Error-Tolerant Algebraic Side-Channel Attacks Using BEE

Ling Song[1,2], Lei Hu[1,2], Siwei Sun[1,2], Zhang Zhang[2],Danping Shi[1,2], Ronglin Hao[1,3]

[1] State Key Laboratory of Information Security,
Chinese Academy of Sciences, Beijing 100093, China
{lsong,hu,swsun,dpshi}@is.ac.cn
[2] Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
{zhangzhang}@iie.ac.cn
[3] Department of Electronic Engineering and Information Science,
University of Science and Technology of China, Hefei, 230027, China
haorl@mail.ustc.edu.cn

Abstract. Algebraic side-channel attacks are a type of side-channel analysis which can recover the secret information with a small number of samples (e.g., power traces). However, this type of side-channel analysis is sensitive to measurement errors which may make the attacks fail. In this paper, we propose a new method of algebraic side-channel attacks which considers noisy leakages as integers restricted to intervls and finds out the secret information with a constraint programming solver named BEE. To demonstrate the efficiency of this new method in algebraic side-channel attacks, we analyze some popular implementations of block ciphers—PRESENT, AES, and SIMON under the Hamming weight or Hamming distance leakage model. For AES, our method requires the least leakages compared with existing works under the same error model. For both PRESENT and SIMON, we provide the first analytical results of them under algebraic side-channel attacks in the presence of errors. To further demonstrate the wide applicability of this new method, we also extend it to cold boot attacks. In the cold boot attacks against AES, our method increases the success rate by over 25% than previous works.

Key words: algebraic side-channel attack, Hamming weight leakage, error-tolerance, cold boot attack.

1 Introduction

In recent years side-channel cryptanalysis has been an active topic of cryptanalysis, in which an attacker targets a certain implementation of a cipher and exploits the physical information during the encryption of the cipher such as computation traces and power traces. In 2002 Suresh Chari et al. proposed template attacks [9] which are usually considered as the most powerful type of side-channel attacks in an information theoretical sense, since a template attack only requires minimum traces.

Combining template attacks with algebraic cryptanalysis, Renauld and Standaert proposed algebraic side-channel cryptanalysis [25]. Their analysis composes of two stages. First, the implementation of the cipher is profiled in advance as a template, and a decoding process is devised to map a single power consumption

trace or electromagnetic trace to a vector of leaks (e.g., the Hamming weight) on the intermediate values of the encryption of the cipher. In the second stage, the cipher as well as the leaks is represented with a system of equations and the system is solved with a SAT solver, assuming the leaks derived from the first stage are accurate.

In fact, due to interference and the limitations of measurement setups, the side-channel information usually involves noise. However, the algebraic cryptanalysis is sensitive to errors and even small errors may make the key recovery attack fail, i.e., the attack finds no or wrong solutions. In the literature, there are several works that have explored error-tolerant algebraic side-channel attacks under some leakage models like Hamming weight leakage model. In [23], a cryptosystem and the corresponding Hamming weight leakages are transformed naturally into a pseudo-Boolean optimization (PBOPT) problem which was then solved with the mixed integer programming (MIP) solver SCIP [5]. Nevertheless, the method in [23] can not fully attack AES in the presence of errors. Later, an enhanced error-tolerant algebraic side-channel attack named MDASCA was introduced in [30], where the leakage is treated as a set of values rather than a single value, so that the correct leakage can be included with great confidence. In MDASCA, the cryptosystem and leakages are transformed into Conjunctive Normal Form (CNF) and then solved with a SAT solver. The MDASCA method outperforms the the SCIP-based method in [23] in dealing with errors. More techniques under a framework similar to MDASCA were discussed in [21].

Our contribution. Inspired by the idea of MDASCA that models the leakage as a set of values to trade robustness for informativeness, we propose a new method of algebraic side-channel attacks that considers noisy leakages as integers restricted to an interval and finds secret information with BEE (Ben-Gurion Equi-propagation Encoder) [20, 4]. BEE is a constraint programming solver for problems represented with Boolean variables and integer variables and provides a high-level descriptive language for use and automatically generates low-level executable CNF for the underlying SAT solver. Using this method, we analyze some popular implementations for three block ciphers—PRESENT, AES and SIMON under the Hamming weight or Hamming distance leakage model where side-channel leakages are modeled as integers (constraints for BEE). We provide the first analytical results of PRESENT and SIMON under the Hamming weight and the Hamming distance leakage model respectively in the presence of errors, and our attack against the AES is better than MDASCA and other existing attacks under the same error model with respect to the error-tolerance and leakages required.

To further show the flexibility of the new method in algebraic side-channel attacks, we extend its use in cold boot attacks [15] and also other applications where the side-channel information can be described as *constraints* for BEE. In the cold boot attacks against AES, our method increases the success rate by over 25% than previous works. Furthermore, all of our experiments are done within several seconds even in the worst case.

Organization. The paper is organized as follows. We briefly describe the process of our method of algebraic side-channel attacks using BEE in Section

2 and present algebraic side-channel attacks on two block ciphers in Section 3. In Section 4 we extend the use of our method into cold boot attacks and other attacks, and discuss the features of BEE in Section 5. Finally, we conclude the paper in the last section.

