EM Analysis in the IoT Context: Lessons Learned from an Attack on Thread*

Daniel Dinu[1] and Ilya Kizhvatov[2]

[1] SnT, University of Luxembourg, dumitru-daniel.dinu@uni.lu
[2] Digital Security Group, Radboud University Nijmegen, i.kizhvatov@science.ru.nl

Abstract. The distinguishing feature of the Internet of Things is that many devices get interconnected. The threat of side-channel attacks in this setting is less understood than the threat of traditional network and software exploitation attacks that are perceived to be more powerful.
This work is a case study of Thread, an emerging network and transport level stack designed to facilitate secure communication between heterogeneous IoT devices. We perform the first side-channel vulnerability analysis of the Thread networking stack. We leverage various network mechanisms to trigger manipulations of the security material or to get access to the network credentials. We choose the most feasible attack vector to build a complete attack that combines network specific mechanisms and Differential Electromagnetic Analysis. When successfully applied on a Thread network, the attack gives full network access to the adversary. We evaluate the feasibility of our attack in a TI CC2538 setup running OpenThread, a certified open-source implementation of the stack.
The full attack does not succeed in our setting. The root cause for this failure is not any particular security feature of the protocol or the implementation, but a side-effect of a feature not related to security. We summarize the problems that we find in the protocol with respect to side-channel analysis, and suggest a range of countermeasures to prevent our attack and the other attack vectors we identified during the vulnerability analysis.
In general, we demonstrate that elaborate security mechanisms of Thread make a side-channel attack not trivial to mount. Similar to a modern software exploit, it requires chaining multiple vulnerabilities. Nevertheless, such attacks are feasible. Being perhaps too expensive for settings like smart homes, they pose a relatively higher threat to the commercial setting. We believe our experience provides a useful lesson to designers of IoT protocols and devices.

Keywords: mesh network · IEEE 802.15.4 · AES-CCM · HMAC · DPA · DEMA

1 Introduction

Over the last few years we have seen a huge increase of IoT-enabled devices available on the market. These devices, intended to make our lives easier by collecting, processing and exchanging data, are manufactured by various companies around the world. To foster the development of industry-wide standards for smart devices, companies from different

*Part of the work done while the first and the second author were an intern and an employee in the Innovation Center Crypto & Security at NXP Semiconductors, respectively. The work of the first author at the University of Luxembourg was supported by the CORE project ACRYPT (ID C12-15-4009992) funded by the Fonds National de la Recherche, Luxembourg. The research of the second author at the Radboud University was funded by the Netherlands Organisation for Scientific Research (NWO) through the projects 13499 TyPhoon and 628.001.005 OpenSesame.

business fields gathered together in various working groups, organizations or consortia. Their aim is to augment the smart objects' capabilities by enhancing the communication and data exchange between devices from different makers. This is a challenging task given the heterogeneous nature of the IoT comprising a vast variety of devices, of which the overwhelming majority is characterized by a multitude of constraints such as energy or power consumption, code size and memory footprint to name a few.

The IoT ecosystem is still in its early inception stages and a lot has to be done until all smart devices can communicate seamlessly with each other. Unfortunately, given the current abundance of emerging standards for the IoT, there is little to no effort to thoroughly analyse the security of these proposals. Thus, neither the companies involved in the development of such standards, nor the end users are fully aware of the security and privacy aspects of future connected products that will flood the market in the coming years.

Attack Surface and Threats for Connected Devices. In the connected world, attacks that can be mounted remotely pose a major threat. Software exploitation and network attacks fall into this category. They require low resources (usually just a connected PC), and do not require physical proximity.

Especially in the context of the connected home, most current devices are within the home perimeter and physical access would mean that the attacker is already inside the house (assuming building access as an asset). However, devices like smart locks and cameras are on the edge or outside of the building perimeter, and thus may be physically accessed. In the near future, we will most likely see devices for outdoor lighting or garden sprinklers connected to the smart home ecosystem. An attack on one of such devices may provide an entry point to the ecosystem, potentially leading to next scalable steps.

Physical proximity attacks pose a relatively larger threat in a commercial setting, for instance a hotel where rooms are equipped with wireless door locks. If an adversary has access to such a smart lock connected to a mesh network in her room, she might be able to exploit it in a similar way to what we describe in this paper to get access to the hotel network. A recent security incident targeting the access control system of a hotel [Nov17] shows that this type of scenario is well possible; see also the physical attack on a door lock presented in [SDK+13].

Motivation. With the growing complexity of exploit-mitigation techniques, recent software attacks need to chain several vulnerabilities to succeed (e.g. [LL16]). Moreover, new network protocols are designed with security in mind.

We are driven by the curiosity to see if extending the attack surface to physical attacks, such as electromagnetic or power analysis (common in the smart card security world), would give additional benefits to the attacker that pay off the need for physical proximity. With the increased availability and reduced cost of both hardware and software tools for side-channel attacks ([O'F17], [Jls17], [Dar17] to list a few) these attacks are becoming familiar and affordable to a wide hacker community. We are interested to evaluate the realistic effort required to apply such an attack in the IoT context.

The network layer of the connectivity stack typically relies on a master key and relatively long-lived network keys to provide the first layer of defense against an attacker in the proximity of an IoT device. A question raised by the designers of IoT hardware is: Do cryptographic implementations in the network layer need protection against side-channel attacks? We have not seen a consolidated opinion on this matter, with academic experts in Differential Power Analysis (DPA) claiming attacks are possible (see related work paragraph below), while industry being on the conservative side. In our view, this disagreement is due to the lack of in-depth case studies that could demonstrate the feasibility (or infeasibility) of such attacks.

Numerous articles and marketing campaigns advertise Thread [Thr16b] as a new, efficient and secure solution for the connected home with the roadmap to expand into the commercial building and professional sectors. The claim of being *always secure* garnered our attention and made us curious to check ourselves if this claim is true, especially in view of the availability of OpenThread. The lessons learned from side-channel analysis applied to the Thread networking stack could be used to improve the overall security level of current and future protocols designed for the IoT.

Our Contribution. We perform, to the best of our knowledge, the first public side-channel vulnerability analysis of Thread. Thread is a complex networking stack and an exhaustive analysis would require a tremendous effort, especially when multidisciplinary attack vectors are considered as in our work. While providing some coverage, we focus on finding fast and effective ways to get access to a Thread network. Our contributions are:

- We perform a vulnerability analysis of Thread specifically with respect to an adversary capable to mount electromagnetic side-channel attacks. In this context, we outline several attack vectors to bypass the security mechanisms of the networking stack. In particular, we target manipulations of the security material (i.e. cryptographic keys).

