SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition

Ashutosh Dhar Dwivedi[1], Miloš Klouček[4], Paweł Morawiecki[1], Ivica Nikolić[3], Josef Pieprzyk[1,2] and Sebastian Wójtowicz[1]

[1] Institute of Computer Science, Polish Academy of Sciences, Poland
[2] Electrical Engineering and Computer Science School, Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia
[3] Nanyang Technological University, Singapore
[4] Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic

Abstract.
We investigate six authenticated encryption schemes (ACORN, ASCON-128a, Ketje Jr, ICEPOLE-128a, MORUS, and NORX-32) from the CAESAR competition. We aim at state recovery attacks using a SAT solver as a main tool. Our analysis reveals that these schemes, as submitted to CAESAR, provide strong resistance against SAT-based state recoveries. To shed a light on their security margins, we also analyse modified versions of these algorithms, including round-reduced variants and versions with higher security claims. Our attacks on such variants require only a few known plaintext-ciphertext pairs and small memory requirements (to run the SAT solver), whereas time complexity varies from very practical (few seconds on a desktop PC) to 'theoretical' attacks.

Keywords: SAT solvers, SAT-based cryptanalysis, logic cryptanalysis, authenticated encryption, CAESAR

1 Introduction

A cryptographic algorithm which provides both confidentiality and authenticity, is called an authenticated encryption (AE) or simply an authenticated cipher. It encrypts and authenticates messages using both a secret key (shared by the sender and the receiver) as well as a public number (called a nonce). AE algorithms are often built as various combinations of block ciphers, stream ciphers, message-authentication codes and hash functions. Several solutions have been standardised by ISO/IEC and one of the most widely used is AES-GCM [Nat07], which is an authenticated cipher based on the Advanced Encryption Standard (AES) [AES01].

In many modern applications, performance and security requirements are set so high that the current AE standards established a decade ago struggle to meet them. A good such example is VMWare View, which is a remote desktop protocol supported by low-cost terminals from various manufacturers. The VMWare 2010 documentation recommends switching from AES-GCM to a faster cipher for "the best user experience". Additionally, the most efficient hardware implementation of AES-GCM does not reach the full potential due to the bottleneck caused by slow multiplication in Galois fields. Apart from efficiency issues, there are also security problems that seem to plague legacy AE algorithms.

The great interest and importance of AE have been manifested by the announcement of a new public call for AE algorithms — the CAESAR competition [CAE]. The contest has started in 2014 and has received worldwide attention. In the first round, 57 algorithms

were submitted and now (October 2016) in the third round, 16 ciphers are still in the race. The competition is planned to finish by the end of 2017 and "will identify a portfolio of authenticated ciphers that (1) offer advantages over AES-GCM and (2) are suitable for widespread adoption".

The primary concern for all AE schemes is a sound security evaluation. As a rule, the initial evaluation has been given by the designers of the schemes, while possible improvements by third-party cryptanalysis. Cryptanalysis of AE schemes is particularly demanding as the attacker works with more constraints than in a typical cryptanalysis framework for block ciphers or hash functions.

Our contribution.

We investigate the resistances of a chosen set of CAESAR schemes against state recovery attacks. Our analysis is based on SAT solvers and falls into the so-called logic cryptanalysis. A state recovery attack may pose a real threat to the scheme as the known state often allows to forge tags or even to recover the secret key. We follow a two-stage approach. First, we construct a SAT problem that corresponds to the state recovery of a scheme. Usually, we need only a handful of ciphertext blocks and the analysis is in the known plaintext framework. Then, we run a SAT solver to find a solution of the problem. We use `lingeling` or its parallel variant `plingeling` [Bie16] — one of the best SAT solvers according to the latest SAT competitions. For each problem, we typically run the solver on 8 cores for a few days.

We analyse six CAESAR schemes, namely ACORN [Wu], ASCON-128a [DEMS], ICEPOLE-128a [MGH+14], Ketje Jr [BDP+], MORUS [WH], and NORX-32 [AJN14b]. All except ICEPOLE-128a are the third-round candidates[1] from the CAESAR competition. We omit the CAESAR candidates with large, 8-bit S-boxes[2] as the algorithms with big and complex S-boxes are known to be very difficult for SAT solvers and give very little hope for a successful attack.

Our analysis reveals that *all the algorithms (their full variants, as submitted to the CAESAR competition) are resistant against SAT-based state recovery.* That is, attacking these schemes with SAT solver is infeasible even when one is given impractically large time (but not more than the claimed security level). Next we focus on approximating their security margin, and thus we consider weakened versions of the schemes. We do this by either reducing the number of rounds or by simplifying the round function. Alternatively, we artificially increase the security level expected from ciphers and use it to benchmark our attacks. In Table 1, we summarize our main results. For instance, we are able to launch a state recovery against ICEPOLE-128a when it uses 4 rounds (instead of the original 6). Similarly, we can recover the state of Ketje Jr, if the claimed security level is higher than 165 bits (in the original submission, the claimed security level is 96 bits).