2 Algebraic Side-Channel Attacks Using BEE

In this section, we elaborate on the new method of algebraic side-channel attacks using BEE. As can be seen in Fig. 1, an attacker needs to build a system of polynomial equations from the target cipher, and then adds the side-channel information he obtained to the equation system as constraints and solves it to recover the secret key. A more detailed description is provided below.

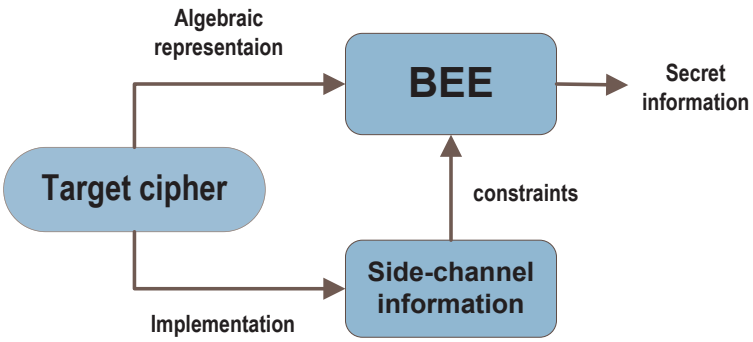

Fig. 1. Algebraic side-channel attacks using BEE

2.1 Building an equation system

Theoretically, each cipher can be represented with a system of Boolean polynomial equations which involve the bits of the cipher key, plaintext and ciphertext as unknown variables. However, since a block cipher usually iterates a nonlinear function many times (e.g., AES iterates 10 or more rounds), it is unlikely to obtain a low degree polynomial representation on the bits of the cipher key, plaintext and ciphertext. To get a system of low degree Boolean equations, intermediate variables (e.g., standing for input and output bits of the nonlinear operations in intermediate round operations) are needed. To build quadratic equations for the nonlinear layer composed of S-boxes, techniques in [10, 6, 18] can be referred.

2.2 Extracting side-channel information

Usually, it is difficult to solve the system of polynomial equations of a block cipher even when the plaintext and ciphertext are known, since there are too many unknown variables in the system [10]. However, under the circumstance of a

side-channel attack, partial information related to the key, plaintext or ciphertext may be known, e.g., physical leakage of the intermediate states in the encryption or key schedule of a block cipher, and this so-called side-channel information can be plugged into the equation system to help to solve it. Unfortunately, the side-channel information usually involves noise in practice, which may make the attack fail, i.e., the attacker finds no or wrong solutions.

To deal with the noise, it is needed to profile a device (similar to the target one) which executes the encryptions and modulates the measurement errors in advance. Once the leakages of the target device are obtained, the side-channel information can be added to the equation system in an error-tolerant way according to a priori error model. In this paper we assume that the error model has been built and the leakages can be extracted in some way. Hence, we focus on the procedure of solving the system of equations and noisy side-channel information.

2.3 Solving the system of equations and noisy side-channel information with BEE

In recent years, Boolean SAT solving techniques were improved dramatically in the field of cryptanalysis [2, 22]. A general idea of SAT-based cryptanalysis is to encode algebraic equations (usually over the binary field $\mathbb{F}_2$) into Boolean formulae in CNF and solve the transformed problem with a SAT solver. Since the efficiency of SAT solving is greatly influenced by the conversion, it is crucial to choose proper conversion methods. Truth table, Karnaugh map [17] and the methods proposed in [2] can be served as such conversions. Another way is to use a SAT-based tool which provides a high-level descriptive language for problems and automatically generates low-level executable files for the underlying SAT solver. STP [14] and BEE [20, 4] are this sort of known examples. In these tools specific techniques are used to optimize the problem solving before invoking the underlying SAT solver.

BEE is a compiler in constraint programming which facilitates users to translate problem instances involving Boolean and integral variables into CNF. The generated CNF is then solved by an underlying SAT solver such as CryptoMiniSat [27] and MiniSat [13]. A brief description of its syntax can be referred to [20]. Compared with a pure SAT solver, its ability to deal with integers and maintain the structure of the original problem instance opens a door for more complex applications.

For example, in an algebraic side-channel attack under the Hamming weight leakage model, a byte $X = x_7x_6 \cdots x_0$ with x_0 as the least significant bit and its Hamming weight can be represented in the BEE syntax as

$$\begin{aligned}&\text{new\_int}(I, w - i, w + j),\\&\text{bool\_array\_sum\_eq}([x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7], I),\end{aligned}$$

where the i and j in the first sentence are used to define the interval of integer I around the most likely value w, which includes the correct Hamming weight of X with great confidence, and the second sentence represents the Hamming weight of

X as I. With additional information of the Hamming weight, the whole equation system of the target cipher ended with the functional sentence as *solve satisfy* can be fed into BEE, and BEE returns an assignment or reports UNSAT. Other functional sentences *solve minimize w* or *solve maximize w* can also be used, which means BEE returns UNSAT or solutions that minimize or maximize the objective integer variable w. A toy example of *minimize* is shown in Appendix B to illustrate the usage of BEE.

Hence, as long as the side-channel information can be modeled as integers, i.e. constraints that speeds up the solving or narrows solution space, BEE works well. However, when the side-channel information is modeled as integers, noise should be considered, that is to say, the integers should be defined with an interval rather than a specific value in order to trade robustness for informativeness.

