin{array}{ll} C_k = ARK(S_k, K_k, I_{k,1}) & \cup SubBytes(I_{k,1}, I_{k,2}) \quad \cup \\ \quad ShiftRows(I_{k,2}, I_{k,3}) & \cup MixColumns(I_{k,3}, I_{k,4}) \cup \\ \quad I_{k,4} = O_k & \cup \\ \quad \bigcup_{r \in 1..4, b \in 0..15} H(I^b_{k,r}) + E^{b+}_{k,r} - E^{b-}_{k,r} = \tilde{H}(I^b_{k,r}) & \end{array}$$

and the objective to minimize is s

$$f_k = \sum_{r \in 1..4, b \in 0..15} E^{b+}_{k,r} + E^{b-}_{k,r}$$

Constraints in M_k link the input state S_k to the output state O_k of round k.

Definition 8 (Whole Model). *While C_k represents the AES constraints and side-channel constraints for round k, C represents the AES constraints and side-channel constraints for all 10 rounds. Similarly, f represents the overall sum of errors for 10 rounds:*

$$f = \sum_{k \in 1..10} f_k$$

With the optimization models for the rounds of AES formalized, it is now possible to formally describe two approaches to solve the full AES model.

5.2 A Staged Approach

Consider the AES Round model $M_1 = \langle X_1, C_1, f_1 \rangle$ that captures the first round of AES. The COP model considered in [7] can be derived from M_1 by adding two additional sets of constraints to capture the *cleartext* requirement and the *key schedule*. The cleartext requirement is straightforward and consist in binding the bytes of S_1 to their value in the cleartext. The key schedule requirement derives the round keys $K_1, K_2, \cdots, K_{10}$ from the AES key K with invertible operations. Namely, the 16 bytes (byte number is the superscript) of the round key K_r are defined with

$$K^0_r = SubByte(K^{13}_{r-1}) \oplus K^0_{r-1} \oplus RC_r \quad (1)$$
$$K^1_r = SubByte(K^{14}_{r-1}) \oplus K^1_{r-1} \quad (2)$$
$$K^2_r = SubByte(K^{15}_{r-1}) \oplus K^2_{r-1} \quad (3)$$
$$K^3_r = SubByte(K^{12}_{r-1}) \oplus K^3_{r-1} \quad (4)$$
$$\forall i \in \{0, ..., 11\}\ K^{i+4}_r = K^i_r \oplus K^{i+4}_{r-1} \quad (5)$$

where $r \in \{1, \ldots, 10\}$, and $K_0 = K$. RC_r is a round-specific constant and `SubByte` is the usual non-linear S-Box of AES. The key expansion function is

encoded as a series of cascading XOR constraints and substitutions via the element constraint. In the following $\texttt{KeySchedule}(K, [K_1, \cdots, K_{10}])$ refers to the set of constraints implementing this key expansion requirement.

Therefore, the model in [7] is:

$$M_1' = \langle X_1, C_1 \cup \texttt{ClearText}(S_1) \cup \texttt{KeySchedule}(K, [K_1, \cdots, K_{10}]), f_1 \rangle$$

Clearly, this is a relaxation since this model ignores all but round 1. The resolution of that relaxation will produce a sequence of improving local optima culminating with the global optimum of the relaxation f_1^*. For any discrete value $v \in f_1^* .. \frac{|E|}{2}$, it is possible to enumerate all candidate solutions that yield that specific objective with the *constraint satisfaction problem*

$$M_1^v = \langle X, C_1 \cup \texttt{ClearText}(S_1) \cup \texttt{KeySchedule}(K, [K_1, \cdots, K_{10}]) \cup \{f_1 = v\} \rangle$$

$$M_{full} = \left\langle X, \begin{array}{c} C \cup \texttt{ClearText}(S_1) \cup \texttt{CipherText}(S_{41}), f \\ \cup \texttt{KeySchedule}(K, [K_1, \cdots, K_{10}]) \end{array} \right\rangle$$

