

A Hybrid BP-SAT Attack on MILENAGE

Combining the Complementary Strengths of Algebraic and Soft-Analytical Side-Channel Attacks

Elena Dubrova, Sönke Jendral, Linus Backlund and Mats Näslund

KTH Royal Institute of Technology, Stockholm, Sweden,
{jendral,dubrova,lbackl,matsna}@kth.se

Abstract. Profiled side-channel attacks combine noisy leakage predictions with the mathematical relations imposed by the target algorithm to recover secrets from a small number of observations. Two established approaches are algebraic side-channel analysis (ASCA), which encodes leakage predictions and algorithmic relations as a Boolean satisfiability (SAT) problem, and soft-analytical side-channel analysis (SASCA), which performs probabilistic inference over a factor graph using belief propagation (BP). We evaluate both on the AES-based MILENAGE authentication algorithm deployed on USIM cards, targeting the first AES-128 invocation on $RAND \oplus OPc$ under the subscriber key K , where OPc is an operator-specific secret unknown to the attacker. Neural networks predict posterior probability distributions over the AES S-box output byte values across multiple rounds and these predictions are supplied to ASCA and SASCA to recover K and OPc from a single trace. SASCA recovers both secrets in 87.7% of attacks, but converges to an incorrect OPc in 6.1% of cases despite recovering K correctly. ASCA enforces global consistency and resolves this ambiguity, but it succeeds in only 34% of attacks under a 1-hour timeout and our constraint-selection strategy. Motivated by these complementary failure modes, we combine the two into a hybrid attack in which BP refines the neural network predictions before they are supplied to the SAT solver. The hybrid approach recovers both secrets in 93.8% of single-trace attacks, resolving all cases where SASCA recovers K correctly but OPc incorrectly. It has a higher success rate and a lower average runtime than either method alone, despite performing more work per attack. Finally, we suggest a countermeasure, based on masking each AES round key with an independent, per-device secret, that reduces an attacker’s advantage over guessing K and OPc to zero bits, and confirm empirically that it defeats the hybrid attack in all evaluated cases.

Keywords: Side-channel analysis · SASCA · ASCA · Belief propagation · SAT · MILENAGE · AES

1 Introduction

Single-trace side-channel attacks pose a serious threat to cryptographic implementations. Countermeasures and defenses that rely on limiting an attacker’s number of observations, such as frequent key refresh or rate-limited authentication, provide no protection once a secret can be recovered from a single execution: the attacker only needs to compromise the one execution they observe. Evaluating the limits of single-trace attacks is therefore central to assessing the side-channel resistance of a cryptographic implementation.

Profiled attacks have substantially improved single-trace key recovery, particularly when combined with *algebraic side-channel analysis* (ASCA) [RS10] or *soft-analytical side-channel analysis* (SASCA) [VCGS14]. Profiled attacks build a leakage model using

templates [CRR02] or neural networks [MPP16], and use it to derive a posterior probability distribution, conditioned on the observed leakage, over all possible values of each targeted intermediate variable. These distributions are then combined with the algorithm’s structure into a single hypothesis for the secret. ASCA encodes the algorithm together with a hard-decision subset of the highest-confidence predictions as a Boolean satisfiability (SAT) instance; SASCA encodes the algorithm as a factor graph and retains the full probability distributions as soft evidence for belief propagation (BP).

We implement and evaluate both approaches on the AES-based MILENAGE authentication algorithm [3GP99], deployed on Universal Subscriber Identity Module (USIM) cards and used to derive authentication tokens and session keys in 3G/4G/5G networks. Unlike a conventional known-plaintext AES attack, MILENAGE’s first AES-128 invocation encrypts $RAND \oplus OPc$ under K . Hence, the challenge $RAND$ is known to the attacker, but the 128-bit operator secret OPc is not.

K and OPc are not constrained symmetrically: the initial round ties $RAND \oplus OPc$ and K to the intermediate values of the first round, but only K benefits from the redundancy provided by the AES key schedule. This asymmetry makes MILENAGE a useful testbed for comparing ASCA and SASCA, which make opposite choices in how to handle uncertain predictions: SASCA retains every prediction, including low-confidence ones, as soft evidence; ASCA commits to a hard-decision subset, trading a smaller search space for the risk of asserting an incorrect constraint. MILENAGE’s K/OPc asymmetry lets us observe where each choice wins, motivates a hybrid attack that combines both, and points to a countermeasure that removes the asymmetry directly.

Our contributions:

- **Single-trace ASCA and SASCA attacks on MILENAGE.** We implement and evaluate both attacks on the AES-based MILENAGE algorithm, in a cross-device setting where profiling and attack traces come from different physical USIM cards. Using the same attack traces and the same neural network models to derive probability distributions on the targeted intermediate variables for both attacks, we isolate the effect of the inference mechanism from the quality of the underlying leakage model. This controlled comparison exposes a difference between failure modes of the two methods: SASCA’s soft BP can converge to a correct K but an incorrect OPc , while ASCA’s exact SAT only terminates within the time budget when sufficiently many correct constraints are provided.
- **Hybrid BP-SAT attack.** Motivated by these results, we propose a hybrid attack combining BP-based refinement with SAT-based exact consistency checking. The hybrid approach recovers both K and OPc from a single trace in 93.8% of 1,000 evaluated attacks, improving on standalone ASCA and SASCA in success rate and, despite performing more work per attack, in average runtime.
- **Countermeasure.** We propose independent per-round-key masking as a countermeasure against the attacks on MILENAGE and confirm empirically that it defeats the hybrid attack in all evaluated cases.

The remainder of the paper is organized as follows. Section 2 reviews the MILENAGE algorithm and the ASCA and SASCA methods. Section 3 presents the threat model. Section 4 details the CNF encoding used by ASCA. Section 5 describes our SASCA implementation: the AES/MILENAGE factor graph, the BP message-passing schedule, and its validation against reference test vectors. Section 6 presents the hybrid attack method. Section 7 describes our equipment, trace acquisition and synchronization, as well as neural network training and testing. Section 8 reports the success rate and runtime of ASCA, SASCA, and the hybrid single-trace USIM attacks. Section 9 proposes and

analyzes a countermeasure against all three attacks. Section 10 discusses related work. Section 11 concludes the paper.

2 Background

This section gives background information on the MILENAGE algorithm and the ASCA and SASCA attacks.

2.1 MILENAGE Authentication Algorithm

MILENAGE is the authentication algorithm specified by the 3rd Generation Partnership Project (3GPP) for the Authentication and Key Agreement (AKA) protocol used in 3G, 4G, and 5G mobile networks (3GPP TS 35.206 [3GP99]). The algorithm is executed inside the USIM to implement the authentication functions $f1$ – $f5$, $f1^*$, and $f5^*$. These functions provide mutual authentication between the subscriber and the network and derive authentication and session parameters, including the authentication response RES , cipher key CK , integrity key IK , authentication codes $MAC-A$ and $MAC-S$, and anonymity keys AK and AK^* .

The security of MILENAGE relies on two 128-bit secret parameters associated with the USIM. The first is the subscriber authentication key K , unique to each subscriber and shared with the subscriber’s home network. The second is OPc , derived from the operator-specific constant OP and K . Depending on the implementation, OPc may be stored in the USIM during personalization or computed internally for each execution by encrypting OP under K ; the sysmoUSIM-SJS1 cards used in our experiments belong to the latter case. In either case, OPc is not available to an attacker observing an authentication exchange.