Table 1: SAT-based state recovery

Cipher	Key size	Rounds	Complexity	Reference
ICEPOLE-128a	128 (128)	4 (6)	2^{108}	Section 3.2
ASCON-128a	128 (128)	2 (8)	2^{32}	Section 3.3
NORX-32	128 (128)	1.5 (4)	2^{96}	Section 3.4
Ketje Jr	>165 (96)	full	2^{165}	Section 3.1
MORUS	>370 (128)	full	2^{370}	Section 4.1

[1] Note, we have started the analysis before the announcement of the third-round candidates and then ICEPOLE-128a was still in the race.

[2] Recall that several CAESAR candidates are based on AES (which uses 8-bit S-boxes)

Related work

As a tool, SAT solvers have been used in the CAESAR competition. Lafitte et al.[LLMH16] perform SAT-based state recovery of ACORN, but end up with a state recovery which requires higher complexity than a simple brute force attack. Stoffelen [Sto16] used SAT solvers to find optimized S-box implementations for several proposals including ASCON, ICEPOLE, Ketje and many others. An interesting work is a new automated tool for finding linear characteristics, where some developed heuristics are inspired by modern SAT solvers [DEM15b]. The tool was applied to some schemes from the CAESAR competition. Note that references regarding the analysed ciphers are given right after a description of a given cipher (Section 3).

2 Preliminaries

2.1 Sponge-based Authenticated Encryption Schemes

Bertoni et al. introduced the sponge function in [BDPV]. The sponge function and its sister construction called the duplex construction [BDPV11] can be used to design a wide range of cryptographic algorithms including authenticated encryption schemes. The duplex construction is used for a relatively large group of algorithms submitted to the CAESAR competition. We analyse four such algorithms: ASCON, ICEPOLE, NORX and Ketje.

The duplex construction is based on a fixed permutation (or transformation) determined by the following two parameters: bitrate r and capacity c. The sum of the two parameters gives the input/output length or the state size. For a fixed state size, different values for bitrate and capacity provide trade-offs between speed and security. A higher bitrate gives a faster algorithm with lower security and vice versa. The duplex construction absorbs input blocks (plaintext) into the bitrate part of the permutation f. If an input block is smaller than r bits, it gets padded to the full r bits. After processing by the permutation f, r-bit output blocks (ciphertext) are squeezed out. Note that the capacity part of the state is never directly manipulated, neither for absorbing nor for squeezing.

Authentication encryption with associated data (AEAD) can be realized using the duplex construction. Figure 1 shows a typical application. First, a secret key K and a nonce is absorbed into the initial state and the permutation f is called. The following steps are iterations performed for pairs: plaintext P_i and ciphertext C_i, where $i = 0, \ldots, n$. For the plaintext P_i, the ciphertext C_i is computed by XOR-ing the plaintext with the bitrate part squeezed out from the state. After P_i is absorbed, the bitrate part becomes C_i. The permutation f is called again and the next iteration proceeds. Besides plaintext blocks, public associated data blocks can be processed by absorbing them without encryption. The encryption is completed by squeezing a r-bit authentication tag T.

The analysed AE algorithms are different although they preserve the general duplex construction. Their security heavily depends on the internal permutation f, typically implemented as a sequence of elementary operations called rounds. It is expected that if the permutation f applies more rounds, then the cryptanalysis becomes more complex and the relevant AE becomes more secure (but also slower).

2.2 CICO problem

In [BDPV] it was shown that the constrained-input constrained-output (CICO) problem is essential for security of cryptographic sponges and primitives, which are built upon them. The problem can be stated as follows. We are given a function f (e.g. a permutation), which maps input bits to output bits. Some bits are fixed and a problem is to determine unknown parts of input and output states, such as the input-output mapping is valid. A generalization of the CICO problem is the multi-block CICO problem, where we deal

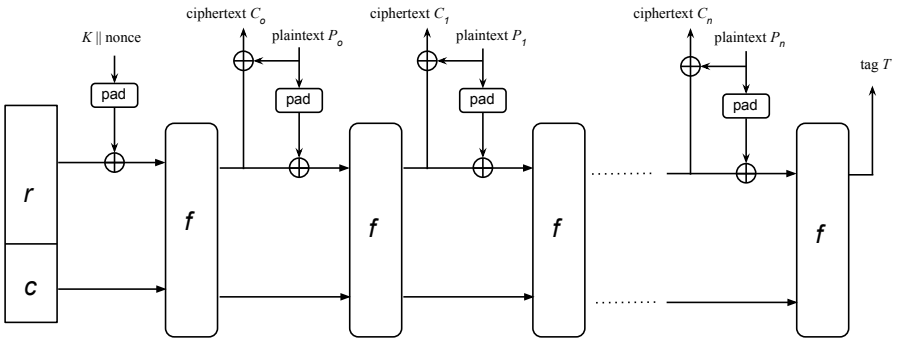

Figure 1: Authentication encryption scheme based on the duplex construction

with a number of f calls and states between them. (See Figure 3.) Again, our task is to determine unknowns.