Algorithm 2. Staged Approach to solve the Full AES model

```
1: v, f_1^* ← optimize(M_1')
2: Best ← +∞
3: F^* = ∅
4: do
5:     Sols ← solveAll(M_1^v)
6:     F ← {c ∈ Sols | validate(c, M_full ∧ f_1 = v ∧ f ≤ Best)}
7:     Best ← min(Best, min_{s∈F}(f(s))
8:     F^* = {s ∈ F^* | f(s) ≤ Best} ∪ F
9:     v ← v + 1
10: while v ≤ |E|/2 ∧ F^* = ∅ return F^*
```

This observation gives rise to an algorithm for the full AES shown in Algorithm 2. The algorithm in line 1 produces the global optima of the relaxation f_1^* and the target value v. Line 2 sets the best objective function to ∞ and line 3 defines the set of solution F^* as empty. The loop covering lines 4–10 start by deriving (line 5) the solution pool obtained if the objective function for round 1 (f_1) is forced to adopt value v. Line 6 discards any candidate solution c from the relaxation that do not correspond to the current relaxed target v nor improve upon the incumbent value $Best$. This validation process clearly uses the full AES model M_{full} which features constraints for all the rounds of AES, all the side-channel information and auxiliary information (plaintext and ciphertext). The `validate` subroutine fixes the state variables of round 1 to their value in the candidate solution c and M_{full} has all the constraints for all the rounds, all the side-channels and its objective function f captures the sum of the errors

over all rounds. Lines 7 and 8 update the value of the incumbent and the set of solutions that deliver the optimal objective value which is stored in F^*. Line 9 finally increases the target value by 1 and the loop repeats the process. The terminating condition relies on a trivial upper bound for the objective function f equal to the number of error variables divided by 2 (no byte can have both a positive and negative slack) and whether F^* is empty. $F^* \neq \emptyset$ indicates that the optimal solution is found since both the plaintext and ciphertext are fixed in M_{full} and guarantee a valid and unique solution.

It is critical to distinguish the two objective functions. f_1 is the first round while f captures the sum of errors over all rounds. It is possible to encounter a solution of the relaxation that features many deviations from the hamming weights in round 1 – hence a poor value for the relaxed objective f_1 – but very few errors in subsequent rounds, ultimately delivering an excellent value for the global objective f. The subroutines in Fig. 2 are as follows

optimize(M) Solves M and produces a local optima f_1^*.
solveAll(M) Generates a solution pool for a fixed value of the objective.
validate(c,M) Verifies, in polynomial time, that c satisfies model M.

Analysis. In most cases, the round-1 relaxation happens to be tight. Namely, its candidate solutions for the relaxation optimum coincides with the global optimum when extended to the full AES and the algorithm produces a set F of cardinality 1 with a feasible and optimal solution in the first iteration. Subsequent iteration yield empty sets F as the candidate solutions of weaker solutions from the relaxation cannot be extended and meet the $f \leq Best$ requirement. Occasionally, the algorithm will have to consider weaker candidate pools that may contain globally feasible and better solutions. In the extreme case, the algorithm may have to expand considerable resources to eventually find out that, because of the *tolerance relaxation of* ± 1, none of the candidate solutions are globally feasible. In the worst case, the algorithm may perform $\mathcal{O}(|E|)$ iterations[1] since the objective function is bounded from above by $\frac{|E|}{2}$ and still not deliver a globally optimal solution because of the *tolerance relaxation*. In practice though, it performs reasonably well thanks to the tightness of the relaxation.

These limitations indicate that an alternative approach may be worthwhile.

5.3 Integrated Approach

The AES round model formulated in Definition 7 explicitly captures a single round k of AES with a COP $M_k = \langle X_k, C_k, f_k \rangle$ in which X_k contains input, output and internal state variables. It is therefore tempting to aggregate all these models since the output state variables of round k are the input state variables of round $k+1$. Similarly, the objective function of the full AES is separable and *exactly* matches the sum of the objective functions. Namely $f = \sum_{k \in 1..10} f_k$. By accumulating all the variables, constraints and the additive objectives together

[1] Recall that E is the set of all error variables.

with the constraints for the key schedule, one can obtain a (large) mathematical formulation of the entire AES state transformation.