A MILENAGE execution takes as input a 128-bit random challenge $RAND$ together with several public parameters, including the sequence number SQN , authentication management field AMF , rotation constants $r_1, \dots, r_5$, and constants $c_1, \dots, c_5$. The first internal AES-128 invocation computes

$$TEMP = AES_K(RAND \oplus OPc),$$

and AES-128 is subsequently invoked five more times to process $TEMP$ and derive the authentication and key-derivation outputs.

2.2 Algebraic side-channel analysis

ASCA [RS10] represents the target algorithm as a Boolean circuit, converts it to Conjunctive Normal Form (CNF) (typically via the Tseitin transformation), and introduces known values and selected leakage observations as unit clauses by fixing the corresponding variables to a given value.

A SAT solver then searches for a variable assignment consistent with both the algorithm’s computation and the asserted constraints. If sufficiently many correct constraints are provided, the resulting satisfying assignment is unique, and the secrets being recovered follow directly from the corresponding CNF variables. If no satisfying assignment exists, the solver reports the instance unsatisfiable (UNSAT).

This hard-decision nature of ASCA creates a trade-off in selecting which predictions to assert. Adding more predictions gives the solver more information and can reduce the search space, but even a single incorrect prediction can either render the instance UNSAT (if inconsistent with the remaining constraints) or permit a satisfying assignment that does not correspond to the true secret values (if the remaining constraints are insufficient to converge to a unique solution).

2.3 Soft-analytical side-channel analysis

SASCA [VCGS14] encodes the target algorithm as a factor graph over discrete random variables in which variable nodes represent intermediate values and factor nodes represent the relations between them. Leakage observations are incorporated as probabilistic evidence factors on the corresponding variables. Unlike ASCA, SASCA never collapses a prediction to a hard decision: even low-confidence predictions remain available as soft evidence throughout inference.

Inference is performed by loopy BP which passes messages between variable and factor nodes, typically using the sum-product algorithm. In the AES case, because the AES round structure and key schedule introduce cycles into the graph, BP computes approximate rather than exact posterior marginals. The resulting marginals provide posterior information about every intermediate value of the algorithm, including the secrets being recovered.

3 Threat Model

We consider an adversary who aims to recover the secrets K and OPc stored in a USIM card. The attacker has temporary physical access to the target USIM and measures its power consumption while the card executes MILENAGE, capturing traces during authentication requests whose challenge $RAND$ is known to the attacker. The attacker does not need to know the resulting $TEMP$ value produced by $AES_K(RAND \oplus OPc)$.

The attack follows the profiled side-channel scenario. During profiling, the attacker trains neural network models on power traces collected from USIM cards under their control to classify the selected intermediate variables, e.g. S-box output bytes in multiple rounds. During the attack phase, the attacker captures a single power trace from the target USIM and uses the trained neural networks to produce a posterior probability distribution over all possible values for each selected intermediate variable. These predictions are then incorporated into ASCA and SASCA to recover K and OPc .

Attack scope. Within this threat model, our attack point is the first AES-128 invocation within MILENAGE. All presented attacks exploit leakage from the outputs of the S-boxes in AES rounds one to nine, yielding $9 \times 16 = 144$ S-box output bytes in total.

The subsequent MILENAGE operations and AES invocations are not required by our attack and are not considered in this paper.

4 ASCA Implementation for MILENAGE

The ASCA implementation uses the same CNF encoding of the first AES-128 invocation within MILENAGE as in the attack presented in [DJB26]. The CNF consists of 9,408 clauses and 189,304 literals. It contains Boolean variables for every bit of K and OPc , all intermediate AES state bits, all AES round-key bits generated by the key schedule, and auxiliary variables introduced by the CNF conversion.

The linear AES transformations (AddRoundKey, ShiftRows, MixColumns) use standard CNF encodings [Gro20]. The nonlinear AES S-box operations are represented using precomputed Boolean constraints in the forward direction, from the 8 input bits to the 8 output bits. Consequently, fixing an S-box’s input bits allows unit propagation to immediately derive the corresponding output bits and continue propagating through the surrounding AddRoundKey, ShiftRows, and MixColumns constraints, with comparatively little branching required from the solver. Fixing the S-box’s output bits does not provide this benefit to the same degree, since propagation must then proceed backward through the same constraint. For this reason, each neural network-derived S-box *output* byte prediction

is converted through the inverse S-box mapping to the corresponding *input* byte prediction before being asserted as a unit clause.

5 SASCA Implementation for MILENAGE

This section describes our SASCA implementation for MILENAGE, including the factor graph, the BP message-passing schedule, and its validation against reference test vectors. This implementation is built and validated from scratch for this work, rather than using existing SASCA tools such as SCALib [CB23], to allow tight integration with the hybrid attack’s SAT stage.

5.1 Factor graph construction

Each AES-state byte, key-schedule intermediate byte, and each of the 32 bytes of K and OPc is represented by a length-256 probability vector. We use three types of factors:

Permutation factors represent the S-box, inverse S-box, and multiplication by a constant in $\text{GF}(2^8)$, as used in MixColumns. These factors perform exact, single-neighbor-conditioned reindexing and introduce no approximation.

XOR factors represent AddRoundKey, the XOR operations in the AES key schedule, and the first round relation between $RAND$, OPc , K , and S-box input variable $SboxIn_1$. The known $RAND$ byte is incorporated as a fixed constant. Messages through XOR factors are computed using the Walsh–Hadamard transform (WHT), which diagonalizes XOR convolution over $\text{GF}(2)^8$.

MixColumns factors represent the complete four-input, four-output MixColumns transformation for each AES state column. Each factor encodes the deterministic $\text{GF}(2^8)$ -linear relation between the four input bytes and four output bytes specified by the AES MixColumns matrix, modeling all eight bytes jointly rather than as four independent input-output equations. Messages are computed in both directions, so information from all four output bytes contributes to the message directed at each input byte, and vice versa. This avoids introducing short cycles within a MixColumns block, unlike a decomposition into four separate per-output-byte factors which produces a densely-cyclic sub-graph.

The factor graph models the initial round, the AES round function (SubBytes, ShiftRows, MixColumns, AddRoundKey) for rounds one to eight, plus the round-9 SubBytes permutation, matching the nine rounds of S-box output evidence available from the neural networks. Round-9’s ShiftRows, MixColumns, and AddRoundKey are not needed and therefore are not modeled.

The initial round relation is the main point at which the MILENAGE target differs from a conventional known-plaintext AES attack. Instead of a binary relation $SboxIn_1 = RAND \oplus K$, we use the quaternary relation

$$RAND \oplus OPc \oplus K \oplus SboxIn_1 = 0.$$

This relation is underdetermined when considered in isolation: it directly constrains only the XOR of OPc and K , rather than either secret individually. In particular, if one of the two secret variables has a uniform belief, the relation provides no information about the other secret in the absence of additional evidence.

The key K is, however, constrained independently through the AES key schedule which does not involve OPc . Once the belief on K becomes sufficiently concentrated, the initial round relation can in turn provide information about OPc . This asymmetric flow of

information is a plausible explanation for the recurring failure mode empirically observed in Section 8, in which BP recovers K correctly but converges to an incorrect OPc .