- We describe a fully implemented attack that chains the exploitation of network-level mechanisms and electromagnetic side-channel analysis techniques to get unauthorized access to an existing Thread network after several hours of acquisitions in the close proximity of a Thread Router or Router-Eligible End Device that is already in the network.

- We explain that the failure of the full attack is due to a side-effect of a feature not related to security (packet fragmentation). Therefore, the protocol weaknesses we discovered are relevant.

- We describe a range of countermeasures for the protocol and for the implementation that can be applied whenever side-channel resistance is required.

We believe this case study provides a useful lesson to designers of IoT protocols and devices. Our work comes early in the life cycle of future Thread products. Because of this, we believe that it has a more profound impact, although being less impressive than breaking an off-the-shelf Thread device.

Related Work. In the past years, numerous papers addressed various aspects of IoT security. One notable direction is the analysis of the security and privacy of software frameworks for the IoT [FJP16, FPR$^+$16]. A survey of the security and privacy of implantable medical devices and body area networks is given in [RRKS14].

Though, the impact of side-channel attacks on the security of connected objects is far from being completely and clearly understood. A step towards this goal was made by the following works. de Meulenaar and Standaert [dMS10] demonstrated the feasibility of passive power analysis attacks on AES and ECC impementations in wireless sensor nodes. O'Flynn and Chen attacked the MAC layer encryption of an IEEE 802.15.4 node [OC16] using power analysis; they describe the approach and implement the basic steps but not the full attack. Their attack builds on previous works of Jaffe [Jaf07] and Kizhvatov [Kiz09]. Ronen *et al.* [RSWO17] exploited popular smart lights to create a worm capable to quickly spread an infection over large areas. Their work used power analysis to recover the global AES-CCM key used to encrypt and authenticate firmware updates.

Compared to the work described in [OC16], our attack is performed in the context of Thread. It bypasses more complex security mechanisms to affect the full Thread networking stack and not only the standalone MAC layer. By our full implementation, we demonstrate that the threat posed by the recovery of the MAC layer key largely depends on the context, specifically on the upper layers, and that there may be unexpected hurdles. Additionally, we improve the attack on AES-CCM of [OC16] by increasing the number of ciphertext bytes under our control. As a consequence, we have to attack one AES round less to recover the 16-byte key. We use EM analysis and do not rely on a dedicated trigger from the target, thus demonstrating an attack in a less invasive setting.

Therefore, to our knowledge, this work is one of the few to demonstrate an EM analysis attack in a complex wireless network setting, and to address the security of an IoT network protocol with respect to adversaries capable to mount side-channel attacks.

Responsible Disclosure. We informed the Thread Group in October 2016 about our findings and proposed countermeasures. We received a confirmation of our findings. Based on our report, the Thread Group decided to elaborate a set of recommendations for implementers in order to enhance the security of Thread products.

2 Background

We provide a minimal recap of side-channel attacks and then we give details on Thread to introduce the concepts used in our analysis. The systematization of Thread's technical details is particularly valuable considering the lack of public technical specifications.

2.1 Electromagnetic Side-Channel Attacks

Side-channel attacks exploit information gained from a device performing cryptographic operations to determine the secret value used during the observed executions. Some popular sources of side-channel information are time, cache hits and misses, power consumption or electromagnetic (EM) emissions of the target device. In this work we chose to focus on electromagnetic emissions because they can be easily acquired using a non-invasive measurement technique (i.e. by placing an EM probe in the vicinity of the target device). The attack is performed in two phases. In the acquisition phase, the attacker, equipped with a digital oscilloscope, an EM probe, and optionally an amplifier, captures the EM emissions of a device while it performs cryptographic operations for various inputs using the same secret key. Then, in the attack phase, the attacker uses statistical tools to correlate the recorded EM traces with a hypothetical model based on the known input and a key guess. The most likely key used during the observed computations will give the highest correlation value.

The Pearson correlation coefficient was initially used in the context of Correlation Power Analysis (CPA) attacks [BCO04] to measure the correlation between the acquired power traces and the hypothetical power consumption, but it can be used for EM traces as well. Indeed, the power consumption and EM emanations of a device are strongly linked. Quisquater and Samyde [QS01] and then Agrawal *et al.* [AARR02] were the first to use the EM leakage of a device as a source of side-channel information. In contrast with the classical Differrential Power Analysis (DPA) attacks [KJJ99] introduced by Kocher *et al.*, the CPA attacks are more efficient and reliable, but more computationally intensive.

Template attacks are the strongest type of side-channel attacks in an information theoretic sense [CRR02]. They assume the attacker has full access to a similar device to the one to be attacked. She uses this device to build patterns for different operations and input values. The result of this profiling phase is the so-called templates. Then, the attacker matches the recorded leakage from a limited set of observations (one or few traces) of the target device with the recorded templates to recover the secret value. However, the creation of templates is a daunting task. Experimental results show that in practice template attacks suffer from the variability caused by different devices or different acquisition campaigns [CK14]. For more details on power analysis attacks we refer the reader to [MOP07].

2.2 Thread

Supported by more than 200 companies, including most major players in the IoT arena, the Thread Group [Thr16b] is a nonprofit organisation that promotes Thread's use in connected home solutions. Thread is a network and transport level stack of protocols designed to simplify consumer lifestyles by controlling and connecting products at home. In November 2016, the Thread Group announced the expansion of Thread beyond the connected home to commercial spaces where people work [Thr16c]. Nest released an open-source implementation of Thread, called OpenThread, on GitHub [Ope16] in May 2016. As of October 2017, the OpenThread GitHub repository is supported by ten members of the Thread Group and is an important resource for hobbyists and early adopters who cannot afford the membership fee. OpenThread runs on a number of wireless hardware platforms.

Based on well-established technologies, the Thread networking stack is built on top of physical and data link layers of IEEE 802.15.4 [IEE17], operating at 250 kbps in the 2.45 GHz band [Thr16b]. Thread uses 6LowPAN to enable IPv6 addressing of up to 250 devices per network. The mesh network topology of Thread accommodates up to 32 routers to create a resilient network with no single point of failure. It provides an efficient way to forward messages between nodes using the RIPng distance vector routing protocol. For its transport layer, Thread uses UDP and DTLS. CoAP is used as application layer for the commissioning of new devices.