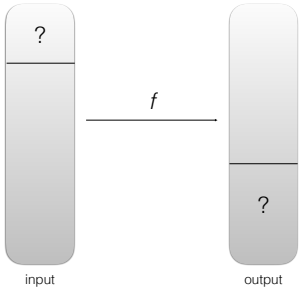

Figure 2: CICO problem

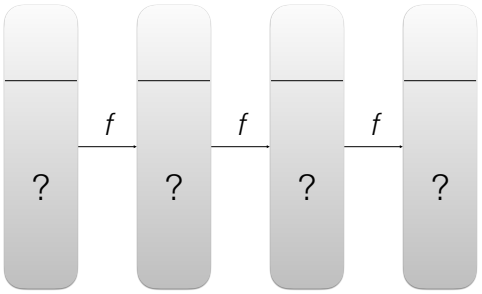

Figure 3: Multi-block CICO problem

Let us now see how the CICO problem refers to the sponge-based authenticated cipher shown in Figure 1. If we know plaintext-ciphertext pairs (a known plaintext framework), then XORing such a pair gives the r-bit part of the state (bitrate part). Thus, we can take two consecutive plaintext-ciphertext pairs and try to determine the unknown part of the states (capacity part). Clearly, such a state-recovery attack is captured by a notion of the CICO problem. For some algorithms, such as Ketje, the bitrate is very small (e.g. one byte) and to uniquely recover the actual state we need more plaintexts-cipher blocks. This scenario is captured by the multi-block CICO problem.

A natural way to solve the CICO problem is to express it as a set of algebraic equations in a set of unknowns and apply algebraic techniques for solving these equations. One of such generic techniques (or tools) is a SAT solver.

2.3 SAT-based Cryptanalysis

SAT is the first known NP-complete problem, proven by Stephen Cook in 1971 [Coo71]. A SAT solver is an algorithm that decides whether a given propositional (boolean) formula has a satisfying evaluation. Finding a satisfying evaluation is infeasible in general, but many SAT instances can be solved surprisingly efficiently. There are many competing algorithms for solving SAT instances and many implementations, most of them have been

developed over the last two decades as highly optimized versions of the Davis, Putnam, Logemann and Loveland (DPLL) procedure [DLL62, DP60]. Modern SAT solvers use finely tuned algorithms and data structures to find a solution for a given instance coded in a conjunctive normal form (CNF) form. To solve an instance of a problem, (1) the instance is first translated into a corresponding SAT problem (in such a way that a satisfying evaluation represents a solution to the instance) and (2) a SAT solver is run to find one or more satisfying evaluations. The first connection between the SAT problem and cryptography dates back to [CM97], where a suggestion appeared to use crypto formulae as hard benchmarks for propositional satisfiability checkers. Courtois and Pieprzyk [CP02] translated cryptographic structures into large systems of low degree equations showing a potential of algebraic approach in analysis of block ciphers. Interestingly, SAT solvers were successfully applied against KeeLoq — the cipher deployed in car industry [CBW08]. Recently, a SAT solver approach was used to prove a resistance of the Salsa20 cipher against differential cryptanalysis [MP13].

(1) CNF generation.

A formula given to a SAT solver has to be expressed in a Conjunctive Normal Form (CNF). Therefore, for each scheme we need to convert the CICO (or multi block CICO) problem into a CNF (solution of which corresponds to the unknown bits of the input and the output in the CICO problem). Most operations in the analysed ciphers can be easily described by short bitwise equations, with the AND and XOR operators. For such operations a translation is straightforward. For instance, for 1-bit XOR equation $z = x \oplus y$, the corresponding CNF has the form

$$(\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

For S-boxes, which are typically specified as a truth table, we try two approaches. First approach is to use the bitwise specification for a given S-box and convert simple bitwise equations to the CNF. A drawback of this simple approach is that we need to add extra temporary variables. The other approach is to give the truth table of an S-box to the Espresso minimisation tool [Rud86] and obtain the CNF. In some cases Espresso produces much smaller CNFs (fewer number of clauses). We confirm the validity of the generated CNF by checking if a solution (output bits) found by a SAT-solver corresponds to the actual solution from test vectors for a given authenticated cipher.

(2) SAT solvers.

We have considered several SAT solvers as potential candidates, including `CryptoMiniSat` [Soo16], `plingeling` [Bie16], `treengeling` [Bie16], `glucose` [AS12]. Most of them performed very well on the latest SAT competitions (particularly in the parallel track category). After certain number of tests, we have decided to use the SAT solver `plingeling` as in our experiments it outperforms the other solvers.

3 Analysis of Sponge-Based Schemes

In this section we present our state recovery attacks on the six AE schemes from the CAESAR competition. We divide the schemes into two groups: sponge-based (given in Sect. 3) and other (in Sect. 4). For each algorithm we give a brief description followed by details of the attack.

3.1 Ketje

Ketje is an authenticated encryption algorithm which targets memory-constrained devices [BDP+]. The scheme supports two families of ciphers, namely Ketje Sr and Ketje Jr. Ketje Sr uses 400-bit states and is based on the round-reduced Keccak-f[400] permutation, whereas Ketje Jr operates on 200-bit states with the Keccak-f[200] as an underlying permutation. In our analysis, we focus on Ketje Jr.