5.2 Message-passing schedule

We use a flooding schedule: at each iteration, every factor computes outgoing messages to all of its neighbors from the previous iteration’s *extrinsic* beliefs, obtained by dividing out the message previously received from that specific factor. All messages are then applied simultaneously at the end of the iteration. This corresponds to standard loopy BP.

It is worth mentioning a small numerical issue which arises when implementing $GF(2^8)$ XOR factors using the WHT. A straightforward approach computes each leave-one-out product as total/W_i , where total is the product of the WHT-transformed messages from all neighbors and W_i is the transform of the message from neighbor i . However, this approach is numerically unsafe when any neighbor has an exactly uniform belief, as is the case for every unknown variable at initialization. The WHT of a uniform belief is zero at all non-DC frequencies, causing total to vanish at these frequencies for every outgoing message, including those directed to non-uniform neighbors. Consequently, division by W_i cannot reliably recover the required leave-one-out product. To avoid this issue, we instead compute each leave-one-out product using division-free prefix/suffix products.

5.3 Validation

Before running on real profiled data, we validated each factor type against independent ground truth. The MixColumns factor was validated against a textbook AES MixColumns test vector, and the key-schedule factors were validated against the FIPS 197 Appendix A key-expansion example [Nat01]. We then validated the complete factor graph, with all AES rounds and both K and OPc treated as unknown, against a plain reference AES-128 implementation whose own round-by-round output was checked against a published FIPS 197 test vector.

Finally, we evaluated the complete graph on synthetic single-trace evidence constructed by perturbing distributions around the true S-box output values using Dirichlet noise. With 25 BP iterations, the graph recovered all 32 unknown bytes correctly across all tested random seeds and evidence confidence levels.

6 Hybrid Attack

In this section, we introduce a hybrid attack that combines ASCA and SASCA to leverage their complementary strengths. BP exploits the dependencies in the AES/MILENAGE factor graph to refine the noisy neural network S-box predictions, but it may struggle to recover underdetermined variables such as OPc . SAT complements BP by enforcing exact global consistency to find K and OPc subject to the refined S-box constraints.

6.1 BP-derived candidates

The hybrid attack runs BP on exactly the same AES/MILENAGE factor graph used for SASCA, with K and OPc both treated as free variables and no ground-truth or oracle information used to seed the graph. However, the hybrid’s BP stage uses only half of the iterations used by the BP stage of SASCA. Less-converged beliefs are sufficient because the subsequent SAT search enforces global consistency on its own.

For each round $r \in \{1, \dots, 9\}$ and each byte $b \in \{0, \dots, 15\}$, the value of the S-box input variable $SboxIn_{r,b}$ with the highest BP posterior is taken as the BP-derived candidate, with its posterior probability retained as confidence. Since these posterior marginals

incorporate information propagated through the full factor graph (the AES/MILENAGE round structure, MixColumns dependencies, and the key schedule) in addition to the local neural network evidence, a BP-derived candidate can correct erroneous local evidence.

6.2 Candidate merging

At each round-byte position, two candidates are available: the original neural network prediction with its softmax confidence, and the refined BP-derived prediction with its BP posterior confidence. The two candidate sets are merged by retaining the higher-confidence candidate at each round-byte position.

We also evaluated a method that always prefers the BP-derived candidate, irrespective of confidence; the confidence-based merge method performed at least as well and is therefore used for the results reported in Section 8.

6.3 SAT completion

The n highest-confidence candidates from the merged set are encoded as additional unit clauses and appended to the CNF representation of AES/MILENAGE, together with the attack-specific *RAND* constraint. The SAT solver then performs an exact search for an assignment to the remaining variables that is globally consistent with these constraints. Since neither K nor OPc is constrained by the BP posterior, an incorrect BP belief about K or OPc cannot itself make the SAT instance inconsistent; only an incorrect merged S-box candidate can potentially do so.

If the resulting SAT instance is UNSAT, n is reduced by a fixed step, and the SAT solver is retried with a smaller constraint set. This trades execution time against robustness: a larger n provides SAT with more information and reduces the search space, but also increases the risk of asserting an incorrect constraint, while a smaller n tolerates more prediction errors at the cost of a longer search.

A satisfying assignment yields the recovered K and OPc , which are compared against ground truth to determine attack success.

7 Experimental Setup

This section describes the measurement equipment, trace acquisition and synchronization, as well as neural network training and testing.

7.1 Equipment

The equipment consists of a ChipWhisperer-Pro, a LEIA-Solo smartcard reader, a PC, and three sysmoUSIM-SJS1 USIM cards, each programmed with a fixed key K and OPc .

The ChipWhisperer-Pro measures the total power consumption of the USIMs. The smartcard reader controls the clock and I/O lines of the USIM, manages communication between the USIM and the PC, and provides a trigger signal to the ChipWhisperer-Pro. The clock signal supplied to the USIM by the reader is also used as the ChipWhisperer-Pro's external clock, giving rough synchronization of the power measurements. Power consumption is sampled at 14 MHz, providing four samples per external clock signal cycle.

The PC controls the measurement setup and emulates the Mobility Management Entity (MME), communicating with the USIM via Application Protocol Data Unit (APDU) commands to initiate the MILENAGE authentication procedure.

The sysmoUSIM-SJS1 is an ETSI/3GPP- and GlobalPlatform-compliant Java Card USIM [Wel23] that implements the standard MILENAGE authentication algorithm specified in 3GPP TS 35.206. Unlike network operator-issued USIMs, it allows the user to

configure K and OP_c , enabling controlled evaluation of attacks. Note, however, that network operator-issued cards may incorporate vendor-specific side-channel countermeasures not present on this development card. Our experiments provide no evidence of masking or random-order shuffling in the AES implementation of the sysmoUSIM-SJS1 card. Consequently, the reported success rates reflect the security of an unprotected or lightly protected USIM and may not generalize to USIMs employing stronger side-channel countermeasures. We do not claim that our results are representative of any particular operator’s deployed USIM cards.

7.2 Trace Acquisition

For the profiling stage, a large set of power traces is collected from USIM cards executing MILENAGE with known fixed K and OP_c for randomly generated authentication challenges $RAND$. For every MILENAGE execution, the corresponding trace and $RAND$ are recorded. We collected 233,500 profiling traces from two USIMs for our experiments.

For the attack stage, a smaller set of power traces is collected from the target USIM. We collected 1,000 traces from the third USIM, different from the profiling USIMs.

7.3 Trace Synchronization

Even traces acquired from the same USIM show substantial timing variation across executions, whether due to inherent clock instability or intentional countermeasures; in either case, this variation is sufficient to misalign the underlying operations across traces.

Trace synchronization is performed in two stages. Given N full traces, first, the segments of traces corresponding to the initial AES-128 invocation within MILENAGE are located using cross-correlation with a template trace covering the ten rounds of AES-128. The N aligned segments are subsequently divided into nine single-round traces corresponding to the first nine AES rounds, resulting in $9N$ single-round traces in total. The tenth round is excluded because it does not contain the MixColumns operation and therefore exhibits a different leakage structure¹.

Fine-grained synchronization is then applied to the resulting set of single-round traces, T . For each byte $b \in \{0, \dots, 15\}$ independently, synchronization windows are automatically determined using statistical leakage information obtained from Welch’s t -test [Wel47]. The set T is partitioned into two subsets, T_1 and T_2 , according to the Hamming weight, HW, of the b th S-box output byte:

$$T_1 = \{T_i \in T \mid \text{HW}(v_{b,i}) < 4\}, \quad T_2 = \{T_i \in T \mid \text{HW}(v_{b,i}) > 4\},$$

where $v_{b,i}$ is the output value of the b th S-box associated with the trace T_i .