Thread devices are classified into two groups (see Table 1) based on power requirements and resource characteristics: Full Thread Devices (FTDs) and Minimal Thread Devices (MTDs).

- An *FTD* is usually supplied directly from the power lines, but it can also run on batteries. An FTD can have three different roles in a Thread network: Router, Router-Eligible End Device (REED), and Full End Device (FED).

- An *MTD* runs a lighter version of the Thread stack with reduced capabilities due to its limited resources; it usually runs on batteries. An MTD can have one of the following roles: Minimal End Device (MED) or Sleepy End Device (SED).

A key difference between FTDs and MTDs is that FTDs keep a communication link with neighboring Routers, while MTDs do not. A device that is not a Router is called an End Device (ED) and it is attached to a Parent with whom it communicates through a direct link.

A device attaches to a Thread network as an ED. During the lifetime of a Thread network, a device can have different roles at different moments of time. For example, a REED can become a Router if the network configuration is favorable; similarly, a Router can become a REED. A Thread network is managed by a Router autonomously elected by the network and called Leader. The Leader assigns router addresses, collects and distributes information about the network state to all Routers. If the current Leader becomes unavailable, another Router will replace it. A Thread Router having other network interfaces (Ethernet, Wi-Fi, Bluetooth, etc.) is called a Border Router. It can forward the traffic between the Thread network and other networks.

Network security is enforced at the MAC (Media Access Control)[1] and MLE (Mesh Link Establishment) layers using AES in CCM mode [Ope16]. All communication within a Thread Network is secured, except for `MLE Discovery Request` and `MLE Discovery Response` messages. Commissioning security is based on a DTLS tunnel established using elliptic curve J-PAKE and the NIST P-256 elliptic curve.

[1]Note that MAC denotes Message Authentication Code when used in cryptographic context. In this paper, to avoid confusion we use this abbreviation solely in the networking interpretation to denote Media Access Control. The only exclusion is HMAC, where MAC keeps its cryptographic meaning.

Table 1: Device types and roles in a Thread network.

Type	**Role**	**Description**	**End Device (ED)**
FTD	Router	acts as a router	✗
	REED	can act as a router	✓
	FED	will never act as a router	✓
MTD	MED	always on	✓
	SED	sleeps most of the time	✓

2.2.1 Security Material

Once successfully commissioned into a Thread network, the connected device gets the 16-byte network master key MK used to secure Thread communication and the Commissioning Key CK used to secure Thread commissioning [Ope16].

Each node keeps its own 4-byte *Sequence* counter in synchronization with neighboring devices through the use of designated fields in the security header of MAC frames (1-byte `Key Index`) and MLE messages (4-byte `Key Source`). The 1-byte `Key Index` is computed from the 4-byte *Sequence* number:

$$KeyIndex = (Sequence \wedge \texttt{0x7F}) + 1 \tag{1}$$

Thread communication is secured using a 16-byte MAC key K_{MAC} or a 16-byte MLE key K_{MLE}. These keys are derived from the 4-byte *Sequence* number concatenated with the ASCII binary representation of the string "Thread" (`0x54 0x68 0x72 0x65 0x61 0x64`) using the keyed-hash message authentication code (HMAC) function $HMAC$ under the network master key MK as described below. The hash function used is SHA-256.

$$K_{MAC} \parallel K_{MLE} = HMAC_{MK}(Sequence \parallel \text{“}Thread\text{”}) \tag{2}$$

Fresh MAC and MLE keys are generated when the default key rotation timer (set to 672 hours) expires. The *Sequence* number is incremented by one, the $KeyIndex$ value is updated, the $HMAC_{MK}$ function is executed and the key rotation timer is rearmed. When refreshing the keys, the outgoing MAC and MLE frame counters are reset to zero.

When receiving an MLE message with a different *Sequence* number set in the `Key Source` field, the receiver computes a temporary key using the received sequence as described in Equation 2. In the case of MAC frames, if the received $KeyIndex$ is not equal to the computed $KeyIndex$ from Equation 1, the receiver will generate a temporary key only when the absolute difference between the two values is one. This temporary key allows a node to synchronize with its Parent after a period of absence from the Thread network.

Each Thread node, regardless of its type and role, stores the security material and network parameters of the Thread network to its non-volatile memory to be able to rejoin the network after a reset without human intervention.

2.2.2 MLE

The Mesh Link Establishment (MLE) protocol facilitates the secure configuration of radio links and exchange of network parameters. The MLE messages are sent inside UDP datagrams with the source and destination ports set to 19788. The security of MLE messages is provided by AES in CCM mode using the MLE key K_{MLE}. The `Auxiliary Security Header`, `Source IP` address and `Destination IP` address are authenticated using a 32-bit message integrity code (MIC), while the payload is encrypted. The `Auxiliary Security Header` of an MLE message includes the 4-byte `Key Source`.

A Thread Router periodically multicasts `MLE Advertisement` messages to advertise its presence. Such a message is sent at an interval between 1 and 32 seconds after the previous advertisement was sent by the same Router. The Thread REEDs advertise their presence by multicasting a similar message every ten minutes on average.

The process of establishing a communication link between two Thread nodes N_1 and N_2 is depicted in Figure 1. In this case, the Child node N_1 is creating a communication link with its Parent N_2 in three phases: *Attaching*, *Child Synchronization*, and *Link Synchronization*. This message exchange occurs, for example, when an MTD reconnects to a Thread network.

Before initiating the MLE message exchange shown in Figure 1, the Child sends an `MLE Discovery Request` to locate the existing Thread devices. It gets in response an `MLE Discovery Response` message containing the Thread network channel number and its PAN ID.

In the *Attaching* phase, the Child (N_1) multicasts an `MLE Parent Request` message with a randomly generated 8-byte challenge. All Routers and REEDs that receive this request store the received challenge and answer with an `MLE Parent Response` message including the received challenge and a new random 8-byte challenge. N_1 selects one of the answers it receives based on the link quality and unicasts an `MLE Child ID Request` message that includes the received challenge to the corresponding node N_2. The Parent sends an `MLE Child ID Response` which may include the network configuration parameters. Then, the *Child Synchronization* takes place. The Child sends an `MLE Child Update Request` to its Parent and gets in response an `MLE Child Update Response` message.