The permutation of Ketje Jr has the bitrate $r = 16$, capacity $c = 184$ and supports keys of lengths up to 182 bits. The claimed security level is $(c - 1 - log_2 M)$ bits, where M is a number of blocks observed online (data complexity). For example, if data complexity is limited to 2^{23} blocks, then the security level is $184 - 1 - 23 = 160$ bits, provided that a key is at least 160-bit long. The Ketje Jr family submitted to the CAESAR competition claims the 96-bit security level with the data complexity limit of 2^{87} block observations. In our SAT analysis, we aim at verifying the claimed security level.

A small bitrate (16 bits) of Ketje Jr suggests that we should focus on the multi-block CICO problem. We investigate the following scenarios:

- **Basic.** In this scenario, we take the outputs of 13 consecutive rounds (as the ratio of state size to the bitrate is 12.5) and try to recover the state from these outputs. The available information from collected data is translated into a corresponding SAT instance with 6440 variables and 44944 clauses. The SAT solver was unable to find a solution within a reasonable time (most of our solvers were running on 8 cores for a few days).

- **Guess in the middle.** In the second scenario, we guess t bits of the state in the middle round (i.e. the round $n_r/2$). This is to say that apart from the 2 output bytes of each of the n_r rounds, we know t bits in the middle. Consequently, the complexity of the state recovery has increased by factor of 2^t, i.e., the complexity is a product of 2^t and time taken by a SAT solver working on the instance. After guessing 17 bytes in the middle ($t = 8 \cdot 17 = 136$ bits) and observing outputs of 4 rounds, we know $17 + 4 \cdot 2 = 25$ bytes. This is enough to determine the entire 25-byte state uniquely[3]. The resulting SAT instance is defined by 1640 variables and 10520 clauses. We were able to solve it instantaneously. Increasing the output rounds to 5 (and reducing the guessed bytes in the middle to 15) did not lead to better results.

- **Guess in every round.** In the third scenario, we guess additional b bits of the state for each of n_r rounds. Therefore, the complexity of the state recovery has increased by a factor of $2^{n_r \cdot b}$. To find a unique solution, the parameters should satisfy $2^{n_r(16+b)} \geq 2^{200}$. When $n_r = 3$, we guess 6 bytes per round and an additional byte in middle round, hence we guess $3 \cdot (6 + 2) + 1 = 25$ bytes in total. This is translated into an appropriate SAT instance with 1160 variables and 7080 clauses, which has been solved instantaneously. We also tried a different strategy by guessing $3 \cdot 6 + 1 = 19$ bytes. It turns out that we can still recover the state. However, increasing a number of rounds does not lead to better results. We have tried $n_r = 4$ and 4-byte guesses per round plus an additional guess in the middle. Unfortunately, the corresponding SAT instances with 1640 variables and 10520 clauses cannot be solved quickly. In fact, even if we know 25 output bytes but $n_r = 4$, we still cannot find a solution.

We conclude that guessing in the middle turns out to be the best strategy: it needs 136-bit guesses in the middle round to recover the state. Corresponding SAT instances are

[3]There is a high probability that fixing 25 bytes of outputs from different rounds correspond to a unique 25-byte state at certain round.

solved in ≈ 2.7 seconds on the 8-core PC (an average value out of 100 trials). If we assume that a time taken by a solver is roughly equivalent to 2^{29} Ketje Jr calls[4], then complexity of the state recovery is $2^{136+29} = 2^{165}$. Therefore, instances of the Ketje Jr family, which offer security levels higher than 165 bits, are susceptible to SAT-based analysis and they will not meet the claimed security levels.

3.2 ICEPOLE

ICEPOLE is a family of hardware-oriented authenticated ciphers [MGH+14]. There are two main variants of the algorithm, one with 128-bit key and 1024-bit block size (rate), the other one with 256-bit key and 960-bit rate. The cipher was designed for high-throughput network nodes, hence the big 1280-bit state. The underlying permutation is an iterative transformation with a linear MDS matrix followed by a layer of 5-bit S-boxes. The 12-round permutation is used in both the initialization and finalization stages, while the 6-round permutation is applied for the processing stage. ICEPOLE has an additional parameter called the secret message number, which has a similar effect as the extra key addition in ASCON.

In 2014, Huang et al. presented differential-linear cryptanalysis of ICEPOLE, where both a nonce and secret message are reused [HTW15]. It helped to refine the nonce-misuse resistance in the algorithm. In [DEM15a], the forgery attack was performed on the round-reduced ICEPOLE-128 (up to 4 rounds). Dobraunig et al. also found linear characteristics, up to 6 rounds, using an automated tool they created [DEM15b]. More references to third-party cryptanalysis can be found on the ICEPOLE official webpage [MGH+14].