The absolute value of the Welch t -statistic is Gaussian-smoothed and prominent local maxima are subsequently identified using peak detection based on peak prominence and minimum separation. Each detected peak defines a synchronization center. Starting from each center, the window is expanded in both directions until the smoothed statistic falls below a fixed fraction of the peak amplitude. An additional padding margin is then applied to account for small variations in the extent of the informative region. The resulting interval defines the maximum synchronization window associated with that center.

For each detected center, a synchronization window is constructed by limiting its half-width to both the detected maximum half-width and half the distance to the neighboring synchronization centers. This restriction prevents the windows associated with adjacent

¹Obviously, it is possible to utilize the leakage from round 10 as well by training a separate set of neural networks on trace segments corresponding to this round and incorporating the resulting evidence into the hybrid attack. We did not pursue this, since the attack success rates obtained from rounds 1–9 alone already leave relatively little room for improvement from an additional evidence source.

centers from overlapping. Within each window, every trace is aligned independently to a reference trace by maximizing the cross-correlation between the corresponding mean-centered windows.

The resulting local shift is applied to the corresponding source samples. The valid translated samples are accumulated over the synchronization window, and the final synchronized samples are obtained by averaging the accumulated values where multiple contributions are present. Samples outside the selected synchronization windows are discarded.

To introduce diversity among the trace sets used for profiling, we repeat the fine-grained synchronization procedure seven times using slightly different synchronization parameters, producing seven distinct synchronized trace sets for each byte. Each trace set is subsequently used to train a separate neural network, as described in the next section.

The fine-grained synchronization method described above combines well-known statistical leakage detection and correlation-based trace alignment techniques into a leakage-guided piecewise synchronization procedure. The Welch t -statistic is used to locate synchronization centers and the associated synchronization windows, while cross-correlation is used to find the best local alignment within each of the multiple windows. The method differs from the synchronization approaches based on statistical moments [DSN⁺11], wavelet-domain processing [PPS22], or elastic [SC78] and sequence-based alignment [UAV⁺24]. It is worth mentioning that Dynamic Time Warping (DTW) [SC78] was also evaluated as an alternative alignment method. However, preliminary experiments showed that it reduces exploitable leakage information, so it was not included in the final procedure.

7.4 Profiling Phase

The synchronized traces are used to train neural networks. For each byte $b \in \{0, 1, \dots, 15\}$, seven multilayer perceptrons (MLPs) with the architecture shown in Table 1 are trained, each on a differently synchronized trace set, and combined into an ensemble, E_b . Training the models on differently synchronized versions of the traces introduces diversity among the ensemble members, which helps reduce correlated prediction errors. In our experiments, ensembles whose models were trained on trace sets obtained using the same synchronization parameters showed worse accuracy.

For each targeted S-box output, the ensemble produces a probability distribution over its 256 possible values by averaging the log-probabilities assigned by the individual models and subsequently normalizing the result. This combination emphasizes output values that receive consistently high probability across the ensemble while suppressing values assigned low probability by one or more models.

The MLP takes as input a trace representing a single AES round. The input is first normalized using batch normalization and then passed through two fully connected hidden layers with 512 and 256 neurons, respectively. Each hidden layer uses He-uniform weight initialization, L2 weight regularization, followed by layer normalization and a ReLU activation. The final layer is a fully connected layer with 256 neurons and a softmax activation, producing a probability distribution over the 256 possible S-box output byte values. The network is trained using the AdamW optimizer with a learning rate of 10^{-3} and weight decay of 10^{-5} , using categorical cross-entropy as the training objective.

We also evaluated other types of neural networks, including convolutional neural networks (CNNs) and transformers. However, we found that investing additional effort in trace synchronization and subsequently using a relatively simple MLP consistently produced the best results. Given the relatively large input dimension of each trace (1,080 samples), CNNs and transformers require substantially larger training datasets to outperform the MLP in this setting.

Table 1: Architecture of MLPs for predicting S-box output byte values in rounds 1–9.

Layer	Output size	Parameters
Input	1080	–
Batch Normalization	1080	4,320
Fully Connected	512	552,960
Layer Normalization	512	1,024
ReLU	512	–
Fully Connected	256	131,072
Layer Normalization	256	512
ReLU	256	–
Output (Softmax)	256	65,792
Total trainable parameters		753,520
Total parameters		755,680

7.5 Attack Phase

During the attack phase, a single power trace is collected from the target USIM executing MILENAGE with the fixed K and OPc unknown to the attacker and a challenge $RAND$ known to the attacker. The segment corresponding to the initial AES-128 invocation within MILENAGE is located using cross-correlation with the same ten-round template trace as at the profiling stage. Then, for each byte $b \in \{0, \dots, 15\}$, the trace segments corresponding to rounds 1–9 are located, extracted and synchronized using the same piecewise synchronization method as at the profiling stage, but with the synchronization centers and windows derived from the profiling traces.

This is repeated seven times for the same seven sets of synchronization parameters as during profiling, producing seven distinct synchronized trace sets for each byte, with nine single-round traces in each set. Each of these nine traces is subsequently processed by the corresponding neural network in the ensemble E_b to predict probability distributions for the b th S-box output byte values in each round $r \in \{1, 2, \dots, 9\}$. This is repeated for all bytes $b \in \{0, 1, \dots, 15\}$, resulting in 144 distributions in total. These distributions are then used in ASCA, SASCA and hybrid attacks to recover K and OPc as described in Sections 4, 5 and 6 respectively.

8 Experimental Results

We evaluated ASCA, SASCA, and the hybrid attack on a set of attack traces acquired and pre-processed as described in Section 7. Success is defined identically for all three methods: an attack succeeds if and only if both the recovered K and the recovered OPc match the ground truth. Because of ASCA’s potentially higher execution time, it is evaluated on the first 100 of the 1,000 available attack traces; SASCA and the hybrid attack are evaluated on all 1,000 traces.

The MiniSat [ES04] solver with the timeout set to 1 hour is invoked to find a satisfying assignment for ASCA and hybrid attacks.

ASCA. Table 2 summarizes the ASCA evaluation results for 100 single-trace attacks. The success rate and runtime are reported as functions of the number of S-box constraints, n , provided to the SAT solver. The results are obtained using the following decreasing- n retry procedure.

We first applied ASCA using the $n = 100$ highest-confidence S-box predictions derived from the neural networks to each of the 100 attacks. Only two attacks were successful. We then reduced n to 90 and repeated the attack for the remaining 98 unsuccessful traces,

Table 2: ASCA success rate and average per-attack runtime as functions of the number of S-box constraints n provided to the SAT solver (for 100 single-trace attacks). Reported success rates are cumulative over a decreasing- n retry procedure: only traces unsuccessful at a given n are retried at the next lower value.

S-box constraints, n	100	90	80	70	60
Success rate	2%	7%	19%	30%	34%
Runtime*, seconds	7.95	8.11	50.78	130.9	262.2

*Averaged over successful attacks, unsuccessful attacks reach the 1-hour timeout; includes the 7.82-second MLP models loading time.

resulting in five additional successful attacks and an overall success rate of 7% for $n \geq 90$. This was continued in steps of 10 until $n = 60$, at which point the cumulative success rate reached 34%. Reducing the number of constraints further to $n = 50$ produced no additional successful attacks within the 1-hour timeout.