The communication link is established in the *Link Synchronization phase*. Initially, the Child multicasts an `MLE Link Request` message containing an 8-byte random challenge. The Parent answers with an `MLE Link Accept & Request` message that includes the received challenge and a new randomly generated challenge. The Child confirms the Parent request by sending an `MLE Link Accept` message, which includes the challenge received from the Parent.

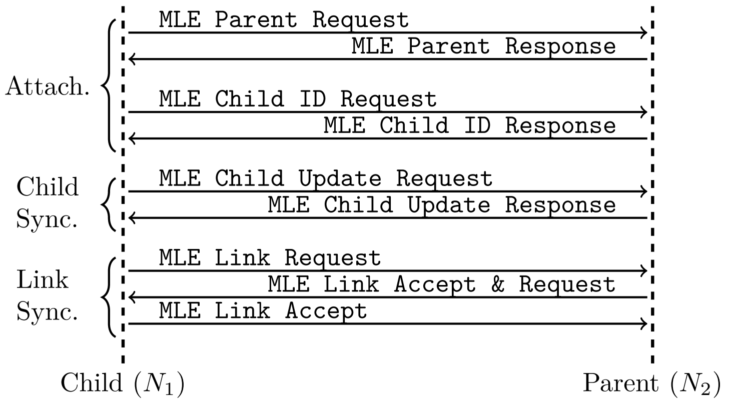


Figure 1: Establishing a communication link between two Thread nodes.

3 Threat Model

The are numerous avenues an attacker can try to compromise a Thread network. While some threats were well understood and properly mitigated in the design phase of the networking stack, others were less obvious and thus overlooked. The latter ones are harder to eliminate as the protocol becomes more mature and widely used [Mic17]. We make a classification of the possible attack types in the IoT environment, without aiming at a complete coverage of all threats specific for these settings. The threat modeling we performed for Thread can be easily adapted for other IoT protocols and the lessons learned from this study can be employed to protect other IoT solutions as well.

The goal of our threat model is to provide a classification and a better understanding of Thread's attack surface. The attacker attempts to affect one or more of the basic security functions: confidentiality, integrity, and availability. The primary goal of the attacker is to get access into a Thread network in order to intercept and understand the communication or to take control of the network. Other objectives include, but are not limited to, disrupting the normal network operation by performing a DoS attack or altering data sent across the network through a man-in-the-middle attack.

Similar to the work of Atamli and Martin [AM14], we consider three main entities that can threaten the security of an IoT system: *legitimate user*, *device maker*, and *malicious adversary*. A *legitimate user* poses a threat to the security of an IoT system when, for example, she seeks to bypass the authentication, authorization, and accounting mechanisms used by the target system. In this way, she might unlock restricted features of the device or use the existing ones without paying for them. A *device maker* can threaten the security of an IoT system either accidentally (e.g. poorly implemented security mechanism) or deliberately (e.g. aiming to collect user's data). Finally, a *malicious adversary* is the classical attacker willing to get unauthorized access to a system or to damage that system. Unlike a legitimate user, a malicious adversary may not always have proximity to the target system (consider e.g. a thermostat that is inside the house of a legitimate user).

Depending on the location of the attacker with respect to the target system, we distinguish between: *remote*, *proximity*, and *invasive* attacks. The powerful *remote* attacks are well understood from the classical Internet-connected systems. Although, some IoT protocols such as Thread assume that not all IoT devices inside a network are directly accessible from the Internet. Thus the attack surface of *remote* attacks on IoT networks is reduced compared to the attack surface of the same attacks on the classical Internet. The IoT is a highly heterogeneous environment with devices deployed in various, including distant, locations. In such settings, it is hard to enforce the physical security of these systems that become vulnerable to *proximity* and even *invasive* attacks. *Proximity* attacks can be performed without physical access to the target device and thus are harder to detect than *invasive* attacks. A summary of the attack types specific to the IoT is given in Table 2.

Table 2: Summary of the attack types specific to the IoT.

Attack Type	**Attack**	**Mitigation**
remote	software exploitation	system hardening
	guessing password	strong security policy
	brute force	
proximity	EM analysis	side-channel countermeasures
invasive	power analysis	
	fault attacks	
	flash read out	read out protection

Proximity attacks are very feasible in the IoT since most protocols use wireless communication means and are deployed in easily accessible spots. Hence, the attacker can easily get in the proximity of a Thread device and observe it performing various operations. Given the ubiquitous nature of the IoT, it is expected that the attacker is able to quickly identify such target devices. We assume the attacker can carry a portable oscilloscope and an EM probe required to perform an EM analysis attack.

We do not restrict the attacker's capabilities in terms of equipment or physical location to accurately capture the current state of security in the IoT with respect to EM analysis attacks.

4 Side-Channel Vulnerability Analysis

The goal of our vulnerability analysis was to investigate attack paths that can provide full access to a Thread network. This allows us to sniff and understand all network traffic, to add new devices into the network, and to take control of the network by changing the security material. In order to achieve this, we explored different attack vectors to recover the security material of the network. We first explore the feasibility of several active attacks paths; these are attacks in which the attacker injects packets to trigger different operations on the target nodes. Then, based on the results of the active attacks, we can estimate how successful a passive attack exploiting the same mechanism could be.

We did not look for implementation-specific issues such as buffer overflow attacks, fragmentation attacks, or improper input sanitization, because they would be meaningful only for a particular software implementation. We did not include attacks that affect the availability of the network such as denial-of-service (DoS) attacks in our scope. Below we present the most promising attack paths. Other attack paths are described in Appendix A.

4.1 Relationship between MK and K_{MLE}

Having the master key MK and *Sequence* number, a node can compute the MLE key K_{MLE} using Equation 2. If a node possesses the MLE key K_{MLE} but does not have the master key MK, it can send an `MLE Child ID Request` to ask for the network master key. Its Parent will answer with an `MLE Child ID Response`, which includes the `Master Key TLV` containing the requested master key MK. This means that MK and K_{MLE} are equivalent, in the sense that if a node has one of them, it can easily compute or retrieve the other one. Giving access to the master key to nodes having a key derived from it generates serious security issues as we will describe later.