Definition 9 (AES Full Model). *Given n rounds, numbered* 1 *through n and their associated AES Round models* M_1 *through* M_{10}*, the AES Full Model for n rounds is the COP*

$$M(n) = \langle X, C, f \rangle$$

Where

$$X = \bigcup_{k \in 1..n} X_k \cup K$$

$$C = \bigcup_{k \in 1..n} (C_k \cup \{O_k = I_{k+1}\}) \cup ClearText(S_1) \cup KeySchedule(K, [K_1, \cdots, K_n])$$

In which the objective is to minimize

$$f = \sum_{k \in 1..n} f_k$$

This formulation encapsulates in a single family of models all the details about AES and progressively eliminates the round relaxation as n increases from 1 to 10. It still depends on the *tolerance relaxation* though. Naturally, $M(10)$ coincides with the full AES model while $M(1)$ was detailed in [7]. As before, an $M(n)$ model can be submitted 'as is' to a constraint programming solver together with a suitable search, or it can be linearized for an IP solver.

Search Heuristic Revisited. The search heuristic is focused on the semantics of the AES transformations and the side-channel information. For each of the 16 bytes of the state variables, there is a hamming weight value attributed to each state. A set of candidate values are produced for each state variable, such that the hamming weight of the value is within the tolerance range as dictated by

$$M_r^i - k \leq H(S_r^i) \leq k + M_r^i$$

Where k is the allowable discrepancy between the measurement M_r^i of byte i in round r and the candidate value for the i^{th} byte of the state variable S_r. Recall that $H(b)$ simply counts the number of bits at 1 in byte b, i.e.,

$$H(b) = \sum_{i \in 0..7} (b \wedge (1 \ll i) = (1 \ll i))$$

The goal of TASCA is to minimize the number of deviations from the predicated hamming weight value delivered from the side-channel analysis. The search heuristic is responsible for driving the variable/value choices to reach the optimal solution early. Value assignments that contribute the least to the objective function will be effective in reaching high-quality solution early.

Variable Heuristic. The search introduced in TASCA-CP used branching on multiple variables at once. Branching on a pair of variables allows for calculation of high-quality *under-estimates* of the actual errors. The main intuition is that branching on a pair of variables allowed for the propagation to fix the values of connecting variables via the AES transform constraints.

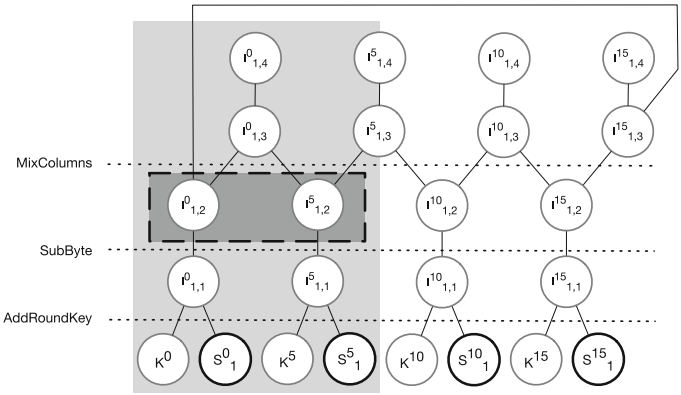

Fig. 1. Circuit for bytes $\{0, 1, 2, 3\}$.