The runtimes reported in Table 2 are averaged over successful attacks only; all unsuccessful attacks reach the 1-hour timeout. The runtimes include the 7.82-second time required to load the MLP models. Since seven models are loaded for each of the 16 bytes, 112 models in total are loaded. We can see that ASCA can complete in less than 10 seconds when 90 or more S-box predictions are provided to the SAT solver. However, the runtime increases by orders of magnitude as the number of constraints is reduced.

It is important to note that the results in Table 2 are based on only 100 attacks and therefore have substantially greater statistical uncertainty than the results for the SASCA and hybrid attacks reported below, which are evaluated on 1,000 attacks (a 95% Wald confidence interval for a 34% success rate over 100 trials is approximately $34\% \pm 9.3$ percentage points). The reported success rate should therefore be interpreted as the best estimate obtainable under the adopted timeout budget.

SASCA. Table 3 summarizes the SASCA evaluation results for 1,000 single-trace attacks. The success rate and runtime are reported as functions of the number of BP iterations.

The success rate is only 35.1% when BP runs at 8 iterations, but it increases to 87.7% at 16 iterations (a 95% Wald confidence interval for this estimate is $87.7\% \pm 2.0$ percentage points), and then flattens. At 1,000 iterations, the success rate remains 87.7%, suggesting that the remaining failures are not due to slow convergence, but reflect a real limit on what BP can recover from the available leakage. At 16 iterations, SASCA’s average runtime is 12.49 s per attack.

Of SASCA’s 123 failures, 61 of 1,000 attacks (6.1%) recover K correctly but OPc incorrectly (the asymmetric-redundancy failure mode) and 62 (6.2%) recover both K and OPc incorrectly.

Hybrid attack. Table 4 summarizes the hybrid attack evaluation results for 1,000 single-trace attacks and compares them to the results of ASCA and SASCA.

Table 3: SASCA success rate and average per-attack runtime as functions of BP iteration count (for 1,000 single-trace attacks).

Iterations	8	10	15	16	100	1000
Success rate	35.1%	74.6%	87.5%	87.7%	87.7%	87.7%
Runtime*, sec.	11.22	11.57	12.25	12.49	25.49	165.7

*Averaged over all attacks; includes the 7.82-second MLP models loading time.

Table 4: Comparison of success rates and average per-attack runtimes of ASCA, SASCA, and the proposed hybrid attack; N stands for the number of evaluated single-trace attacks.

Attack method	N	Success rate	Runtime, s.	Notes
ASCA (SAT only)	100	34.0%	262.2*	1-hour timeout
SASCA (BP only)	1,000	87.7%	12.49	16 BP iterations
Hybrid (BP+SAT)	1,000	93.8%	11.41	8 BP iterations

*Averaged over successful attacks, unsuccessful attacks reach the 1-hour timeout.

The hybrid attack recovers both secrets correctly on 938 of 1,000 attacks (93.8%, 95% Wald confidence interval $93.8\% \pm 1.5$ percentage points), averaging 11.41 s per attack, which is both a higher success rate *and* a lower average runtime than ASCA and 16-iteration SASCA used in isolation. The hybrid attack fails if and only if SASCA also fails to recover K (at any BP iteration count we tested); both occur on the same 62 of 1,000 attacks. In other words, *every one* of the 61 attacks where SASCA recovers K correctly but OPc incorrectly is resolved correctly by the hybrid attack’s SAT stage.

The runtime result is worth commenting on separately. The hybrid attack’s BP stage uses only 8 iterations, at which SASCA succeeds in only 35.1% of attacks, a 60% relative decrease from the 87.7% success rate at 16 iterations, and far below the hybrid attack’s overall 93.8%. The hybrid attack is therefore not merely SASCA followed by a "cleanup" SAT step to fix the OPc failures. Rather, the SAT compensates for a lighter BP run with fewer iterations. Because SAT resolves quickly when supplied a sufficiently large correct candidate list, the reduction in BP iterations more than offsets the added SAT runtime, yielding a lower total average runtime than the fully-converged SASCA pipeline despite performing strictly more work per attack.

Analysis of failed attacks. To understand why 62 attacks fail to recover K or OPc , we analyzed the corresponding 62 traces, together with two comparison groups: the 938 fully successful attacks, and the 61 attacks for which SASCA recovers K but fails to recover OPc . For each trace, we compared each single-round segment against the mean single-round segment of the successful-attack group, using zero-lag Pearson correlation and maximum correlation over lags within the expected jitter range.

For both groups with correct K (the successful attacks and the 61 OPc -only failures) zero-lag and best-lag correlations coincide (0.952), with zero average shift, confirming strong correlation and effective synchronization. In contrast, the 62 failed attacks have substantially lower correlation (0.895 versus 0.952). Their lag optimization yields only a negligible improvement (0.8954 to 0.8957), with an average shift of 0.014 samples. This suggests that the failures are consistent with degraded acquisition or measurement quality rather than insufficient synchronization or limitations of ASCA, SASCA, or the hybrid method.

9 Countermeasure

Motivated by the role key-schedule redundancy plays in the attacks, we propose a countermeasure based on masking each AES round key with an independent per-round secret that reduces an attacker’s advantage over guessing K and OPc to zero bits.

9.1 Construction

Let RK_r denote the round key applied by AddRoundKey at the end of round r , $r \in \{1, \dots, 10\}$. We introduce ten independent, uniformly random 128-bit masks $M_1, \dots, M_{10}$

and replace each RK_r with $RK_r \oplus M_r$ in AddRoundKey. The initial whitening relation $RAND \oplus OPc \oplus K$ in MILENAGE’s native construction is left unmodified.

Masking AES round keys with independent per-round values is not a new idea. A 2006 U.S. patent application, *Key Masking for Cryptographic Processes* [Geb06], describes masking each AES round key with a distinct mask, noting that a single mask reused across round keys leaves the master key exposed. More recently, masked AES/AES-like implementations targeting embedded platforms have adopted the same principle in their key expansion, using a different mask for each round key [AP20]. Our contribution is not the masking primitive itself, but the observation that, applied to MILENAGE, round key masking eliminates the specific key-schedule redundancy that single-trace attacks rely on. To the best of our knowledge, this connection has not previously been made for MILENAGE.

9.2 Security Argument

Since each M_r is independent and uniform, and used only in $RK_r \oplus M_r$, the masked round keys are jointly independent of K :

$$I(K; RK_1 \oplus M_1, \dots, RK_{10} \oplus M_{10}) = 0,$$

where I denotes Shannon mutual information. For every round $r \in \{1, 2, \dots, 9\}$ and byte $b \in \{0, 1, \dots, 15\}$, the observed $SboxIn_{r,b}$ variable is a deterministic function of $RAND$, $OPc \oplus K$ and the masked round keys alone, so it also carries no information about K beyond $OPc \oplus K$. Only MILENAGE’s native initial round relation $OPc \oplus K = RAND \oplus SboxIn_1$ remains unaffected by the masking, and it constrains only that XOR, leaving K and OPc individually undetermined regardless of leakage quality or inference method.