However, this approach has a major limitation. The `Master Key TLV` is just a small fraction of the data included in the `MLE Child ID Response`. Although the attacker can ask for some specific TLVs in his request, the Parent decides the exact content of the response message. If the answer fits into a single MLE message, then the response is encrypted only at the MLE layer. As the total payload length of the `MLE Child ID Response` message exceeds the maximum transmission unit (MTU) of 127 bytes, the MLE message is fragmented at the 6LowPAN layer. When fragmentation occurs, all resulting fragments are encrypted at the MAC layer using K_{MAC}. Thus, the attacker has to first decrypt the MAC frames, then to reassemble the fragments of the original MLE message, and finally to decrypt the MLE message in order to get the value of the `Master Key TLV`.

Hence, the attacker can get the master key when she knows only the MLE key if the response MLE message is not fragmented and thus not encrypted at the MAC layer. We note that this mechanism is not affected by the `OBTAIN_MASTER_KEY` bit of the `Security Policy TLV`. When set, the `OBTAIN_MASTER_KEY` bit enables a Commissioner to extract the master key for out-of-band commissioning after she was authenticated.

This mechanism does not help a node to reconnect to a Thread network if the network master key was changed while it was sleeping because the node does not possess a valid MLE key to ask for the new master key. Thus its MLE requests will be dropped, and it has to be commissioned again by a human to the Thread network.

4.2 Processing of an `MLE Parent Request`

An obvious option for an attacker is to exploit the very first message exchange that allows a Child to connect to a Parent in a Thread network. Next we detail how a Router processes the first message sent by a Child that wishes to establish a communication link with a Parent.

Upon receipt of an `MLE Parent Request` message, the receiving Router extracts the received *Sequence* number from the `Key Source` field. Then, it compares the value of the received *Sequence* with its current internal *Sequence* number. There are two possible cases:

- If the two sequence numbers are equal, then the Router continues by processing the authentication tag of the received MLE message using the current MLE key K_{MLE} of the Thread network.

- If the two sequence numbers are not equal, then the Router derives a temporary key from the received *Sequence* number. The Router uses this temporary MLE key K'_{MLE} to process the authentication tag of the received message.

If the resulting tag is the same as the authentication tag present in the received `MLE Parent Request` message, then the Router prepares a response. Else, it will drop the received MLE message.

Whatever processing path the Router follows, it will perform at least an AES-CCM operation on the received message. Thus, an attacker can easily trigger executions of HMAC-SHA256 or AES-CCM by pretending to be a Child willing to connect to a Parent. If the attacker chooses to trigger HMAC-SHA256 executions on the receiving Router, she has to inject `MLE Parent Request` messages with a different *Sequence* number from the one observed in `MLE Advertisement` messages. On the other hand, if the attacker is interested in observing AES-CCM computations with the current network MLE key K_{MLE}, she has to inject `MLE Parent Request` messages with the same *Sequence* number. Hence, an attacker not yet connected to the target Thread network can take advantage of a normal network mechanism used to establish a communication link between a Child and a Parent Router to trigger executions of the underlying cryptographic algorithms with sensitive key material at her own will.

4.3 Attack on Key Generation

An attacker can inject `MLE Parent Request` messages with a chosen *Sequence* number. When receiving an `MLE Parent Request` with a different *Sequence* number from its own *Sequence* number, a Router will derive a temporary MLE key using Equation 2.

Although it is possible to trigger executions of the $HMAC$ function, the number of input bytes controlled by attacker is not enough to make the recovery of the master key MK possible as we show next.

The key derivation is pictorially shown in Figure 2. The one-way compression function of SHA-256 is denoted by F. The input message m to the HMAC function is obtained by concatenating the 4-byte *Sequence* number with the 6-byte representation of the "Thread" string, the 46 bytes of padding, and the 8-byte message length len: $m = \textit{Sequence} \parallel$ "*Thread*" $\parallel$ `0x80 0x00` ... `0x00` $\parallel len$. It is easy to observe that the only variable part of the message m is the *Sequence* number. Thus, the attacker controls exactly four bytes of the input message of the HMAC function.

If the attacker could recover $k_1 = F(IV, MK \oplus \textit{ipad})$ and $k_2 = F(IV, MK \oplus \textit{opad})$, then she could generate the MLE and MAC keys having only the correct *Sequence* number, but not the master key MK. Though, having the current MAC and MLE keys of the Thread network, the attacker can get the network master key from a Thread node as

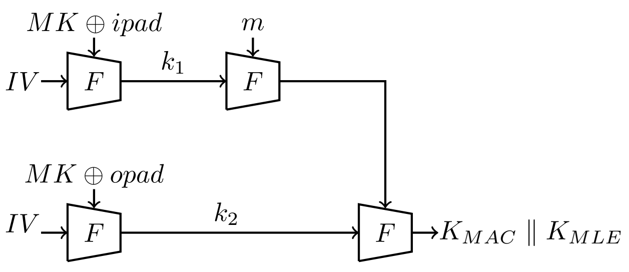


Figure 2: Key generation using HMAC.

described in Section 4.1. In order to recover k_1 and k_2, the attacker will target executions of the compression function $F(k_1, m)$. The attacker controls four bytes (a 32-bit word) of the input message, which are mixed in the first iteration of the compression function F with constant but unknown bytes of the internal state. As a consequence, she can learn the relationship between four unknown but constant 32-bit words by attacking a 32-bit modular addition in the first iteration. The attack stops here, because the attacker cannot propagate it to the next iterations in absence of known and variable data.

Thus, an attacker cannot exploit executions of the *HMAC* function in unprotected implementations using a CPA attack due to the limited control she has over the input. Though, the attacker still has the option to perform a template attack. Due to the complexity of the profiling phase required for mounting a template attack, we decided to stop our investigation at this point and to explore other attack paths instead.

4.4 Attack on the AES in CCM Mode

By injecting `MLE Parent Request` messages with the same *Sequence* number as the one used by the target Thread network, an attacker triggers executions of the AES in CCM mode with the current MLE key on the receiving Routers and REEDs.