Figure 1 offers a schematic view of the search for one quarter of the circuit modeling round 1 of AES. The schematic is best understood bottom-up. At the bottom layer, four bytes of the plaintext (Bold variables S_1^0, S_1^5, S_1^{10} and S_1^{15}) are combined via the `addRoundKey` constraints with the matching bytes of the round key (K^0, K^5, K^{10} and K^{15}) to produce the intermediate state $I_{1,1}^0$, $I_{1,1}^5$, $I_{1,1}^{10}$ and $I_{1,1}^{15}$. The internal state variables are mapped via the `SubByte` constraints to $I_{1,2}^0$, $I_{1,2}^5$, $I_{1,2}^{10}$ and $I_{1,2}^{15}$. These are then composed via the `mixColumn` operation which links the last two sets of internal state bytes (a mix of XOR and shifts). Note again, that the last internal state variables are simply made equal to the output variables of the round. The main insight is that *given the known plaintext*, the `addRoundKey` and `SubByte` transformations are bijective. Therefore labeling any variables connected to the `addRoundKey` and `SubByte` constraints will fix the other variable via propagation. In particular, labeling variable $I_{1,2}^0$, will, through propagation, fix the $I_{1,1}^0$ and K^0 variables. Fixing $I_{1,2}^5$ at the same time would, likewise, propagate to $I_{1,1}^5$ and K^5. Simultaneously fixing $I_{1,2}^0$ and $I_{1,2}^5$ would exploit the ternary constraint connecting the dotted box containing that pair to $I_{1,3}^0$ and help propagation fix $I_{1,4}^0$ and therefore O_1^0 allowing the search to get a fair estimate of the hamming weight errors associated to these variables.

Subsequently, the search will consider other pairs that reuse one of the two variables from the first pair. For instance, if $\langle I_{1,2}^0, I_{1,2}^5 \rangle$ was selected first, it is tempting to consider the two pairs $\langle I_{1,2}^{15}, I_{1,2}^0 \rangle$ and $\langle I_{1,2}^5, I_{1,2}^{10} \rangle$ as the domain of $I_{1,3}^{15}$ and $I_{1,3}^5$ are already reduced by the first choice.

Value Heuristic. The objective is driven by the sum of measurement errors on intermediate state variables. If the search considers a pair of values $\langle a, b\rangle \in D(I^0_{1,2}) \times D(I^5_{1,2})$ it can assess the impact that the simultaneous assignments $I^0_{1,2} = a \wedge I^5_{1,2} = b$ would have on the errors at the intermediate state variables in the leftmost gray column. This assessment is an under-approximation of the true error induced by the assignments. Indeed the intermediate variable $I^5_{1,3}$ can expose errors caused by the choice of value b for $I^5_{1,2}$, but that falls outside the gray column and is therefore ignored.

Therefore the value heuristic considers pairs of values and assess the quality of the pair with a scoring function SC given as:

$$SC(\langle a, b\rangle) = SC_{leg}(a, [I^0_{1,2}, I^0_{1,1}, K^0]) + SC_{leg}(b, [I^5_{1,2}, I^5_{1,1}, K^5]) + SC_{mc}(a \oplus b, I^0_{1,4})$$

The functions SC_{leg} and SC_{mc} model the error attributes to a "leg" ($[I^0_{1,2}, I^0_{1,1}, K^0]$) or to the top of the "leg" (the MixColumns operation).

Optimality Pruning. The objective function is defined as the total contribution of the errors in the predicted hamming weight values of the state variables. As the search continues to find an improving solution, the objective function will dive deeper reducing the number of errors required to find the next best satisfiable solution. At each incumbent solution, the search tree can prune any path that will not lead to an improving solution.

Overall Search. The search aims at labeling variables that contribute the least amount of error to minimize the objective function. This allows the solver to reach high quality solutions quickly. The strategy that governs the order in which search tree nodes are explored is a standard depth first search. When considering a model with multiple rounds, it makes sense to partition all the variables according to their round and branch on them in increasing round order. Within a round, the same search heuristic as before can be adopted (branching on pairs of variables).

Analysis. This search procedure is sufficient to experiment with a full AES model. Surprisingly, the first empirical results were disappointing. Indeed, the integrated approach needed a considerable search effort (10–100 times more search nodes) to deliver a global optimum often exceeding the time out limit of 10 min. The analysis of the result revealed that: (1) The search delivered only one globally optimal solution; and (2) The search did not deliver *any* sub-optimal solutions. These two observations lead to a conjecture on the search behavior:

Conjecture 1 (Degeneracy). The AES Full COP Model has only one feasible solution, even in the presence of a ± 1 tolerance relaxation.