9.3 Empirical validation

Using the hybrid attack as a worst-case, strongest-attacker baseline, we validated the argument above in two ways. On synthetic traces with near-ideal, high-confidence evidence at every round, the masked factor graph never recovered K across repeated trials with independently sampled masks and secrets. We also ran the full hybrid attack on the masked graph on the same 1,000 traces used in Section 8 (with all masks fixed to all-zero, so the underlying traces are unchanged and only the target relation is modified); the attacker’s factor graph and CNF encoding treat each mask as an independent unknown variable, exactly as they would for a nonzero value, so this setting isolates the effect of the additional XOR relations rather than granting the attacker any knowledge of the masks. The hybrid attack recovered neither K nor OPc on any trace.

9.4 Deployment considerations

Because the construction changes the effective AES computation, it requires coordinated support on both the USIM and the network-side authentication center, and the masks $M_1, \dots, M_{10}$ must be fixed, reproducible per-device values for AKA’s determinism requirement. They are therefore additional long-term per-device secret material, analogous to OPc , adding $10 \times 128 = 1,280$ bits of storage and ten 128-bit XORs per AES evaluation. This makes the construction a modified MILENAGE variant rather than a firmware-only patch to deployed cards, conceptually similar to the existing configurability of $r_1, \dots, r_5$ and $c_1, \dots, c_5$, though serving a different purpose.

9.5 Practical alternative

In the construction above, the ten independent masks introduce 1,280 bits of additional per-device secret information, even though the underlying AES key only offers a 128-bit security level. A more practical alternative would be to derive the masks pseudorandomly, for example by evaluating AES in counter mode (CTR) under the same key K . A device-specific value such as the International Mobile Subscriber Identity (IMSI) could be used as a basis for the counter mode initialization value to ensure that the generated masks are unique per device. The masks would be generated once at the manufacturing stage and stored in the USIM. Generating them on-the-fly would expose the mask-generation computation to a side-channel attack.

Such a construction does not admit the same simple information-theoretic argument, since the masks are no longer independent random variables but deterministic functions of a secret key. Nevertheless, applying the attack method presented in this paper would require incorporating the AES-CTR-based mask-generation computation into the factor graph and SAT instance, rather than treating the ten round masks as independent unknowns. We conjecture that this substantially increases the complexity of the resulting inference problem and makes the attack infeasible in practice. If the construction with AES-CTR-generated masks were breakable by the presented attack, while the same construction with uniformly random masks were not, then the attack could serve as a randomness-distinguisher for AES-CTR. We leave the formal analysis of such a derived-mask construction, including whether its security can be reduced to standard pseudorandomness assumptions for AES, for future work.

10 Related Work

This section describes prior work on ASCA, SASCA, combined ASCA and SASCA attacks, side-channel attacks on MILENAGE, and related countermeasures.

10.1 Algebraic Side-Channel Analysis

Potlapally et al. [PRR⁺06, PRR⁺07] presented early work on combining satisfiability solving with side-channel cryptanalysis. They proposed a satisfiability-based framework in which the computation of a cryptographic software implementation is represented using Boolean constraints, while side-channel observations are used to restrict the possible values of intermediate variables. Their experiments on DES and 3DES demonstrated that information about intermediate values can significantly reduce the key-recovery problem.

The term *algebraic side-channel attacks* (ASCA) was introduced by Renaud and Standaert in [RS10]. Using a template attack to recover Hamming weights of intermediate values, they demonstrated single-trace key recovery against PRESENT exploiting information from multiple intermediate computations. Their approach can operate without a known plaintext/ciphertext pair, illustrating the advantage of combining side-channel information with the exact algebraic structure of the target algorithm.

Renaud, Standaert, and Veyrat-Charvillon subsequently investigated the application of ASCA to AES [RSVC09]. Using the same template-attack methodology to recover Hamming weights, they showed that an unprotected AES implementation on an 8-bit controller can be attacked using leakage from different AES rounds, with the secret key recovered from a single encryption. Their results are particularly relevant to our work because they demonstrate that the diffusion and key-schedule relations of AES provide useful redundancy for ASCA. Our setting differs in that the plaintext $RAND \oplus OPc$ is unknown not because it is withheld from the attacker but because it depends on the secret OPc itself.

A limitation of the original ASCA formulation is that side-channel predictions are ultimately converted into hard constraints. Oren et al. [OKPW10] studied this problem in the presence of measurement errors and introduced *Tolerant Algebraic Side-Channel Analysis* (TASCA). Instead of requiring every leakage prediction to be correct, their approach formulates the attack as a pseudo-Boolean optimization problem and explicitly accounts for erroneous observations. They demonstrated successful single-trace attacks against KeeLoq and AES even with substantially higher error rates than those tolerated by a direct hard-constraint formulation. This work highlights a central tension in algebraic attacks: side-channel information is valuable as a constraint, but an incorrect hard constraint can prevent the solver from recovering the correct secret.

The recent ASCA attack on AES presented in [Dub25] encodes an AES implementation as a SAT instance and uses deep-learning-based power analysis to obtain information about S-box input and output byte in multiple rounds. The results demonstrate that neural network predictions can be effectively used as constraints for a SAT solver, enabling single-trace key recovery from an AES implementation on an ATxmega128D4 microcontroller.

Most closely related to our work is [DJB26], which presents the first ASCA attack on MILENAGE. It provides an important foundation for the ASCA component of our approach: we use the same CNF encoding of AES/MILENAGE and follow the same attack strategy, while introducing adaptations to address the more challenging cross-device setting. In [DJB26], neural networks are used to predict the output bytes of the targeted AES S-boxes across nine rounds, and the resulting predictions, ranked according to their posterior probabilities, are incorporated as constraints into a SAT model of MILENAGE. The attack recovers K and OPc from a single trace with an average success probability of 40%. The experiments use traces acquired from three USIMs, with a 10,000-trace test set held out from the same three devices used for training. Our work builds on this approach in several respects. First, DTW is used for trace synchronization in [DJB26], while we employ the leakage-guided piecewise synchronization procedure, which we found to give better results than DTW in our experiments. Second, a single MLP predicts the S-box output values for each byte $b \in \{0, 1, \dots, 15\}$ in [DJB26], while we use an ensemble of seven MLPs trained on differently synchronized trace sets. This improves prediction accuracy, which is important in a cross-device setting.

10.2 Soft-Analytical Side-Channel Analysis

Soft-analytical side-channel attacks (SASCA) were introduced by Veyrat-Charvillon, Gérard, and Standaert [VCGS14] as a means of combining noisy side-channel information with the global algorithmic structure exploited by algebraic attacks. The underlying inference procedure in SASCA is commonly formulated using the sum-product algorithm on the factor graph representing the cryptographic computation [KFL01].

SASCA has been made widely accessible through the open-source tool SCALib [CB23] which provides a general-purpose, high-performance factor graph and BP implementation for side-channel analysis, along with profiling and evaluation primitives. SCALib has been adopted in subsequent SASCA-based attacks, e.g. [BCS21]. Our SASCA implementation is built independently of SCALib, motivated by the coupling required between the BP and SAT stages in the hybrid attack.

A limitation of conventional SASCA is that inference is typically performed using loopy BP, which provides no general guarantee of convergence or of obtaining the exact posterior distribution. Wedenig et al. [WNC⁺25] addressed this limitation with *Exact Soft Analytical Side-Channel Attacks* (ExSASCA), which uses knowledge compilation and tractable probabilistic circuits to perform exact probabilistic inference. They also observed that conventional SASCA’s treatment of MixColumns, typically decomposed into per-byte or per-bit factors, produces a densely-cyclic sub-graph of up to 168 binary variables, and resolved this by compiling MixColumns into a single exact factor rather than a set of loopy

approximate ones, a strategy that parallels our own joint MixColumns factor, though implemented exactly rather than within loopy BP. For AES-128, ExSASCA is shown to achieve substantially higher key recovery rates than conventional SASCA while requiring computational effort close to that of approximate inference.