A typical input block for the two stages (AES-CBC and AES-CTR) of the AES-CCM is shown in Figure 3. The constant (fixed) input bytes are given in hexadecimal notation, while the variable bytes are colored in gray. The first input block of the AES in CBC mode is very similar to the first input block of the AES in CTR mode. The first input byte is used for flags and thus is fixed. The last three input bytes are also fixed. The antepenultimate byte specifies the security level (`0x05`). As in the MAC layer of IEEE 802.15.4, this value indicates the use of encryption and authentication with a 4-byte message integrity code (MIC). In the case of AES-CBC the last two bytes are used to indicate the input plaintext length, while in the case of AES-CTR they represent the counter value. The counter value starts from one because the all-zero counter value is used for the computation of the authentication tag [Dwo07]. The variable bytes for both input blocks are the 8-byte source MAC address and the 4-byte frame counter.

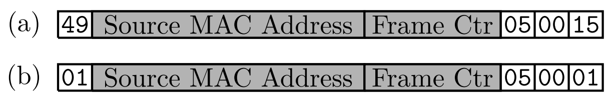


Figure 3: Input format for the first block of (a) AES-CBC and (b) AES-CTR.

An attacker can choose to craft `MLE Parent Request` messages having different payload lengths. As a consequence, the last byte of the AES-CBC is variable. This additional variable byte does not improve the attack outcome, but it is very likely to trigger an alert for abnormal network traffic in an intrusion detection/prevention system (IDS/IPS), if any.

To successfully mount a CPA attack, the attacker needs to vary a part of the input of the AES-CCM executions. As shown, an attacker can control up to 12 bytes of the input for AES executions. Thus, she can target either the first execution of the AES in CBC mode or the first execution of the AES in CTR mode.

Since the attacker does not control all input bytes, she has to attack three rounds of the AES in order to recover the 16-byte key. To propagate the CPA attack to the later rounds, the attacker must use the method described by Jaffe [Jaf07] to compute temporary keys that incorporate the constant input bytes. We found this MLE key recovery attack vector very promising and we chose to exploit it. Details of the full attack are given in Section 5.

5 Implementation of the Most Feasible Attack

In this section we describe each individual step of an attack path against a Thread network that chains the vulnerabilities presented above in the most feasible way. Then, we present the experimental setup we used to perform the attack. We show and analyze the results of the attack. Finally, we discuss the cost and complexity of the full attack.

The attack consists of the following four steps:

1. The attacker eavesdrops on the network traffic and records the *Sequence* number present in the `MLE Advertisement` messages sent by Thread Routers and REEDs.

2. The attacker observes and records the EM emanations of a target Router or REED while she injects `MLE Parent Request` messages. The injected messages use the observed *Sequence* number to trigger executions of AES with the current network MLE key. The variable and known inputs necessary to perform the CPA attack are the source MAC address and frame counter fields. They are randomly generated for each injected message. We stress that the side-channel attack is applicable only to Thread Routers and REEDs because they process the `MLE Parent Request` messages. This step requires the attacker to be in the proximity of the target device such that she is able to reliably measure the EM emissions. After the attacker has recorded enough traces, she continues with the next step of the attack.

3. The attacker correlates the observed EM leakages with a hypothetical model of a key-dependent sensitive intermediate variable in order to determine the unknown key. In practice, CPA is an efficient technique and thus it is employed to recover the MLE key used during the observed computations.

4. Having the current MLE key of the network, the attacker attaches to a Thread Router. She asks the Router for the network configuration parameters including the master key by sending an `MLE Child ID Request` message. The Router will give the attacker the requested information in an `MLE Child ID Response` message.

It is essential for the success of the attack that all the above-mentioned steps succeed. Failure of any of these steps will render the full attack infeasible. The success of the last step highly depends on how the Parent handles the `MLE Child ID Request` messages and on the length of the `MLE Child ID Response` message as discussed in Section 4.1. If the message is fragmented and therefore additionally encrypted with the MAC key, the attacker will need to recover K_{MAC}. For example, the MAC key can be recovered by mounting a CPA attack on the AES-CCM executions that use K_{MAC}.

O'Flynn and Chen [OC16] showed how to attack the MAC layer encryption of IEEE 802.15.4 nodes. Since Thread uses the MAC layer defined in the IEEE 802.15.4 standard, the attack strategy described by O'Flynn and Chen can be easily applied to mount an attack against the MAC layer encryption of Thread nodes. For clarity, we briefly describe how the attack works in the context of Thread nodes. The attacker can inject custom-crafted MAC frames that meet the following requirements. The 8-byte source MAC address must be set to the MAC address of a neighbor of the attacked node. The first byte of the `Auxiliary Security Header` must be set to `0x0D` to trigger encryptions with the current MAC layer key K_{MAC}. As in [OC16], the attacker can control the 4-byte value of the `Frame Counter` field. The high rate of MAC frames compared to MLE messages in a Thread network creates a good opportunity for a passive attack. Hence, the attacker can simply observe the processing of valid MAC frames instead of injecting custom-crafted ones. Since this attack was already studied in the literature, we did not implement it. We refer the reader to the work of O'Flynn and Chen [OC16] and the source code of OpenThread [2] for more details.

An important factor that can be controlled to a certain extent by the attacker is the quality of the side-channel acquisition. The better the signal-to-noise ratio of the recorded traces, the fewer traces the attacker will need. The number of traces required to recover the MLE key influences the duration of the acquisition. Thus, the risk of the attacker being noticed increases as she needs more EM traces. The reason is twofold. Firstly, the attacker has to spend more time in the vicinity of the target Router or REED. Secondly, she has to inject more messages to trigger executions of the AES.

Once the attacker has the network configuration parameters, including the security material, she has all the rights of a genuine member of the Thread network. Hence, she is able to communicate with other nodes and can understand the communication between other nodes. The attacker can commission new devices to the Thread network. She can become a Router and then change the network parameters so that the owner of the Thread network loses the control over his network.

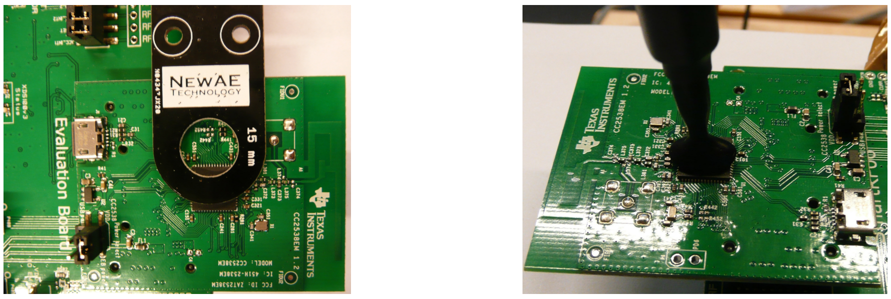
5.1 Experimental Setup

Thread Network. We created our own Thread network consisting of two CC2538EM wireless microcontrollers [Tex15]. These devices were an obvious choice since they were the first to support the OpenThread implementation [Ope16]. Moreover, our experimental network uses one of the seven products (device and software stack bundle) certified by the Thread Group [Thr17]. The CC2538EM microcontroller has an ARM Cortex-M3 processor clocked at 32 MHz, up to 512 KB of Flash memory and an IEEE 802.15.4 radio transceiver. The purchasing cost of this hardware was much smaller than that of similar devices shipped with proprietary implementations of Thread.