In our analysis we focus on the ICEPOLE variants without a secret message number, specifically ICEPOLE-128a. For this variant, once we know the state, we can invert the permutation and get the secret key. ICEPOLE-128a works with the 128-bit key and the 1024-bit bitrate. So, we are dealing with a relevant CICO problem that is defined for 1024 known input and output bits (the bitrate part), while the remaining 256 bits of the capacity part are unknown. Since the 1024-bit bitrate part is much bigger than the 256-bit capacity part, we need only two consecutive plaintext-ciphertext pairs to uniquely determine the state.

A single round of the ICEPOLE permutation is described by a CNF with 12416 variables and 39424 clauses. Note that the two steps of the ICEPOLE algorithm, namely ρ (rotation along lanes) and π (permutation between lanes) do not increase complexity of the CNF relations as they permute the state bits only.

We are able to recover the state for the 4-round permutation (the original ICEPOLE uses 6 rounds). To make our attack successful, we need to guess 64 bits in the middle (e.g., after 2 rounds). So, the time complexity of our attack is $2^{64} \cdot t$, where t is a time taken by a solver to find a solution. The attack has been implemented on a desktop PC 3.7 GHz with 8 threads and t is around 13 hours. This is roughly equivalent to 2^{44} ICEPOLE encryptions on a given PC. Thus, time complexity of the 4-round key-recovery attack is about $2^{64+44} = 2^{108}$.

3.3 ASCON

ASCON is a family of authenticated encryption algorithms [DEMS]. The designers specify two variants whose bitrates are 64 and 128. The size of the key, nonce and tag are 128 each. The state consists of five 64-bit words so it is 320-bit long. The algorithm uses

[4] Output bits taken from 13 rounds ($13 \times 16 = 208$ bits) should correspond to a unique 200-bit state. In 1 second, on the 8-core 2GHz PC, we evaluate around 2^{28} 13-round Ketje Jr calls. We stress that it is very difficult to precisely convert a time taken by a solver to Ketje Jr evaluations. For example, operations on cores may not scale as one would expect, different platforms and implementations might affect time as well.

two permutations p_a and p_b that are constructed from the same elementary permutation round. The round is built from three basic operations: constant addition, an nonlinear S-box layer and a linear transformation. It is worth noting that the sponge-based mode in ASCON is made stronger by extra key addition in initialization and finalization. This is to prevent a straightforward key-recovery once the whole state is known.

The designers analysed their algorithm through a number of techniques and their findings were published at CT-RSA'15 [DEMS15]. Their best result was a key-recovery attack on 6 rounds (out of 12). In 2016, Tezcan presented truncated, impossible and improbable differential attacks, achieving 5 rounds of the permutation. More references can be found on the ASCON official website [DEMS].

We analyse ASCON-128a, where both the bitrate and key have 128 bits. First, we try the state recovery that follows the one described for ICEPOLE. For the CICO problem corresponding to the attack, 128 bits of both input and output are known and we hope that our SAT solver finds the (unknown) capacity part of the state. We have derived CNF relations for a single round with 2304 variables and 7936 clauses. Although the CNF is smaller than the corresponding CNF for ICEPOLE, solving it turns to be more difficult and we can reach two rounds only. This counter-intuitive phenomenon has been also observed in the related work [HMRS12]. Interestingly, what makes a CNF instance difficult is not the 'width' (or a total number of variables and clauses) but the 'depth' of the formula (number of literals in clauses). The ASCON Sbox is more complex than the ICEPOLE Sbox. This seems to be the main reason for increased complexity of solving ASCON CNF instances, despite the fact that ASCON has a state that is four times smaller than the one in ICEPOLE.

For ASCON, the key-recovery attack is not straightforward due to an extra key addition in the initialization and the finalization phases. We try to recover the key assuming that the state is already recovered and known to the attacker. This time the CICO problem models the finalization phase, where an unknown input is the secret key and a known output is a 128-bit authenticated tag. Basically, the problem in hand looks very similar to the state-recovery considered earlier. As expected, our SAT solver is able to find the secret key for 2 rounds only. A time taken by a solver is roughly equivalent to 2^{32} ASCON encryptions.

We have considered 3- and 4-round variants with some bits guessed in the middle (e.g. 64 bits after 1.5 rounds) but these instances turn to be too difficult for the solver.

3.4 NORX

NORX is a family of authenticated ciphers with two main variants based on 32 or 64-bit words. It was optimized to be efficient in both software and hardware with a SIMD-friendly core. The underlying permutation F is inspired by ChaCha stream cipher [Ber]. However, the integer addition is replaced by a simple bitwise operation. Thus NORX relies only on the bitwise XOR, AND, rotations, which leads to better hardware efficiency and somewhat simplifies the cryptanalysis. The round function operates on a state composed of 16 words in a similar fashion to the hash function BLAKE [AHMP]. The recommended variant of NORX works with 256-bit key, 256-bit tag and 128-bit nonce, whereas a number of rounds is 4.

In [AJN14a], the designers of NORX have investigated differential and rotational properties of the algorithm. More recently, state and key recovery on 2-round variants have been presented [BHJ+16]. These results were obtained with the 'guess and determine' technique and the internal differential attack. Interestingly, using higher-order differential analysis, a 4-round distinguisher was shown [DMM15]. However, the distinguisher cannot be used to attack the whole scheme, but only to show a weakness of the underlying permutation in NORX.

