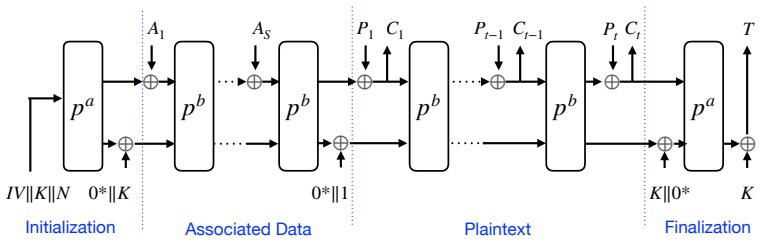
A Template Attack on ASCON AEAD

Shih-Chun You, Markus G. Kuhn (University of Cambridge)
Sumanta Sarkar, Feng Hao (University of Warwick)

We present a profiled side-channel power-analysis attack, combining the techniques of Fragment Template Attack, Belief Propagation, and Key Enumeration, which recovers the key used by a 32-bit implementation of ASCON AEAD, a NIST Lightweight Cryptography candidate. We achieve success rates close to 100% even with only a single trace. However, these results depend significantly on the optimization settings when compiling the target. With some options we instead require about 10 traces for a high success rate. We also achieve a high success rate for a masked implementation, using 100 traces for the same key, with some additional constraints.

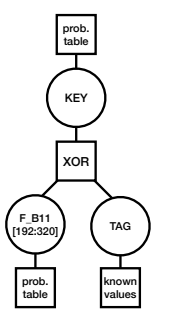
The ASCON cipher suite provides duplex-sponge-based algorithms for authenticated encryption with associated data (AEAD), built using an SPN-based lightweight 320-bit permutation round p that is called a or b times:



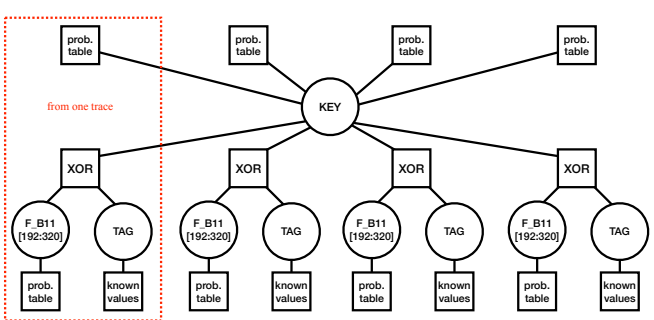
Attack strategy: Our setting is a profiled fixed-length known plaintext attack. In the profiling stage, the attacker knows the plaintexts P, associated data A, keys K, nonces N, ciphertexts C, and tags T, along with recorded power traces. In the attack stage, only the tags T and recorded power traces are needed to recover the secret key K.

The attack stage consists of three main steps:

1. Firstly, in a *fragment template attack* [1], we obtain probability tables of every candidate for target (8-bit or 16-bit) fragments of the key and for the output of the permutation p^a in the Finalization step.
2. In the second step we perform *belief propagation* (SASCA, Veyrat-Charvillon et al., ASIACRYPT 2014). Note that ASCON AEAD encryption calculates the tag T by XORing the 128-bit key K and the last 128 bits of the output of p^a in Finalization. We build a small factor graph targeting this XOR operation for single-trace belief propagation and then expand it to the factor graph for multi-trace belief propagation, for the situation that attackers record traces of multiple encryptions with the same key:



factor graph for single encryption



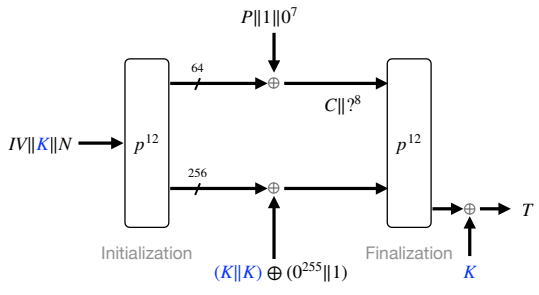
factor graph for multiple encryptions

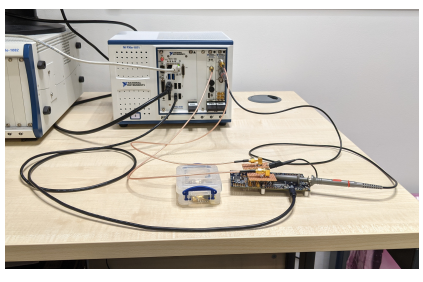
The outputs of this step are the probability tables of candidates for fragments of K, where the probability has been updated by belief propagation.