Initially, we experimented using the source code of OpenThread to understand the communication and the network mechanisms. Then, we modified the source code such that we were able to generate various sequences of messages. Following this approach we were able to better understand the source code and we inferred part of the Thread specifications. We emphasize that OpenThread's codebase is very complex. Thus, understanding the relevant network mechanisms to our analysis was a challenging task that required a tremendous effort. Finally, we modified the state machine of the OpenThread implementation to make an attacker able to inject custom-crafted messages.

Measurement Setup. For the acquisition of the EM emissions from the target device, we used a Teledyne LeCroy WaveRunner 625Zi [Tel16] oscilloscope. Initially, we ran AES encryptions in a loop on the target device and we used an EM probe to locate the spots that leak information during these executions. To hasten the process we firstly checked the area around the chip and then several decoupling capacitors we could identify using the schematic and layout of the board [Tex15].

[2]Function `ProcessReceiveSecurity` implemented in https://github.com/openthread/openthread/blob/master/src/core/mac/mac.cpp



(a) H-field probe.

(b) Near field probe.

Figure 4: The EM probes used for measurement of the EM leakage.

We started to capture the EM signal with a relatively inexpensive NewAE H-field probe having a 15 mm coil as shown in Figure 4a. Because we could not find a spot where we could visually identify a meaningful pattern for alignment, we switched to a more precise (coil sizes of about 1 mm) and relatively more expensive set of near field probes from Langer (see Figure 4b). We set the probe a few millimeters above the target board. For the results reported in this paper we fixed the sampling rate to 1 GS/s. A lower sampling rate (500 MS/s) significantly affected the attack outcome, while a higher sampling rate (5 GS/s) did not significantly improve the results.

The attacker's board running the custom implementation of OpenThread generated a trigger signal on an output port each time an injected message was sent. This signal was used by the oscilloscope to record the EM emissions. The injected messages and the corresponding EM traces acquired by the oscilloscope were saved on a personal computer. The target board ran a genuine implementation of OpenThread and acted as a Router in our Thread network. We stress that no dedicated trigger signal was provided from the target board. We chose to power the target board from a regulated power supply rather than from a PC USB port to reduce the noise. These settings accurately replicate a real usage scenario of a Thread device.

Admittedly, the network traffic may not be similar to the one of a real Thread network with very active data transfers. In our experiments we were able to inject our crafted MLE Parent Requests at a rate of up to 25 messages per second. On a busy network with a significantly higher legitimate packet rate there may be mismatches between captured traces and packets. The number of traces for the attack may then grow. Identifying relevant traces by pattern may address this problem.

5.2 Alignment of the EM Traces

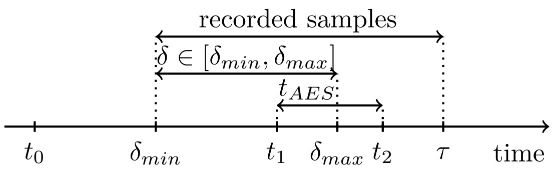


Figure 5: Timing of various events that occur during the acquisition of an EM trace.

Once the side-channel acquisition is over, the attacker has to align the EM traces. The timing of different events that occur during the acquisition of a trace are shown in Figure 5. The injected message is sent at t_0 and the oscilloscope is triggered. The oscilloscope records the sampled signal between δ_{min} and $\tau = \delta_{max} + t_{AES}$, while the relevant part for the attacker t_{AES} is between t_1 and t_2.

The attacker has to determine experimentally the interval in which the relevant part of the AES execution is very likely to start at $\delta \in [\delta_{min}, \delta_{max}]$. Besides, she can record a sufficiently long interval that includes the relevant samples for most of the measurements. For our experiments, the value of δ was in the interval $[555, 564]$ μs, while the relevant part of the AES execution t_{AES} took 14.656 μs.

As illustrated by Figure 5, the relevant AES execution occurs at different locations in the recorded traces. For the CPA attack to work, the attacker has to extract and align the relevant samples from the recorded traces. To this end, the attacker builds a pattern consisting of interesting samples by precisely identifying the relevant computations within a trace. We were able to visually identify the first round of the AES in the first trace and thus we used this part as the alignment pattern.

We explored two methods for alignment of the EM traces: Sum of Absolute Differences (SAD) and Cross-Correlation (CC). For the sake of accuracy we quantified the precision of the alignment for each of the two methods. We raised a signal when the relevant part of a trace starts and we compared the corresponding sample number to the determined sample number by the SAD and CC methods. The comparison showed that a difference between the two values of more than three samples occurs for less than 1% of the traces. Therefore, both methods are very efficient. We chose to use the SAD method because it was a little bit faster while discarding slightly less traces than the CC method.

5.3 Attack Results

The most difficult step of the attack was the side-channel acquisition. Indeed, we spent about half of the total time devoted to mount the full attack on improving the side-channel attack outcome. This was an iterative process, which required a good understanding of the EM leakage of the target and fine adjustments of the attack parameters. The type and position of the probe are crucial for the quality of the side-channel acquisition.

The number and quality of the EM traces required to recover the MLE key determine the cost of the full attack. Our experimental results showed that 10,000 EM traces are sufficient. We note that two key bytes were much more difficult to recover than the rest. We tried different side-channel techniques, including linear regression attacks [LPR13], but we did not see an improvement. This behaviour is determined by the hardware characteristics of the target device, the clock frequency, and the sequence of instructions executed by the target device. It does not depend on the value of the attacked key byte, but on its location. Similar results were reported in the side-channel literature [LMPT15].

The acquisition of 10,000 EM traces took about three hours. Given the cost of the active attack, a passive attack scenario is rendered almost impossible. It is very likely that the temporary keys are changed before the attacker has observed enough executions for different input messages.