Our analysis is focused on the version with 32-bit words called NORX32. This version has a 512-bit state. During the encryption phase, NORX32 uses 4 BLAKE-like rounds to process a plaintext block of 12 words (i.e. the rate of the sponge is 384 bits). Note that the authors claim 128-bit security level for this variant. The state recovery of NORX32 can be looked as an instance of the CICO problem and be converted into a relevant SAT instance. We start from the state recovery for encryption reduced to a single round. We assume that we are given two consecutive pairs of known plaintext-ciphertext and try to recover the state. The resulting SAT instance is defined with 2560 variables and 16896 clauses. It is solved in a matter of seconds, so the state recovery of NORX32 reduced to one round is feasible.

We try to extend our analysis to 1.5 rounds. For this scenario, the SAT instance has 3584 variables and 24960 clauses. However, it is beyond the reach of the SAT solver. To progress, we guess extra bits of the state. More precisely, we guess bits from the capacity part of the output state and also from the state of the middle round. To introduce additional flexibility, we employ two guessing strategies: consecutive guessing of bits of state words (i.e. if we guess 80 bits of the state, then it means we know 2.5 words) and partial bit guessing, where we fix the values of a few bits of each (unknown) word of the state. It turns out that regardless of the guessing strategy, we still need to guess around 64 bits of the state to make SAT instances feasible for the solver. For instance, if we guess one additional word in both the input and output, then the resulting SAT instance (3584 variables, 25024 clauses) can be solved in a few seconds. A time taken by a solver is roughly equivalent to 2^{32} encryptions, then the total time complexity of the attack is $2^{64+32} = 2^{96}$. If we guess one word at the end, and one in the middle, we end up with a similar complexity. However, only the first type of guessing (the consecutive bits) leads to such results. The partial guessing does not lead to a feasible SAT instance when the number of guesses is around 64.

We try to recover the state for the variant of NORX32 reduced to 2 rounds, but the total complexity (bit guesses and workload of the SAT solver) is over 2^{128}. The most efficient recovery that we have found requires 56 bit guesses in the middle state and 56 bit guesses of state at the output (i.e. 2^{112} factor only for bit guesses). The SAT solver still needs around 10 seconds to produce a solution, thus the total complexity exceeds a simple state recovery based on an exhaustive key search.

To summarize, given two consecutive pairs of plaintext-ciphertext, it is possible to recover the state of the 1-round NORX32 in a matter of seconds. If NORX32 is reduced to 1.5 rounds, the recovery is still possible, although its time complexity of 2^{96} makes the attack only theoretical.

4 Analysis of Other Schemes

4.1 MORUS

MORUS is a family of authenticated encryption algorithms, suitable both for hardware and software efficient implementations. It has three variants: MORUS-640-128, MORUS-1280-128 and MORUS-1280-256, where the first number denotes a state size, and the second number is a key size, both given in bits. The state consists of 5 words (either 128-bit or 256-bit, depending on the variant). The state update function has 5 very similar rounds. There are four operations, namely XOR, AND, rotation in words (denoted by $\lll$) and rotation in subwords (denoted by *Rotl*). Every round modifies only two registers, one is modified by $\lll$, while the other one by a set of ANDs, XORs, and *Rotl*. Security of the algorithm heavily depends on a strong initialization phase, where the state update function is called 16 times. MORUS does not rely on any specific mode, particularly it is not a sponge construction. Its encryption process can be seen as a stream cipher with

large state which is updated continuously. For a full description of the algorithm, we refer the reader to [WH].

In [MDV15], Mileva et al. presented MORUS analysis, including the forgery attack. However, all the attacks reuse nonce, which violates MORUS security requirements. Some concerns about MORUS security is highlighted in [Saa15], where the author claims that MORUS represents significantly elevated adaptive-chosen-plaintext attack risk.

We investigate the SAT-based state recovery of MORUS-640. At message processing phase, a single call of state update functions takes one 128-bit message word and outputs one 128-bit ciphertext word. With 128-bit key, the claimed security level is 128 bits. As the ratio of the state size to the ciphertext block size is 5, we need 5 consecutive pairs of plaintext-ciphertext to recover the state (almost) uniquely.

To keep our analysis simple, we assume that 5 plaintext blocks are all-zero blocks, that is we are operating in the chosen-plaintext framework[5]. We proceed according to the following scenarios:

- **Basic.** We reduce the state recovery of MORUS-640 to a SAT instance, which encodes 5 consecutive calls to the state update function. As a result, we get an instance with 3200 variables and 34560 clauses. Our SAT solver has failed to find a solution.

- **Simplified output function.** For MORUS-640, a ciphertext block C is calculated from a simple formula $C = P \oplus S_0 \oplus (S_1 \lll 96) \oplus (S_2 \& S_3)$, where P is a plaintext block and $S_0 \dots S_3$ are 4 state words obtained from the state update function. Here we simplify the formula by taking only the first word $C = P \oplus S_0$. As we work in a chosen plaintext scenario and assume that plaintext blocks are all-zero blocks, then $C = S_0$. For this simplified version of MORUS-640, a corresponding SAT instance (that aims to recover a state) has 3200 variables and 31360 clauses. Even though now we have a smaller number of clauses, the instance is still infeasible for our SAT solver — a few days of processing has failed to find a solution.