If true, all candidate solutions from the round 1 are ultimately infeasible which propagation discovers causing backtracking early. Fundamentally, when branching in round order, once round 1 is complete, the solver *checks* the constraints of

the subsequent round as propagation alone fixes everything. In essence, candidate solutions emerging from round 1 can only be either extended to all rounds and deliver a global solution without further branching, or a constraint (i.e., a hamming weight constraint) will *fail* and cause backtracking.

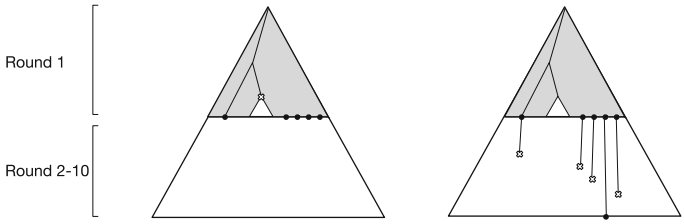

Fig. 2. Search tree comparison.

Figure 2 offers a depiction of two search trees. The left triangle represents the search tree for $M(1)$, i.e., the one-round model solved in [7]. The right triangle illustrates the search "tree" for $M(10)$, the AES full model. What it suggests is that the search over $M(1)$ delivered many feasible solutions that were used to bound away some subtrees like the small white triangle in the gray zone. However, in the right part of the picture, those solutions feasible at the boundary of round 1 become infeasible and therefore one never finds an incumbent and never tighten the upper bound. Consequently, the same little white triangle must be explored in the full model and this is what contributes to the significant increase in search effort.

Corollary 1 (DFS is unadapted). *If one (or very few) feasible solutions exist, the objective function can only be used to guide the search heuristic and it is essential to discover the (sole) feasible solution as early as possible.*

The search heuristic was designed specifically to exploit the objective function and minimize deviations from the prescribed hamming weights. Under the assumption that the heuristic is effective, a natural option is to consider *limited discrepancy search* [6] that slowly and progressively distrusts the search heuristic, authorizing a limited number of deviation from its recommendations.

Recall that there are 16 bit-vector variables to be labeled, which is a large search tree. If the feasible solution is on the far right side of the search tree, the DFS strategy will explore all nodes (left to right) before reaching the solution. Limited Discrepancy Search, by gradually increasing potential "wrong turns", increases the likelihood to hit the feasible solution sooner. The implementation of LDS was detailed in [10]. Empirical results indicate that limited discrepancy search is quite effective on this problem.

6 Experimental Setup

The adopted CP solver is Objective-CP [9] that combines modeling and search as well as user-defined searches. The IP approach relies on Gurobi (7.5.2). The experiments ran on a 16-core Intel Xeon E52640 at 2.40 Ghz with 16 MB cache and Ubuntu 16.04 LTS. 1000 instances were generated. Each instance is based on a randomly chosen plaintext and cipher key in $\{0,1\}^{128}$. Each instance contains, for its plaintext, 996 Hamming weight leaks that correspond to the full 10 rounds. For each instance, a 10% error rate is uniformly applied to the 996 Hamming weight leaks, the Hamming weights are modified by ± 1.

To evaluate the performance of the approaches considered in the paper, experiments are conducted with either 1, 2 or 10 rounds of AES encryption *and* side-channel information. In particular, the paper considers the integrated approach with the LDS and DFS search strategies as well as the staged approach.