3 Algebraic Side-Channel Attacks under the Hamming Weight Leakage Model

In this section we explain the efficiency of our method in algebraic side-channel attacks under the Hamming weight leakage model. We take block ciphers PRESENT-80 [7] and AES-128 [11] for examples. Also, another algebraic side-channel attack of SIMON-32 [3] under Hamming distance leakage model can be found in Appendix C. In these examples we assume that the noisy Hamming weight or Hamming distance information during the encryption is obtained (no leakage during the key schedule of the cipher), and we focus on solving the corresponding equation system with our new method.

To begin with, we introduce the notion of *offset*, standing for the offset of the measured Hamming weight (or Hamming distance) from the correct one. For simplicity, we only consider three typical cases with offset 0, ± 1 and ± 2, and denote them as offset = 0, offset = 1 and offset =2. More general offset can also be analyzed. Obviously, larger offsets tolerate greater noise, so the offset is determined by the error model of the template.

3.1 PRESENT-80

PRESENT is a cipher with substitution-permutation network and with a block size of 64 bits [7]. The recommended size for the key is 80 bits, while 128-bit keys are also suggested. In this paper we just analyze PRESENT-80, the version with 80-bit keys.

The encryption of PRESENT-80 is composed of 31 rounds, each of which consists of an XOR operation to introduce a round key K_i for $1 \leq i \leq 31$, a linear bitwise permutation and a nonlinear substitution layer which parallelly applies a 4-bit S-box 16 times. An additional round key K_{32} is used for post-whitening.

Both the encryption and key schedule of PRESENT-80 are simple with respect to algebraic representations. In our attack, each S-box is described with four equations as shown in Appendix A. Since the algebraic representations of the S-boxes are the most complex part of the equation system, we introduce new

variables for the input and output bits of each S-box to get low degree polynomial equations. In this way we get a system of 4276 equations in 4292 variables.

For an implementation of PRESENT-80 in a PIC 16F877 8-bit RISC-based microcontroller, a measurement setup described in [28] can be exploited to extract a power consumption that strongly correlates with the Hamming weight of the data manipulated. This is also the model used in [25] where the Hamming weight of the data commuting on the bus is indicated to be recovered with a probability of 0.986. In this implementation, the whole encryption of a single plaintext leaks the Hamming weight information corresponding to the computation of $2 \times 8 \times 31$ bytes.

Let us first consider the case that the accurate Hamming weight information can be recovered, and then move on to other cases of inaccurate Hamming weight information. All of our experiments are conducted with MiniSat as the underlying solver of BEE on a PC with 3.4GHz CPU (only one core is used) and 4 GB Memory. All solving times are given in seconds and are averaged over on 50 random instances.

For accurate Hamming weight information, it can be added to the equation system as constraints in the way described in Section 2 with $i = j = 0$. Following [25], we consider four different attack scenarios according to known/unknown plaintext-ciphertext pairs and consecutive/random Hamming weight leakages. Table 1 lists the solving times using one trace compared with [25] when a 100% success rate is reached, where "#rounds" means the number of rounds which are observed in the side-channel information, and the leakage rate means the ratio between the number of leaked bytes and the number of all bytes in the measured rounds. Thus, 50% indicates random leakages and 100% infers to consecutive leakages.

Table 1. Experimental results of PRESENT-80 with a single trace when offset = 0. Experiments in [25] were performed on an Intel-based server with a Xeon E5420 processor cadenced at 2.5GHz running a linux 32-bit 2.6 Kernel.

Scenarios	leakage rate	[25]		this paper	
		#rounds	time (s)	#rounds	time (s)
known P/C	100%	8	79.69	2.5	0.21
known P/C	50%	18	117.10	5	104.31
unknown P/C	100%	8	45.59	4.5	92.55
unknown P/C	50%	26	214.12	> 8	> 3600

According to Table 1, our attack requires less leakages than [25]. However, under the unknown plaintext and ciphertext scenario with random leakages, our attack fails and the experiments ran overtime.

Next, we consider the cases where the Hamming weight leakages are noisy with offset = 1 and offset =2. Table 2 exhibits the experimental results under known plaintext/cihertext scenario. If the leakages are consecutive, two traces are enough to retrieve the cipher key within an hour even when the offset is ± 2. For the case that offset = 2 and the leakage rate is 0.7, three traces are required.

For unknown plaintext/ciphertest scenario, it takes a very long time to return a solution.

Table 2. Experimental results of PRESENT-80 when offset = 1,2

Scenarios	offset	leakage rate	#rounds	#traces	time (s)
known P/C	1	100%	4	2	6.86
known P/C	1	50%	4	2	39.40
known P/C	2	100%	5	2	3084.39
known P/C	2	70%	4	3	33.27

3.2 AES-128

The block cipher AES-128 [11] accepts 128-bit blocks and 128-bit keys and iterates 10 rounds. Each round, except the last one, consists of four operations—AddRoundKey, SubByte, ShiftRow and MixColumn. The last round omits the MixColumn operation. The SubByte operation applies an 8-bit S-box 16 times in parallel and is the only nonlinear operation in AES.