10.3 Comparing and Combining BP and SAT

The complementary nature of SASCA and ASCA has been noted in prior work, though not exploited by a single combined attack. Grosso and Standaert [GS15] compared SASCA with ASCA and differential power analysis (DPA) with enumeration, and showed that SASCA can achieve more efficient key recovery than ASCA even in noise-free simulated settings, while also demonstrating the practical applicability of SASCA to an actual AES implementation and its ability to substantially reduce the number of traces required for key recovery compared with profiled DPA. Of particular relevance to our setting, their analysis also considered scenarios with unknown plaintexts or ciphertexts, illustrating that exploiting algorithmic dependencies can be especially valuable when conventional statistical attacks cannot directly rely on known input or output values.

Closer to our setting, Strieder et al. [SIH⁺23] compared ASCA and SASCA specifically on AES key-schedule recovery, using real single-trace measurements from the 2018 CHES capture-the-flag challenge. They report that loopy BP can struggle to handle the XOR operations of the key schedule reliably, and propose graph restructuring and adapted input distributions to mitigate this. Our SASCA implementation faces the same class of XOR-handling difficulty, which we address instead through a division-free WHT message-passing scheme; we view the two as complementary practical fixes to the same underlying numerical issue. Unlike [SIH⁺23], which targets key recovery from key-schedule redundancy alone under a known-plaintext model, our setting additionally requires joint recovery of a second, non-redundant secret, OPc , from a single trace. Both [GS15] and [SIH⁺23] establish that ASCA and SASCA can fail for structurally different reasons on the same target, but neither constructs an attack in which the two methods' outputs feed into one another.

The general principle of using BP to guide, rather than replace, an exact solver has also been investigated outside the side-channel analysis setting. In the study of random constraint satisfaction and k -SAT, *BP-guided decimation* algorithms [MRTS07] compute BP marginals, fix the value of one or a few variables with the most concentrated belief, simplify the resulting formula, and repeat, eventually handing a reduced instance to an exact solver. Related *survey propagation*-guided decimation methods [MPZ02, BMZ05] have solved random k -SAT instances at clause densities beyond the reach of local-search solvers alone. Our work builds on the same premise, that approximate message passing and exact search address complementary weaknesses, but is, to our knowledge, the first to combine SASCA-style BP refinement with ASCA-style exact SAT solving into a single side-channel attack, and to apply this combination to a cryptographic secret-recovery problem rather than a random CSP.

10.4 Side-Channel Attacks on MILENAGE

MILENAGE is known to be vulnerable to power analysis attacks on USIM cards, see [Kip25] for an excellent overview. Liu et al. [LYS⁺15] performed DPA on implementations of MILENAGE in 3G/4G USIM cards and demonstrated recovery of K and OPc . Their attacks are based on statistical analysis of multiple (several hundreds) power traces and therefore differ fundamentally from the single-trace setting considered in this paper.

Devine, San Pedro, and Thillard [DPT18] further investigated the practical feasibility of DPA against USIM cards using low-cost equipment. Their study examined nine USIM cards and found one to be vulnerable. Their work demonstrates that the AES executions

of MILENAGE can be identified and targeted in practical measurements, but the attack remains based on multi-trace statistical analysis.

Brisfors et al. [BFD20] demonstrated that deep learning can substantially reduce the number of traces required for attacks on USIM cards (four traces on average to recover a key byte in a cross-device setting). Their results show that neural networks can extract useful information from the AES computations even when traditional correlation-based techniques are less effective. This provides an important practical foundation for the approach taken in our work: neural networks can serve as the front end that converts a single power trace into probabilistic predictions of internal AES values, while a separate inference procedure exploits the cryptographic dependencies between those values.

The attacks [LYS⁺15, DPT18, BFD20] separate MILENAGE key recovery into two stages. First, $K \oplus OPc$ is recovered by power analysis using leakage from the first round. Knowledge of $K \oplus OPc$ makes it possible to compute the AES state at the output of MixColumns in the first round. With this knowledge, at the second step, the first round key, RK_1 , is recovered by power analysis from the leakage in the second round, and then inverted through the AES key schedule to obtain K . Once both $K \oplus OPc$ and K are known, OPc follows from

$$OPc = (K \oplus OPc) \oplus K.$$

This strategy relies on recovering sufficiently accurate intermediate information at each stage and, in the conventional power analysis setting, on aggregating information from multiple executions. In contrast, our objective is to recover both K and OPc from a *single* execution.

10.5 Countermeasures

Masking is a standard countermeasure against power analysis attacks on AES: each key-dependent variable is split into randomized shares such that no individual share, and hence no single leakage point, reveals the secret, with security provable up to a chosen statistical order [RP10]. *Hiding* countermeasures such as operation shuffling pursue a different goal, reducing the signal-to-noise ratio of leakage rather than eliminating its statistical dependence on the secret, and have been shown to require care in their evaluation, since a poorly analyzed shuffling scheme can provide substantially less protection than intended [VCMKS12].

Despite the availability of countermeasures, prior analyses of commercial MILENAGE implementations report that deployed USIM cards frequently lack state-of-the-art countermeasures altogether. Maghrebi and Bringer [MB17] studied side-channel analysis of TUAK, MILENAGE’s Keccak-based counterpart, and observed that operator-defined secret parameters do not by themselves improve implementation security against power analysis. Where countermeasures are present, they are not always effective against a sufficiently strong adversary: Jin et al. [JZQ⁺22] reverse-engineered a Rotating S-boxes Masking (RSM) countermeasure on a real commercial 4G USIM card using side-channel information alone, and recovered the underlying secrets. These works demonstrate that countermeasures for MILENAGE implementations are both under-deployed and, when deployed, not always robust to reverse engineering by a determined attacker.

Our proposed countermeasure differs from the prior approaches in both target and guarantee. Rather than masking AES’s internal S-box computations, we mask the round keys themselves with independent per-round values, mirroring MILENAGE’s native $K \oplus OPc$ construction. Because the guarantee it provides concerns the information available to an attacker about K and OPc individually, its security analysis holds unconditionally, independent of leakage quality or inference method.

11 Conclusion

We implemented and evaluated ASCA, SASCA, and a hybrid BP-SAT attack on the AES-based MILENAGE authentication algorithm deployed on USIM cards, in a single-trace, cross-device setting. Unlike a conventional known-plaintext AES attack, MILENAGE requires an attacker to jointly recover two 128-bit secrets, the key K and the operator parameter OPc , and only K benefits from the redundancy provided by the AES key schedule. This asymmetry produces the complementary failure modes of ASCA and SASCA which we observed experimentally.

The hybrid attack, in which BP-refined S-box predictions are supplied to a SAT solver, succeeds on 93.8% of the 1,000 evaluated traces, resolving all 6.1% of SASCA’s OPc failures. It has a higher success rate than either method alone, while using only half as many BP iterations as SASCA. Despite performing strictly more work per attack than either standalone method, it also achieves a lower average runtime, since BP-refined candidates let SAT resolve quickly.