3. Finally, we apply the *key enumeration* algorithm by Veyrat-Charvillon et al. (SAC 2012) to find the correct combination of the key fragments.

Experimental setup: We targeted Rhys Weatherley et al.'s ASCON AEAD-128 ($a = 12$, $b = 8$) implementation [2] compiled onto a 32-bit ARM Cortex-M4 core (STM32F303RCT7) on a ChipWhisperer-Lite (CW-Lite) board. We recorded an AC-coupled power trace using an NI PXIe-5160 10-bit oscilloscope phase-locked to a function generator (NI PXIe-5423) supplying the target board with a 5 MHz square-wave clock signal.

In our attack demonstration, we used empty associated data and 7-byte plaintexts, to keep the traces aligned and minimize their length when covering the entire encryption process. In this case, there are only 2×12 invocations of permutation round p:



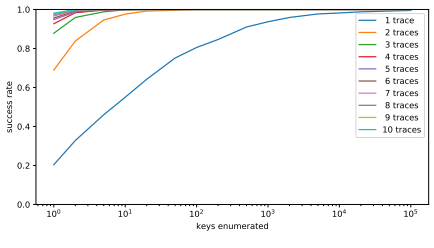


During the attack phase, we recorded traces of up to 10 encryptions, each with the same key K, but varying plaintext P. For the key enumeration step, we limited the search to enumerating up to 100 000 candidate keys when evaluating the success rate.

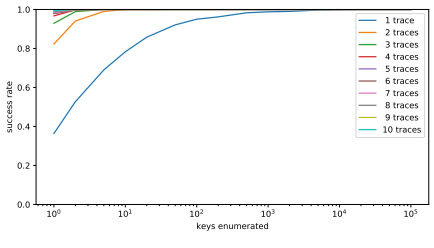
Results: We evaluated the success rate of our template attack for both 8 and 16-bit fragments, and for two target binaries, compiled with either `arm-none-eabi-gcc` (v9.2.1) compiler options `-Os` or `-O3`:

Success rates (with `gcc -Os` optimization for space):

byte fragments:



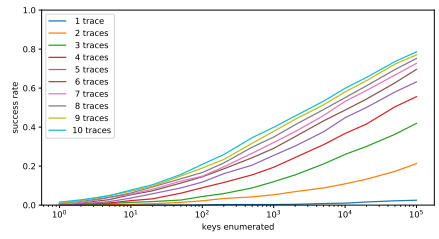
16-bit fragments:



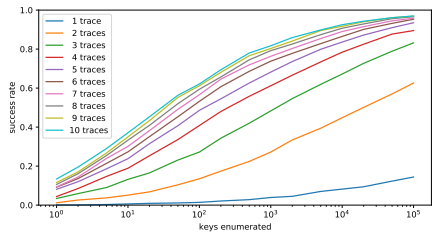
We can see that after 100 000 keys enumerated, the success rate is very close to 100%, even from single attack traces.

Success rates (with `gcc -O3` optimization for time):

byte fragments:



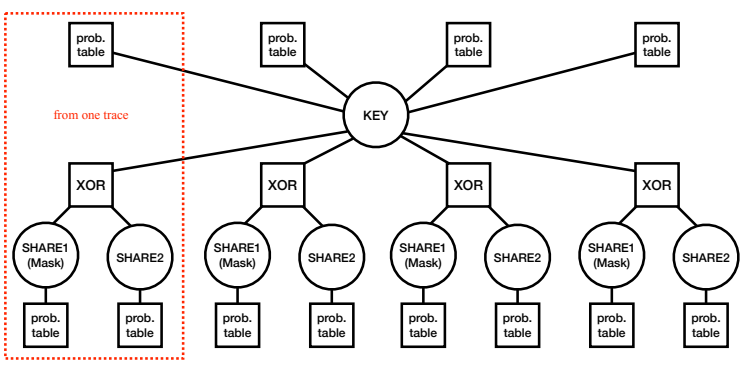
16-bit fragments:



When we compiled the target with optimization level 3, we found that it is more difficult to recover the key with single traces, but we still reach a high success rate with only a few additional attack traces for the same key K.