In order to represent AES as a system of low degree Boolean polynomial equations, we introduce new variables for both the input bits and output bits of SubByte and apply the technique proposed in [10] to describe the S-box with 23 quadratic equations. Then the key schedule algorithm and the encryption of a plaintext can be represented with a system of 6296 equations in 3168 variables.

For an 8-bit PIC microcontroller implementation of AES, there are 84 Hamming weight leakages in each round corresponding to 16 weights in AddRoundKey, 16 weights in SubBytes and $4 \cdot 13$ weights in MixColumn [26, 23, 30, 21]. The work in [26] is the first algebraic side-channel attack against AES, which showed how the keys can be recovered from a single measurement trace, provided that the attacker can identify the correct Hamming weight leakages of several intermediate computations during the encryption process. Later, Zhao el al. proposed an enhanced method named MDASCA to deal with the noise in measurement using a set rather than a single value to describe the Hamming weight [30]. More elaborate techniques in a framework similar to MDASCA were discussed in [21]. Recently, a novel method that models the cipher and the template as a graph and finds the most possible key with a decoding algorithm from low-density parity check codes was proposed in [32].

Following the notations in the previous subsections, we give our experimental results on AES comparing with two latest works [21, 30]. The experiments are carried out with CryptoMiniSat as the underlying solver of BEE on a PC with 3.4GHz CPU (only one core is used) and 4 GB Memory. The results are summarized in Table 3. It shows the method in [30] outperforms that of [21], while our attacks push the results of [30] further. Specifically, for offset = 1 our method can recover the secret key of AES using a single trace, less than what is needed in [30]. For a greater offset, offset = 2, only three traces are needed. Note that method in [32] also provides good results. In [32], the noise level is

parameterized with signal-to-noise ratio, which makes it difficult to compare our results with that of [32].

Table 3. Experimental results of AES-128. The symbol "-" indicates that no related information was provided. The experiments of [30] were ran on an AMD Athlon 64 Dual core 3600+ processor clocked at 2.0 GHz.

Scenario	offset	leakage rate	[25]		[30]		this paper	
			#rounds	time(s)	#rounds	time(s)	#rounds	time(s)
known P/C	0	100%	2	-	1	10	1	2.74
known P/C	0	50%	10	-	5	120	3	118.00
unknown P/C	0	100%	5	-	2	10	2	6.75
unknown P/C	0	50%	not clear	-	6	100	6	69.17
known P/C	1	100%	-	-	2(2 traces)	120	2(2 traces) 3	37.11 974.67
known P/C	2	100%	-	-	-	-	1(3 traces)	33.29

4 Cold Boot Attacks and Other Applications

In general, we can feed into BEE any problem instance that can be solved by a SAT solver. In fact, BEE was designed for problems that are characterized by certain constraints, for example algebraic side-channel attacks under the Hamming weight leakage model. Besides this, BEE can also be applied in other algebraic attacks like cold boot attacks [15].

4.1 Cold boot attacks against AES-128

Cold boot attacks were first proposed in [15]. In a cold boot attack, an attacker tries to recover the cryptographic key from DRAM based on the fact that the data may persist in memory for several minutes after removal of power by reducing the temperature of memory. For block ciphers, a cold boot attack is to recover the secret key from an observed set of the round keys in memory, which are disturbed by errors due to memory bit decay. From the point of view of algebraic analysis, the cold boot problem can be modeled as solving a polynomial system $\{f_1, f_2, \cdots, f_m\}$ with noise.

There are different methods proposed to tackle the cold boot problem [1, 16]. In [1] a system of equations derived from the cold boot attack is solved with a mixed integer programming solver SCIP [5]. Specifically, for each f_i a new Boolean variable e_i is added, and then the system is converted to a mixed integer programming problem with Σe_i as the objective function. Hence, the cold boot problem turns to be a maximum satisfiability problem, which may also be solved with another method named ISBS [16]. The basic idea of ISBS is to solve the polynomial system $\{f_1 + e_1, f_2 + e_2, \cdots, f_m + e_m\}$ with the characteristic set method [29] by searching all $\{e_1, e_2, \cdots, e_m\}$. The solution with $\{e_1, e_2, \cdots, e_m\}$ achieving the smallest Hamming weight is the target solution.

Following the ideas of [1] and [16], BEE can be used instead in the following way. First, represent the polynomial system $\{f_1 + e_1, f_2 + e_2, \cdots, f_m + e_m\}$ with

the BEE syntax, and introduce a new integer variable, say w, to describe the Hamming weight of $\{e_1, e_2, \cdots, e_m\}$, and then solve the problem with the BEE functional sentence as *solve minimize w*.