- **Toy variant with a reduced word size.** We would like to find out whether a significant reduction of a word size makes the problem easier for a solver. We try to estimate the complexity of the SAT-based state recovery by solving a reduced version of MORUS-640 with 60 instead of 640 state bits. We need to modify the algorithm for the reduced state size. XOR and AND operations are applied in a straightforward way, while rotations are adjusted so they reflect the original design as closely as possible. Consequently, for the word-reduced version of MORUS, the corresponding SAT instance consists of 300 variables and 3240 clauses. The SAT solver is able to find a solution in a few days of computation. However, the generic key-recovery requires 2^{12} guesses only (as an appropriate key size of the variant is 12 bits only). We conclude that the SAT-based approach does not break the algorithm as it would take much more time then the exhaustive search of the key. Even though this is not a formal proof that MORUS-640 is immune to the SAT-based state recovery, it is a good indication that the algorithm stands strong against this type of analysis.

The above findings suggest that MORUS-640 does not show any weakness against analysis with SAT solvers. Let us examine an upper bound on the security level of this scheme.

Here, as in previous analysis, we mount the state recovery attack to get the secret key. Let us recall that a ratio of the state size to the ciphertext block size is 5, thus we need to

[5] We can switch to the known-plaintext attack at the expense of increasing a number of clauses in SAT instances.

encode 5 consecutive calls to the state update function (getting 5 ciphertext blocks) to determine the state uniquely. We start our experiment using a big number of guessed state bits, making the instance easy to solve. Then, we gradually reduce the number of guessed bits and stop when the state recovery starts taking more than a few seconds. This allows us to quantify the time complexity of appropriate SAT instances. We have many choices about which state variables should be guessed. We experiment with guessing consecutive bits of the state and also with guessed bits which are spread equally over 5 words of the state. It turns out that we get the best results, when we guess consecutive bits in the 'middle', that is after the second call to the state update function. On average, we need 340 guessed bits to have a formula solved. If we approximate the time taken by a solver by a factor of 2^{30}, we can conclude that MORUS-640 cannot offer a security level higher than around $340 + 30 = 370$ bits. Please note that it does not affect MORUS security claims as it is specified only to support either 128- or 256-bit keys. However, MORUS works on a big internal state and one can be tempted to adjust the algorithm for bigger keys and corresponding security levels. Then our analysis shows there is a limit.

4.2 ACORN

ACORN is basically a stream cipher based on linear feedback shift registers (LFSR). It operates on the 293-bit state, which is split into a short 'buffer' segment followed by 6 LFSRs. This design allows for 32 bits to be processed in parallel, leading to efficient hardware and software implementations. Interestingly, plaintext bits are not only XORed with keystream (to produce ciphertext), but also XORed into the main feedback function. Having the state directly affected by plaintext helps realize the authenticity property. The main feedback function is the only non-linear part of the state update. For a full description of the algorithm, see [Wu].

There are two ways we could mount the state recovery attack for ACORN. First, one can encode the whole initialization phase into a SAT formula with key bits as unknowns and some keystream bits as an input. However, this approach is infeasible as there are 1792 steps in the initialization phase, so the formula is too big and complicated for a solver. The other approach is to take an arbitrary state in the encryption phase and generate keystream equations in the state variables. Again, for 293 unknowns in the state, the problem turns out to be too hard for a solver. To make the problem easier, we fix some state bits and with 170 known bits we are able to recover the state in about 8 hours. However, guessing that many bits makes our analysis much above an exhaustive search for a 128-bit key.

We conclude that ACORN is strong against SAT-based cryptanalysis. Particularly, we notice that state bits involved in an equation for one keystream bit are not present in the following 40 keystream equations. Consequently, it is hard to build a set of equations, which have common variables and at the same time are not too complex. We also notice that the state update function creates long XOR equations, which are known to be difficult for SAT solvers. We believe these two factors contribute the most to the resistance against our SAT-based attack. Because ACORN has no rounds or blocks that could be reduced, there is no simple and natural way of creating a reduced or toy version that we would be able to break.

5 Conclusion

Our findings have filled an important and common gap in the cryptanalysis of CAESAR proposals — the resistance of the schemes against SAT-based state recovery attacks. We have analysed six authenticated encryption algorithms from the CAESAR competition and our extended analysis has shown that none of these schemes, as submitted to CAESAR,

allows even a "theoretical" break using SAT-based state recovery attacks. On the other hand, some round-reduced versions or variants with artificially increased security levels are susceptible to state and key recovery attacks. This helps to quantify the security margin and we conclude the analysed ciphers have a sufficient security margin against SAT-based cryptanalysis.