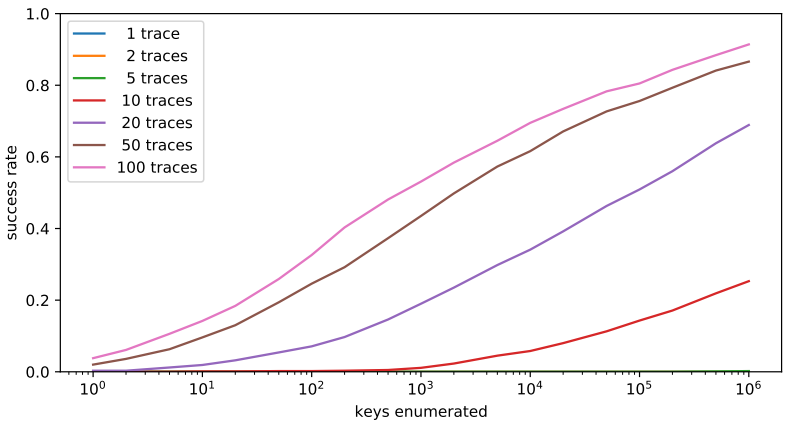
Attacking a masked version: We also performed an attack on a masked version of ASCON AEAD, from the same implementation package [2], using a similar procedure, including fragment templates, belief propagation, and key enumeration. This implementation protects the key with first-order masking, separating the key value into two shares: one is the mask provided by a pseudo-random generator (based on ChaCha), and the other is the XOR value of the key and the mask. Since this masked implementation aims to mitigate CPA/DPA-style attacks, it freshly masks the key at the start of each encryption. We target the XOR of this first-order masking step.

Our multi-trace solution assumes that attackers can access the key and the seed of the pseudo-random generator in the profiling stage, so that they can produce fragment templates for the key and its two shares. In the attack stage, we require no knowledge of any values processed. We solely use the probability tables obtained from the templates to perform a belief propagation with the following factor graph:



factor graph for multiple encryptions of masked ASCON AEAD

Note that when attacking the unmasked version, we could use 16-bit fragments, because the tag fragments were already known, and therefore saved us from iterating all their 2^{16} candidates. Here, however, all candidates for both shares need to be considered, so smaller fragments are more practical. Therefore, we provide here only the results of byte-fragment experiments, evaluated for 1000 different keys. For each key, we collect power traces of up to 100 encryptions, for multi-trace belief propagation. The following figure shows the success rate of our attack, performed on the implementation with compiler option `-Os`, enumerating up to 1 000 000 keys. The success rate is higher than 90% with 100 power traces from encryptions with the same key.



However, we only successfully recovered two keys out of 1000 attempts with compiler option `-O3`.

Conclusion: Overall, we have shown that we can attack two ASCON AEAD implementations solely by targeting XOR operations involving the key. In the case of the unmasked implementation, there are four such XORs appearing in the template for K. In the case of the masked implementation, we have only one XOR in the template of K, but four XORs for the two shares. While this reduces our success rate for a given number of traces, the attack can remain feasible even for the masked version. Compiler optimization options matter, too.

[1] SC You, MG Kuhn: Single-trace fragment template attack on a 32-bit implementation of Keccak. CARDIS 2021, DOI: 10.1007/978-3-030-97348-3_1

[2] `github.com/rweather/lwc-finalists/tree/master/src/individual/ASCON`, commit `5d2b22c`, 21 June 2021

`https://www.cl.cam.ac.uk/~scy27/ches2022-ascon-poster.pdf`

Email: scy27@cl.cam.ac.uk