Before experimentally verifying the efficiency of BEE for solving cold boot problems, the bit decay model should be illuminated first. According to [15], bit decay in DRAM is usually asymmetric: bit flips $0 \to 1$ and $1 \to 0$ occur with different probabilities, depending on the ground state. Consider an efficiently computable vectorial Boolean function $\mathcal{KS} : \mathbb{F}_2^n \to \mathbb{F}_2^N$ where $N > n$, and two real numbers $0 \leq \delta_0, \delta_1 \leq 1$. Let $K = \mathcal{KS}(k)$ be the image for some $k \in \mathbb{F}_2^n$, and K_i be the i-th bit of K. Given K, we compute $K' = (K'_0, K'_1, \cdots, K'_{N-1}) \in \mathbb{F}_2^N$ according to the following probability distribution:

$$Pr[K'_i = 0|K_i = 0] = 1 - \delta_1, Pr[K'_i = 1|K_i = 0] = \delta_1,$$
$$Pr[K'_i = 1|K_i = 1] = 1 - \delta_0, Pr[K'_i = 0|K_i = 1] = \delta_0.$$

Therefore, K' can be considered as a noisy version of $K = \mathcal{KS}(k)$ for some unknown $k \in \mathbb{F}_2^n$, where the probability of a bit 1 in K flipping to 0 is δ_0 and the probability of a bit 0 in K flipping to 1 is δ_1.

In our experiments, the target is the key schedule of AES-128 and the same algebraic representation of the key schedule is used as in Subsection 3.2. We set δ_1 to be 0.001 to generate the K' as in [1, 16, 31] and assume $\delta_1 = 0$ for solving the cold boot problem. Figure 2 shows the experimental success rate of key recovery of AES for different δ_0, compared with methods from [1, 16] denoted as SCIP and ISBS respectively. It is manifest that BEE provides an increase over 25% in success rate. The details of running time can be found in Appendix C which shows that BEE solves the problem in several seconds even when δ_0 increases to 0.5. This may be explained by the low nonlinearity of the key schedule of AES-128 and the optimization of BEE. The experiments also include the case where BEE simply takes the cold boot problem as a satisfiability problem denoted as "BEE satisfiability". If we use information of all rounds of the key schedule, this method almost corresponds to the method in [31] and the key can be recovered in 1.3 hours even when $\delta_0 = 0.8$.

4.2 Side-Channel Cube Attacks

Cube attacks were proposed by Dinur and Shamir at Eurocrypt 2009 [12]. It is a type of generic key recovery attacks applicable to any cryptosystem whose ciphertext bit can be represented by a low degree multivariate polynomial in the secret and public variables. The aim of a cube attack is to derive a system of linear or quadratic equations with the secret key bits as unknowns which can be solved easily. However, block ciphers tend to resist against cube attacks, since they iteratively apply a nonlinear round function a large number of times such that it is unlikely to obtain a low degree polynomial to represent any ciphertext bit.

Fortunately, in a side-channel attack where some intermediate variables leak, cube attacks are still useful [19]. Due to the side-channel noise, the derived equation system may contain errors, which resembles the situation in a cold

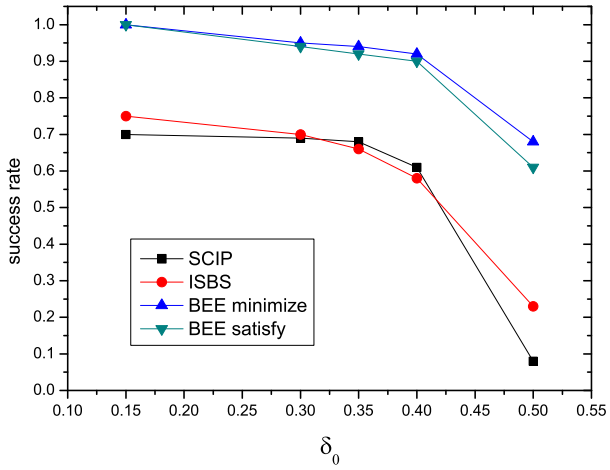

Fig. 2. Success rate of key recovery considering 4 rounds of key schedule output

boot attack. As a consequence, it is likely that our method can also be applied efficiently in side-channel cube attacks against block ciphers.

5 Discussion

We have introduced a new method of algebraic side-channel attacks that considers the noisy leakage as an integer restricted to an interval and finds secret information with BEE, a SAT-based constraint programming solver. We exemplified the efficiency of our method by analyzing some popular implementations of three block ciphers—PRESENT, AES and SIMON under the Hamming weight or Hamming distance leakage model. In addition we applied our method to cold boot attacks against AES.

As a new SAT-based constraint solver, BEE can be added to the tool set for side-channel attacks. In the literature, SAT solvers usually outperform MIP solvers in cryptanalysis. For instance, an MIP solver determines the initial state of Bivium B [24] in $2^{63.7}$ seconds [8] while the MiniSat takes only $2^{42.7}$ seconds. For side-channel attacks in the presence of errors as mentioned above, the SAT-based BEE works better than the methods based on MIP solvers. It is likely that BEE has its own advantage for certain algebraic attacks. The feathers of our BEE-based method are summarized below.

- With the side-channel information modeled as constraints, BEE shows a widespread use in algebraic side-channel attacks, for example, side-channel attacks under Hamming weight leakage model, cold boot attacks and cube attacks.
- BEE, as a SAT-based tool, may outperform the tools based on MIP solvers in many classes of algebraic attacks.

– It is natural for use, can simplify constraints and optimizes the resulted CNF automatically.
– Algebraic side-channel attacks are more efficient than classical side-channel attacks like differential power attack (DPA) in respect of the number of traces needed.

There is also a limitation of BEE that the integers in BEE are defined with an interval, not a set. In the cases that the most likely values for some integer variables are not continuous, our method may bear no advantage over MDASCA, though BEE realizes optimizations in itself.