Motivated by the role key-schedule redundancy plays, we proposed masking each AES round key with an independent per-round value, mirroring MILENAGE’s native $K \oplus OPc$ construction. This reduces an attacker’s information about K and OPc individually to zero bits regardless of leakage quality or inference method.

More broadly, our results suggest that evaluating single-trace resistance solely under one inference paradigm, purely algebraic or purely probabilistic, can understate an implementation’s actual exposure: a target resistant to SASCA alone or ASCA alone may still be vulnerable to an attack that combines both. We expect this methodology to be applicable beyond MILENAGE, to other constructions in which a public key schedule provides redundancy for only one of several jointly-required secrets. Future work includes investigating whether the BP-SAT combination generalizes to single-trace attacks on other cryptographic algorithms.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Research Council (Grant No. 2025-05080).

References

- [3GP99] 3GPP. Specification of the MILENAGE algorithm set; An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5, f5*; Document 2: Algorithm specification. Technical Specification TS 35.206, 3rd Generation Partnership Project (3GPP), 1999.
- [AP20] Alexandre Adomnicaï and Thomas Peyrin. Fixslicing AES-like ciphers: New bitsliced AES speed records on ARM-Cortex M and RISC-V. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):402–425, December 2020. Artifact available at <https://artifacts.iacr.org/tches/2021/a6>.
- [BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. *Cryptology ePrint Archive*, Paper 2021/817, 2021.

- [BFD20] Martin Brisfors, Sebastian Forsmark, and Elena Dubrova. How deep learning helps compromising USIM. In *Proc. of the 19th Smart Card Research and Advanced Application Conference (CARDIS'2020)*, Nov. 2020.
- [BMZ05] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 09 2005.
- [CB23] Gaëtan Cassiers and Olivier Bronchain. SCALib: A side-channel analysis library. *Journal of Open Source Software*, 8(86):5196, 2023.
- [CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
- [DJB26] Elena Dubrova, Sönke Jendral, and Linus Backlund. Single-trace SAT-assisted side-channel attack on MILENAGE. In *ACM Workshop on Assisted Cryptographic Technologies for Implementation Verification (ACTIVE 2026)*. ACM, 2026.
- [DPT18] Colin Devine, Maria S. Pedro, and Aurélien Thillard. A practical guide to differential power analysis of USIM cards. In *SSTIC*, 2018.
- [DSN⁺11] Nicolas Debande, Youssef Souissi, Maxime Nassar, Sylvain Guilley, Thanh-Ha Le, and Jean-Luc Danger. Re-synchronization by moments: An efficient solution to align side-channel traces. In *IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2011.
- [Dub25] Elena Dubrova. Solving AES-SAT using side-channel hints: A practical assessment. In *2025 IEEE 55th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 147–152, 2025.
- [ES04] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, pages 502–518, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [Geb06] Catherine Gebotys. Key masking for cryptographic processes. US Patent Application US20060256963A1, Nov. 2006. Filed May 13, 2005; published November 16, 2006.
- [Gro20] Meel Group. aes-cnf-gen: CNF generator for AES. <https://github.com/meelgroup/aes-cnf-gen>, 2020. Accessed: 2026-09-08.
- [GS15] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with enumeration: Which one beats the other and when? Cryptology ePrint Archive, Paper 2015/535, 2015.
- [JZQ⁺22] Chengbin Jin, Yongbin Zhou, Xinkuan Qiu, Qi Feng, and Qian Zhang. Breaking real-world COTS USIM cards with unknown side-channel countermeasures. *Comput. Secur.*, 113(C), February 2022.
- [KFL01] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *Transactions on Information Theory*, 47(2):498–519, Sept. 2001.

- [Kip25] Daniel Kipp. SoK: Extracting secrets from UICCs: How to clone SIM cards in 2025, 2025. Accessed: 2026-07-24.
- [LYS⁺15] Junrong Liu, Yu Yu, François-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie. Small tweaks do not help: Differential power analysis of MILENAGE implementations in 3G/4G USIM cards. In *Computer Security – ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, page 468–480, Berlin, Heidelberg, 2015. Springer-Verlag.
- [MB17] Houssem Maghrebi and Julien Bringer. Side-channel analysis of the TUAK algorithm used for authentication and key agreement in 3G/4G networks. In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications*, pages 39–56, Cham, 2017. Springer International Publishing.
- [MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering (SPACE 2016)*, pages 3–26. Springer International Publishing, 2016.
- [MPZ02] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 08 2002.
- [MRTS07] Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. Solving constraint satisfaction problems through belief propagation-guided decimation. In *45th Annual Allerton Conference on Communication, Control, and Computing*, volume 1, pages 352–359, 9 2007.
- [Nat01] National Institute of Standards and Technology (NIST). FIPS PUB 197: Advanced encryption standard (AES), 2001. Appendix A: Key Expansion Examples.
- [OKPW10] Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. Algebraic side-channel analysis in the presence of errors. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 428–442, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [PPS22] Saravanan Paramasivam, Srividhyaa Alamelu PL, and Prashanth Sathyamoorthi. Maximal overlap discrete wavelet transform-based power trace alignment algorithm against random delay countermeasure. *ETRI Journal*, 44(3):512–523, 2022.
- [PRR⁺06] N.R. Potlapally, A. Raghunathan, S. Ravi, N.K. Jha, and R.B. Lee. Satisfiability-based framework for enabling side-channel attacks on cryptographic software. In *Proceedings of the Design Automation & Test in Europe Conference*, volume 2, pages 1–6, 2006.
- [PRR⁺07] Nachiketh R. Potlapally, Anand Raghunathan, Srivaths Ravi, Niraj K. Jha, and Ruby B. Lee. Aiding side-channel attacks on cryptographic software with satisfiability-based analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(4):465–470, 2007.

- [RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [RS10] Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology, Inscrypt 2009*, pages 393–410, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [RSVC09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: Why time also matters in DPA. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 97–111, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
- [SIH⁺23] Emanuele Strieder, Manuel Ilg, Johann Heyszl, Florian Unterstein, and Silvan Streit. ASCA vs. SASCA: A closer look at the AES key schedule. In *Constructive Side-Channel Analysis and Secure Design: 14th International Workshop, COSADE 2023, Munich, Germany, April 3–4, 2023, Proceedings*, page 65–85, Berlin, Heidelberg, 2023. Springer-Verlag.
- [UAV⁺24] Heitor Uchoa, Vipul Arora, Dennis Vermoen, Marco Ottavi, and Nikolaos Alachiotis. Exploring genomic sequence alignment for improving side-channel analysis. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *Computer Security – ESORICS 2023*, pages 203–221, Cham, 2024. Springer Nature Switzerland.
- [VCGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 282–296, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [VCMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazuo Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 740–757, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [Wel47] Bernard L. Welch. The generalization of Student’s problem when several different population variances are involved. *Biometrika*, 34(1–2):28–35, 1947.
- [Wel23] Harald Welte. *sysmoUSIM-SJS1 / sysmoISIM-SJA2 User Manual*. sysmocom - systems for mobile communications GmbH, 2023. Revision v11, October 2023.
- [WNC⁺25] Thomas Wedenig, Rishub Nagpal, Gaëtan Cassiers, Stefan Mangard, and Robert Peharz. Exact soft analytical side-channel attacks using tractable circuits. ArXiv, Report 2501.13748, 2025.