6.1 IP Approach

The average runtime of IP using Gurobi with one round AES and side-channel information is 222 s with a standard deviation 135 s. This results matches what was reported in [7]. It becomes significantly harder for the IP solver when solving with 2 rounds of AES and side-channel information. First, ten easy (10 s), medium (300 s) and hard instances (500–600 s) are identified based on their runtime on the CP solver. This restricted set of 30 instances was then submitted to the IP solver with 2 rounds. Easy instances took the IP solver around 5 h. Medium instances took from 5 to 10 h of solving time. Out of the 10 hard instances, only one terminated with the solution right before the 24 h limit. All others timed out. Then 19 easy instances of MIP (finished within 2 h) are submitted to CP solver. For those instances that are easy for MIP, CP is over an order of magnitude faster (finished within 10 s). The results above show that CP has an overwhelming advantage over MIP.

As for the memory consumption, the IP solver took 5 to 8 times more memory than CP with LDS for solving 1 round. For 2 rounds, that memory increase factor climbs to a range of 10x–16x.

6.2 Integrated

Figure 3 below illustrates the performance for the integrated approach with 1 and 10 rounds of AES encryption and side-channel information respectively. More than 80% of instances finish within 100 s, and the majority of the instances finish within 20 s. Increasing the rounds of AES encryption and side-channel information does not heavily affect the run time.

Figure 4 reports the runtime of LDS and DFS for 10 rounds. LDS is substantially faster than DFS and has a more stable performance throughout all 1000 instances. DFS timeouts after 10 min on 30% of the instances when considering multiple rounds whereas LDS experiences only 4% of timeouts.

Table 1. Performance for LDS, DFS and Staged approach

Rounds	LDS(Integrated)		DFS(Integrated)		Staged	
	μ_T	σ_T	μ_T	σ_T	μ_T	σ_T
1	41.25	85.66	44.40	89.29	73.5	116.36
2	24.78	63.97	54.68	115.33		
10	26.71	65.74	58.58	122.34		

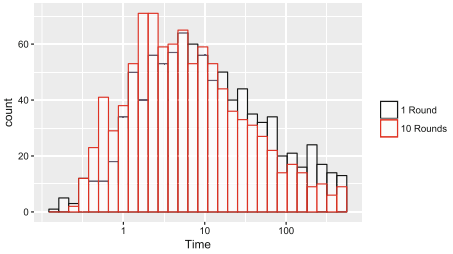

Fig. 3. LDS performance

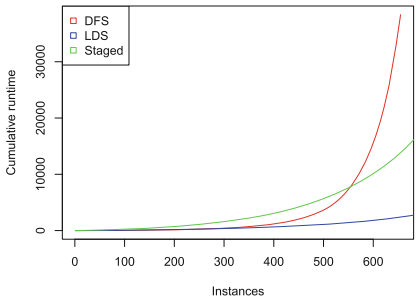

Fig. 4. Methods comparison

6.3 Staged

Since "staged" and "integrated" (10 rounds) are complete approaches, staged is compared to integrated on 10 rounds – with both LDS and DFS for integrated–. Table 1 shows that the staged approach is, on average, 3 times slower than LDS with a large deviation. Figure 4 shows that $LDS(Integrated)$ has a much smaller "surface under the curve" when compared to *Staged*. While the staged approach displays reasonable running times, it is worth remembering that it requires the availability of the ciphertext as discussed in Sect. 5.2.

6.4 Solution Pool

The premise in [16] is that one is already able to effectively attack AES with only one round of relaxation since the pool of candidate solutions from the relaxation is reasonably small and therefore the candidates it offers can be checked exhaustively. That premise is based on the assumption that the pool of candidate solution associated to the global optimum f_1^* of $M(1)$ does indeed contain the global optimum. While this assumption happens to hold for the 1,000 randomly generated instances, it is not true in general. Nonetheless, it is informative to consider how the candidate pool size evolves when going from $M(1)$ to $M(10)$, i.e., as we consider increasingly tighter relaxations. As the empirical result shows, starting at round 2, the solution pool size for all instances collapses to 1. And the only one solution in the pool is indeed the correct key.