6 Conclusion

In this paper we introduced a new method of algebraic side-channel attacks that treats the noisy leakage as an integer restricted to an interval and finds secret information with a constraint programming solver BEE. We showed that this method is efficient and flexible.

We analyzed some popular implementations of PRESENT, AES and SIMON under the Hamming weight or Hamming distance leakage model to illustrate the efficiency of constraint programming in cryptanalysis. The results on AES are better than previous ones under the same error model and we provided the first analytical results on PRESENT and SIMON under the Hamming weight and Hamming distance leakage model respectively in the presence of errors.

To further show the flexibility of BEE in algebraic side-channel attacks, we extended its use in cold boot attacks. In the cold boot attacks against AES, our method provides an increase over 25% in success rate and all of our attack experiments were done within seconds. It is also likely to apply our method in cube attacks and it is a future work to verify its efficiency experimentally in new applications.

References

1. Albrecht, M., Cid, C.: *Cold Boot Key Recovery by Solving Polynomial Systems with Noise.* In: Lopez, J., Tsudik, G. (eds.): ACNS 2011. LNCS, vol. 6715, pp. 57-72. Springer, Heidelberg (2011)
2. Bard, G., Courtois, N., Jefferson, C.: *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomial over GF(2) via SAT-Solvers.* In: IACR Cryptology ePrint Archive, number 024 (2007)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404 (2013), http://eprint.iacr.org/
4. BEE. http://amit.metodi.me/research/bee/.
5. Berthold, T., Heinz, S., Pfetsch, M.E., Winkler, M.: *SCIP C Solving Constraint Integer Programs.* In: SAT 2009 competitive events booklet (2009)
6. Biryukov, A., De Canni‘ere, C.: *Block Ciphers and Systems of Quadratic Equations.* In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 274-289. Springer, Heidelberg (2003)
7. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: *PRESENT: An Ultra-Lightweight Block Cipher.* In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450-466. Springer, Heidelberg (2007)

8. Borghoff, J., Knudsen, L.R., Stolpe, M.: *Bivium as a Mixed-Integer Linear Programming Problem.* In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 133-152. Springer, Heidelberg (2009)
9. Chari, S., Rao, J.R., Rohatgi, P.: *Template Attacks.* In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (ed.) CHES 2002. LNCS, vol. 2523, pp. 13-28. Springer, Heidelberg (2003)
10. Courtois, N., Pieprzyk, J.: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations.* In: Zheng, Y. (ed.) Asiacrypt 2002, LNCS, vol. 2501, pp. 267-287. Springer, Heidelberg (2002)
11. Daemen, J., Rijmen, V.: *AES Proposal: Rijndael.* (1998)
12. Dinur, I., Shamir, A.: *Cube Attacks on Tweakable Black Box Polynomials.* In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278-299. Springer, Heidelberg (2009)
13. Een, N., Sorensson, N.: *MiniSat v1.13 - A SAT Solver with Conflict-Clause Minimization.* Available at `http://minisatse/Papers.html`. (2005)
14. Ganesh, V., Dill, D.L.: *A Decision Procedure for Bbit-vectors and Arrays.* In: Damm, W., Hermanns, H. (ed.) CAV 2007. LNCS, vol. 4590, pp. 524-536. Springer, Heidelberg (2007)
15. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: *Lest We Remember: Cold Boot Attacks on Encryption Keys.* In: USENIX Security Symposium, pp. 45-60. USENIX Association (2009)
16. Huang, Z., Lin, D.: *A New Method for Solving Polynomial Systems with Noise over F2 and Its Applications in Cold Boot Key Recovery.* In: Knudsen, L.R., Wu, H.(ed.) SAC 2012. LNCS, vol. 7707, pp. 16-33. Springer, Heidelberg (2013)
17. Karnaugh, M.: *The Map Method for Synthesis of Combinational Logic Circuits.* AlEE Transactions on Communications and Electronics, 72(9), 593-599 (1953)
18. Knudsen, L.R. Miolane, C.V.: *Counting Equations in Algebraic Attacks on Block Ciphers.* International Journal of Information Security 9(2), 127-135 (2010)
19. Li, Z., Zhang, B., Fan, J., Verbauwhede, I.: *A New Model for Error-Tolerant Side-Channel Cube Attacks.* In: Bertoni, G., Coron, J.-S. (ed.) CHES 2013. LNCS, vol. 8086, pp. 453-470. Springer, Heidelberg (2013)
20. Metodi, A., Codish, M.: *Compiling Finite Domain Constraints to SAT with BEE.* In: TPLP, 12(4-5), 465-483. (2012)
21. Mohamed, M.S.E., Bulygin, S., Zohner, M., Heuser, A., Walter, M., Buchmann, J.: *Improved Algebraic Side-Channel Attack on AES.* In: IEEE HOST 2012, pp. 146-151 (2012)
22. Mouha, N., Preneel, B.: *Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20.* In: IACR Cryptology ePrint Archive, number 328 (2013)
23. Oren, Y., Kirschbaum, M., Popp, T., Wool, A.: *Algebraic Side-Channel Analysis in the Presence of Errors.* In: Mangard, S., Standaert, F.-X. (ed.) CHES 2010. LNCS, vol. 6225, pp. 428-442. Springer, Heidelberg (2010)
24. Raddum, H.: *Cryptanalytic results on Trivium.* eSTREAM report 2006/039 (2006), http://www.ecrypt.eu.org/stream/triviump3.html
25. Renauld, M., Standaert, F.-X.: *Algebraic Side-Channel Attacks.* In: Bao, F., Yung, M., Lin, D., Jing, J. (ed.) Inscrypt 2009. LNCS, vol. 6151, pp. 393-410. Springer, Heidelberg (2010)
26. 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. (ed.) CHES 2009. LNCS, vol. 5747, pp. 97-111. Springer, Heidelberg (2009)
27. Soos, M., Nohl, K., Castelluccia, C.: *Extending SAT Solvers to Cryptographic Problems.* In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244-257. Springer, Heidelberg (2009)
28. Standaert,F.-X., Archambeau, C.: *Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages.* In: Oswald, E., Rohatgi, P. (ed.) CHES 2008. LNCS, vol. 5154, pp. 411-425. Springer, Heidelberg(2008)
29. Wu,W.T.: *Basic Principles of Mechanical Theorem-proving in Elementary Geometries.* In: Journal Automated Reasoning 2, 221-252 (1986)
30. Zhao, X., Zhang, F., Guo, S., Wang, T., Shi, Z., Liu, H., Ji, K.: *MDASCA: An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation.* In: Schindler,W., Huss, S. (ed.) COSADE 2012. LNCS, vol. 7275, pp. 231-248. Springer, Heidelberg (2012)