Acknowledgement

This project was financed by National Science Centre, Poland, project registration number DEC-2014/15/B/ST6/05130. The 2nd author was partly supported by the grant SVV-2016-260336. Ivica Nikolić is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[AHMP] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. SHA-3 proposal BLAKE. http://www.131002.net/blake/.

[AJN14a] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. Analysis of NORX: Investigating Differential and Rotational Properties. In *Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers*, pages 306–324, 2014.

[AJN14b] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. Norx: parallel and scalable aead. In *European Symposium on Research in Computer Security*, pages 19–36. Springer, 2014.

[AS12] Gilles Audemard and Laurent Simon. GLUCOSE 2.1 in the SAT Challenge 2012 . In *Proceedings of SAT Competition 2012: Solver and Benchmark Descriptions - SAT Competition 2012*, page 21, 2012.

[BDP+] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer. CAESAR submission: KETJE v2 . http://ketje.noekeon.org.

[BDPV] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Cryptographic Sponges. http://sponge.noekeon.org/CSF-0.1.pdf.

[BDPV11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 320–337, 2011.

[Ber] Daniel J. Bernstein. ChaCha, a variant of Salsa20. https://cr.yp.to/chacha/chacha-20080120.pdf.

[BHJ+16] Nasour Bagheri, Tao Huang, Keting Jia, Florian Mendel, and Yu Sasaki. Cryptanalysis of Reduced NORX. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 554–574, 2016.

[Bie16] Armin Biere. Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016. In *Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions - SAT Competition 2016*, pages 44–45, 2016.

[CAE] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html.

[CBW08] Nicolas Courtois, Gregory Bard, and David Wagner. Algebraic and Slide Attacks on KeeLoq. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *LNCS*, pages 97–115. Springer Berlin / Heidelberg, 2008.

[CM97] Stephen A. Cook and David G. Mitchell. Finding Hard Instances of the Satisfiability Problem: A Survey. pages 1–17. American Mathematical Society, 1997.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[CP02] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer Berlin / Heidelberg, 2002.

[DEM15a] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Forgery Attacks on Round-Reduced ICEPOLE-128. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 479–492, 2015.

[DEM15b] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 490–509. Springer, Heidelberg, November / December 2015.

[DEMS] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon A Family of Authenticated Encryption Algorithms. http://ascon.iaik.tugraz.at.

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 371–387. Springer, Heidelberg, April 2015.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 7(5):394–397, 1962.

[DMM15] Sourav Das, Subhamoy Maitra, and Willi Meier. Higher Order Differential Analysis of NORX. *IACR Cryptology ePrint Archive*, 2015:186, 2015.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[HMRS12] Ekawat Homsirikamol, Pawel Morawiecki, Marcin Rogawski, and Marian Srebrny. Security Margin Evaluation of SHA-3 Contest Finalists through SAT-based Attacks. In *11th Int. Conf. on Information Systems and Industrial Management*, volume 7564 of *LNCS*. Springer Berlin Heidelberg, 2012.

[HTW15] Tao Huang, Ivan Tjuawinata, and Hongjun Wu. Differential-linear cryptanalysis of ICEPOLE. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 243–263. Springer, Heidelberg, March 2015.

[LLMH16] Frédéric Lafitte, Liran Lerman, Olivier Markowitch, and Dirk Van Heule. SAT-based cryptanalysis of ACORN. *IACR Cryptology ePrint Archive*, 2016:521, 2016.

[MDV15] Aleksandra Mileva, Vesna Dimitrova, and Vesselin Velichkov. Analysis of the Authenticated Cipher MORUS (v1). In *Cryptography and Information Security in the Balkans - Second International Conference, BalkanCryptSec 2015, Koper, Slovenia, September 3-4, 2015, Revised Selected Papers*, pages 45–59, 2015.

[MGH+14] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICEPOLE: High-Speed, Hardware-Oriented Authenticated Encryption. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 392–413, 2014. http://www.icepole.org.

[MP13] Nicky Mouha and Bart Preneel. A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis. Cryptology ePrint Archive, Report 2013/328, 2013. http://eprint.iacr.org/.

[Nat07] National Institute of Standards and Technology. Recommendations for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST special publication 800-38D, November 2007.

[Rud86] Richard L. Rudell. Multiple-Valued Logic Minimization for PLA Synthesis. Technical Report UCB/ERL M86/65, EECS Department, University of California, Berkeley, 1986.

[Saa15] Markku-Juhani O. Saarinen. The BRUTUS automatic cryptanalytic framework - Testing CAESAR authenticated encryption candidates for weaknesses. *Journal of Cryptographic Engineering*, 6(1):75–82, 2015.

[Soo16] Mate Soos. The CryptoMiniSat 5 set of solvers at SAT Competition 2016 . In *Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions - SAT Competition 2016*, page 28, 2016.

[Sto16] Ko Stoffelen. Optimizing S-Box Implementations for Several Criteria Using SAT Solvers. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 140–160, 2016.

[WH] Hongjun Wu and Tao Huang. The Authenticated Cipher MORUS. https://competitions.cr.yp.to/caesar-submissions.html.

[Wu] Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v3). https://competitions.cr.yp.to/round3/acornv3.pdf.