7 Conclusion

This paper extends [7] with the ability to conduct a Tolerant Algebraic Side-Channel Attack over the complete AES algorithm. Two approaches: *Staged* and *Integrated* are introduced. The staged approach obtains the optimum from TASCA-CP model to enumerate and check all the solutions against full-round AES structural constraints and the accompanying side-channel constraints and ciphertext. The integrated model aggregates each single round AES with COP and therefore is able to construct 1 to 10 rounds of AES model with side-channel information. To improve the performance of the customized search, Limited Discrepancy is adopted by the integrated approach. The empirical results show the integrated approach with LDS is faster and more consistent than the stock DFS strategy or the staged approach and are orders of magnitude more scalable in space and time than an IP.

References

1. Beierle, C., et al.: The skinny family of block ciphers and its low-latency variant mantis. Cryptology ePrint Archive, Report 2016/660 (2016). https://eprint.iacr.org/2016/660
2. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
3. Gérault, D., Lafourcade, P., Minier, M., Solnon, C.: Revisiting AES related-key differential attacks with constraint programming. IACR Cryptology ePrint Archive 2017, 139 (2017). http://eprint.iacr.org/2017/139
4. Gerault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 584–601. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44953-1_37
5. Gerault, D., Minier, M., Solnon, C.: Using constraint programming to solve a cryptanalytic problem. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 4844–4848. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/679
6. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995, vol. 1, pp. 607–613. Morgan Kaufmann Publishers Inc., San Francisco (1995). http://dl.acm.org/citation.cfm?id=1625855.1625935
7. Liu, F., Cruz, W., Ma, C., Johnson, G., Michel, L.: A tolerant algebraic side-channel attack on AES using CP. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 189–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_13
8. Michel, L.D., Van Hentenryck, P.: Constraint satisfaction over bit-vectors. In: Milano, M. (ed.) CP 2012. LNCS, pp. 527–543. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33558-7_39
9. Michel, L., Van Hentenryck, P.: A microkernel architecture for constraint programming. Constraints **22**(2), 107–151 (2017). https://doi.org/10.1007/s10601-016-9242-1

10. Michel, L., See, A., Van Hentenryck, P.: Transparent parallelization of constraint programming. INFORMS J. Comput. **21**(3), 363–382 (2009). https://doi.org/10.1287/ijoc.1080.0313
11. Mister, S., Adams, C.: Practical S-box design. In: Selected Areas in Cryptography (1996)
12. Mohamed, M.S.E., Bulygin, S., Zohner, M., Heuser, A., Walter, M., Buchmann, J.: Improved algebraic side-channel attack on AES. J. Cryptographic Eng. **3**(3), 139–156 (2013). https://doi.org/10.1007/s13389-013-0059-1
13. NIST: Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES) (2001)
14. Oren, Y., Kirschbaum, M., Popp, T., Wool, A.: Algebraic side-channel analysis in the presence of errors. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 428–442. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_29
15. Oren, Y., Renauld, M., Standaert, F.-X., Wool, A.: Algebraic side-channel attacks beyond the hamming weight leakage model. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 140–154. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_9
16. Oren, Y., Wool, A.: Tolerant algebraic side-channel analysis of AES. IACR Cryptology ePrint Archive, Report 2012/092 (2012). http://iss.oy.ne.ro/TASCA-eprint
17. Oren, Y., Wool, A.: Side-channel cryptographic attacks using pseudo-boolean optimization. Constraints **21**(4), 616–645 (2016). https://doi.org/10.1007/s10601-015-9237-3
18. Ramamoorthy, V., Silaghi, M.C., Matsui, T., Hirayama, K., Yokoo, M.: The design of cryptographic S-Boxes using CSPs. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 54–68. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23786-7_7. http://dl.acm.org/citation.cfm?id=2041160.2041169
19. Renauld, M., Standaert, F.-X.: Algebraic side-channel attacks. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Inscrypt 2009. LNCS, vol. 6151, pp. 393–410. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16342-5_29
20. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic side-channel attacks on the AES: why time also matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_8
21. Sun, S., et al.: Analysis of AES, skinny, and others with constraint programming. IACR Trans. Symmetric Cryptol. **2017**(1), 281–306 (2017). https://doi.org/10.13154/tosc.v2017.i1.281-306