31. Kamal, A.A., Youssef, A.M.: *Applications of SAT Solvers to AES key Recovery from Decayed Key Schedule Images.* In: Proceedings of the Fourth International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2010, Venice/Mestre, Italy, July 18-25 (2010)
32. Veyrat-Charvilon, N., Gérard, B., Standaert, F.-X.: *Soft Analytical Side-Channel Attacks.* Cryptology ePrint Archive, Report 2014/410 (2014), http://eprint.iacr.org/

A Equations of the S-box in PRESENT

Suppose the input nibble of the 4-bit S-box in PRESENT is $X = x_3x_2x_1x_0$ and the output nibble is $Y = y_3y_2y_1y_0$. Then four Boolean polynomial equations on x_is and y_js are as follows.

$$\begin{aligned}
x_0 + x_2 + x_1x_2 + x_3 + y_0 &= 0,\\
x_1 + x_0x_1x_2 + x_3 + x_1x_3 + x_0x_1x_3 + x_2x_3 + x_0x_2x_3 + y_1 &= 0,\\
1 + x_0x_1 + x_2 + x_3 + x_0x_3 + x_1x_3 + x_0x_1x_3 + x_0x_2x_3 + y_2 &= 0,\\
1 + x_0 + x_1 + x_1x_2 + x_0x_1x_2 + x_3 + x_0x_1x_3 + x_0x_2x_3 + y_3 &= 0.
\end{aligned}$$

B A Toy Example of Using BEE

Suppose we get a system of 10 linear equations in 4 Boolean variables x_0, x_1, x_2 and x_3 as follow. However, these equations include disturbance. In order to describe the equations accurately, we introduce an error variable for each equation, and then the system involves 13 variables, which is more than the number of equations. Assume that the error rate is below 0.5, say 0.4. Then this knowledge can be used to solve the system.

$$\begin{cases}
x_0 + x_1 = 1\\
x_0 + x_2 = 0\\
x_0 + x_3 = 0\\
x_1 + x_2 = 1\\
x_1 + x_3 = 0\\
x_2 + x_3 = 0\\
x_1 + x_2 = x_3 = 1\\
x_0 + x_1 + x_3 = 1\\
x_0 + x_1 + x_2 = 0\\
x_0 + x_1 + x_2 + x_3 = 0
\end{cases}
\Longrightarrow
\begin{cases}
x_0 + x_1 + e_0 = 1\\
x_0 + x_2 + e_1 = 0\\
x_0 + x_3 + e_2 = 0\\
x_1 + x_2 + e_3 = 1\\
x_1 + x_3 + e_4 = 0\\
x_2 + x_3 + e_5 = 0\\
x_1 + x_2 + x_3 + e_6 = 1\\
x_0 + x_1 + x_3 + e_7 = 1\\
x_0 + x_1 + x_2 + e_8 = 0\\
x_0 + x_1 + x_2 + x_3 + e_9 = 0
\end{cases}$$

If the error rate is below 0.4, then the Hamming weight of $(e_0, e_1, \cdots, e_9)$ is less or equal to 5. Using BEE this problem can be represented with three types of sentences as below.

1. new_bool(x) and new_int(w, i, j), used to define a Boolean variable x or an integer variable w with interval from i to j;
2. bool_array_xor($[a, b, c, d]$), used to describe an XOR clause $a + b + c + d = 1$;

3. bool_array_sum_eq($[a, b, c, d], w$), used to represent the Hamming weight of (a, b, c, d) with integer w.

After rewriting the equation system with these three types of sentences, we can feed it into BEE directly with functional sentence being *solve minimize* w where w is defined as the Hamming weight of $(e_0, e_1, \cdots, e_9)$. Then BEE returns solutions that satisfy the constraints. In this case the solutions are as follows.

Answer 1:

$x_0 = 1 \quad x_1 = 0 \quad x_2 = 1 \quad x_3 = 0$
$e_0 = 0 \quad e_1 = 0 \quad e_2 = 1 \quad e_3 = 0 \quad e_4 = 0 \quad e_5 = 1 \quad e_6 = 0 \quad e_7 = 0 \quad e_8 = 0 \quad e_9 = 0$
$w = 2$

Answer 2:

$x_0 = 0 \quad x_1 = 1 \quad x_2 = 0 \quad x_3 = 0$
$e_0 = 0 \quad e_1 = 0 \quad e_2 = 0 \quad e_3 = 0 \quad e_4 = 1 \quad e_5 = 0 \quad e_6 = 0 \quad e_7 = 0 \quad e_8 = 1 \quad e_9 = 1$
$w = 3$

C Algebraic Side-Channel Attack of SIMON-32 under Hamming Distance Leakage Model

SIMON is a family of lightweight block ciphers designed by researchers from the National Security Agency (NSA) of the USA to provide an optimal hardware implementation performance [3]. SIMON comes in a variety of block sizes and key sizes. In this paper we analyze its smallest version, SIMON-32, which accepts 64-bit keys and 32-bit blocks and iterates 32 rounds.

The design of SIMON follows a classical Feistel structure, operating on two n-bit halves in each round. For SIMON-32, n is 16. The round function makes use of three n-bit operations: XOR ($\oplus$), AND ($\&$) and circular shift ($\lll$), and given a round key k it is defined on two inputs x and y as

$$R_k(x, y) = (y \oplus f(x) \oplus k, x),$$

where $f(x) = ((x \lll 1) \& (x \lll 8)) \oplus (x \lll 2)$. The key schedule algorithm of SIMON is a linear transformation.

The algebraic representation of SIMON is intuitive due to the simplicity of the round function. We introduce new variables for the intermediate state after each round operation in both the encryption and key schedule algorithm. Consequently, a system of 960 equations in 992 variables is built from the encryption of a single plaintext and the key scheduling.

A typical implementation of SIMON is based on ASIC [3] which leaks the Hamming distance between the input and the output of the round function. The Hamming distance varies from 0 to 16. Following the attacks on PRESENT and AES, we consider the measured Hamming distance, compared with the correct one, to be with a small offset offset = 0,1,2.

Table 4 exhibits our results of experiments running on a PC with 2.83GHz CPU (only one core is used) and 3.25GB Memory with CryptoMiniSat as the underlying solver of BEE, where all solving times are given in seconds and are averaged over 50 random instances. Our attacks are performed under the known plaintext or known ciphertext scenario and there is no difference between them. For offset=0, the BEE works seriously overtime with only one trace and 5 traces are advisable to recover the key within a proper time. For offset=2, as many as 13 traces are required. It is observed that the Hamming distances leaked near the known plaintext or ciphertext is more useful than the ones in the middle for solving the system in a reasonable time. Note that more traces are required for attacking SIMON-32 than for AES or PRESENT. This is because the Hamming distance leakages on 16-bit states provide less information than that on 8-bit states.

Table 4. Experimental results of SIMON-32

offset	#traces	#rounds	time (s)
0	5	6	36.32
1	11	7	386.34
2	13	8	320.11

D Running Time of Cold Boot Attack against AES-128

In this appendix the running time of the cold boot attack against AES-128 is shown in Table 5, where "SCIP" stands for the method in [1]; "ISBS" represents the method in [16]; "BEE min." and "BEE sat." are methods in this paper and short for "BEE minimize" and "BEE satisfiability" respectively. Our experiments are carried out with CryptoMiniSat as the underlying solver of BEE on a PC with 3.4GHz CPU (only one core is used) and 4 GB Memory. Experiments of "SCIP" and "ISBS" are conducted on PCs with 2.6 Ghz and 2.8 Ghz respectively.

Table 5. The running time of key recovery considering 4 rounds of key schedule output of AES-128 (in seconds)

δ_0	Method	limit t	min t	avg. t	max t
0.15	SCIP	60	1.78	11.77	59.16
	ISBS	60	0.002	0.07	0.15
	BEE min.	60	1.29	2.02	2.36
	BEE sat.	60	1.25	1.94	2.33
0.30	SCIP	3600	4.86	117.68	719.99
	ISBS	3600	0.002	0.14	2.38
	BEE min.	60	1.34	2.15	2.64
	BEE sat.	60	1.34	2.27	2.55
0.35	SCIP	3600	4.45	207.07	1639.55
	ISBS	3600	0.002	0.27	7.87
	BEE min.	60	1.41	2.10	2.76
	BEE sat.	60	1.42	2.34	2.59
0.40	SCIP	3600	4.97	481.99	3600
	ISBS	3600	0.002	0.84	20.30
	BEE min.	60	1.48	2.44	3.34
	BEE sat.	60	1.44	2.43	3.11
0.50	SCIP	3600	6.57	3074.36	3600
	ISBS	3600	0.002	772.02	3600
	BEE min.	60	1.47	1.94	4.18
	BEE sat.	60	1.62	2.82	3.94