The last step of the attack appeared to be impossible in a recent version of the OpenThread implementation we experimented with (commit `11c1b49`), because the fragmentation of the `MLE Child ID Response` messages (and therefore their additional encryption with K_{MAC}) cannot be avoided. We do not exclude the possibility to get non-fragmented answers from other stacks depending on the Parent implementation. Alternatively, an additional DEMA to recover the K_{MAC} can be mounted as described above.

5.4 Improving the Attack

Although the attacker can control up to 12 bytes of the input as shown in Section 4.4, she might want to fix the two input bytes corresponding to the key bytes that are difficult to recover. This requires some understanding of the target's leakage, but it is by far easier to perform and much more reliable than a template attack. For example, the attacker can use a similar device running OpenThread to learn which key bytes are more difficult to attack and then adjust the variable input bytes accordingly.

To further optimize the attack, one can search an input configuration that minimizes the number of attacked bytes while considering the input constraints. The number of AES rounds that have to be attacked in order to recover the full key must be kept to the minimum value of three, since attacking more rounds requires more EM samples and thus increases the cost of the measurement equipment as well as the duration of the offline attack.

In our case, there are 45 out of the $2^{10} - 1$ possible input configurations that minimize the number of attacked bytes. Compared to the straightforward attack, the improved attack reduces the number of individual CPA attacks from 44 to 37, which results in a 16% improvement.

The novelty of the improved attack stems from the fact that the attacker can adjust her attack strategy depending on the leakage of the target by fixing some input bytes she can control. To the best of our knowledge, we are the first to use this attack techniques to improve the outcome of a CPA attack. The total number of traces required to recover the full key can be decreased by about 50%, to no more than 5,000 EM traces. Another advantage of this approach is that the attack may pass undetected when it uses fewer variable input bytes because the injected packets resemble normal network traffic. The technique used to improve this attack is detailed in [BDC17].

6 Feasibility and Limitations

The main question arising is perhaps the relative complexity of the attack and the realism of the setting where the attacker needs to be in a very close proximity of a Thread Router or REED with a digital oscilloscope. This is a reasonable question, especially in the setting where Thread brings IPv6 to the end nodes and opens up the remote attack surface. We are realistic to state that the attacks we outlined are currently beyond the reach of an average hacker familiar just with software and networking techniques (and absolutely not for "script kiddies"), and apply only in particular settings. Appendix B shows a formalized quantification of the attack effort common to the smart card world.

Equipment Cost. The need of specialized equipment (i.e. a digital oscilloscope, an EM probe, and if needed an amplifier) hinders fast widespread application of the attack. In our experiments, we have used a high-end digital oscilloscope because we just had one available. Our experiments suggest that perhaps the cheapest available side-channel analysis hardware, ChipWhisperer [O'F17], would not be sufficient to succeed on most of the targets due to its relatively low sampling rate[3]. However, low- to mid-range digital oscilloscopes such as the Picoscope [Pic17] should be sufficient. Combined with the increased availability of tooling to perform the analysis part of the side-channel attack, starting from the software tooling and tutorials of [O'F17] to higher-performance toolkits like [Dar17] and [Jls17], this makes us claim that the attack is well feasible. With side-channel techniques and expertise becoming more mainstream in the hacker community, the threat of such attacks increases.

Portability. Our attack is moderate in portability. Namely, on another target family (a different hardware or software implementation) the attacker would most likely need to tune the side-channel attack to that target in terms of probe position, alignment parameters, etc. Hence, she has to invest into the identification phase of the attack. Due

[3]O'Flynn and Chen [OC15] demonstrate synchronous sampling with clock recovery feature of Chip-Whispeper in the power analysis setting; this feature may make ChipWhisperer applicable in the setting we describe.

to the physical nature of side-channel attacks, our complexity estimate is based on one particular implementation we analyzed. For other implementations the complexity may be lower or higher; the attack may require a less or more expensive oscilloscope; however, for an unprotected cryptographic implementation we expect the same order of magnitude in terms of the amount of traces (and therefore time).

Other Attacks. Though we considered in our analysis a side-channel capable adversary, we were not excluding attack paths that do not require the use of side-channel techniques and therefore specialized equipment. However, we did not discover any paths that do not require specialized equipment. They may still exist, though. We did not consider other implementation attacks such as fault injection attacks or timing attacks. They may be applicable and there may be settings where they are a realistic threat, especially timing attacks that can be performed remotely without specialized equipment [ASK07].

The advantage of our attack is that it would circumvent IP protocol protections such as firewalls, akin to the recent ZigBee worm [RSWO17], thus may serve as a more feasible entry-point to the system. A limitation of our attack is that it does not address the application layer security mechanisms that would normally be deployed on top of the Thread networking stack. However, such mechanisms are not addressed by Thread.

7 Countermeasures

Although it is desirable to achieve a defense in depth for a Thread network by employing redundant security mechanisms, other factors such as manufacturing costs or usability pose major constraints. Thus, a trade-off between these contradicting requirements should be sought to ensure an appropriate level of security. Though inspired by our case study of Thread, the countermeasures laid out next are applicable to other IoT protocols and devices as well. They are valuable for both protocol designers and engineers of IoT products.

Tamper Resistance. We suggest the use of shielded and tamper resistant components and cases. A trade-off between cost and product dimensions would be to insert small air gaps between the circuitry and the external case. This would most likely require device disassembly to enable EM analysis, making the attack more cumbersome to perform.

Protected Cryptographic Implementations. We stress that in scenarios where side-channel attacks pose a threat, Thread implementations should employ side-channel protection mechanisms for the manipulation of the security material. Consequently, the loading of the security material as well as all cryptographic algorithm implementations should use countermeasures, such as masking and hiding [MOP07]. If the cost of the countermeasures is prohibitive, offloading the cryptographic algorithms to hardware cryptographic engines might be a good trade-off. In general, it is harder (but still feasible) to attack a hardware implementation than a software implementation, as demonstrated in [LMPT15]. Protected hardware implementations should be considered where both high security and high performance are necessary.

Thread-Specific Protocol Level Mitigations. We suggest to consider ways to mitigate the presented attack paths at the Thread protocol level, changing the network mechanisms that facilitate the attacks. Most importantly, we recommend to never transfer the network master key MK encrypted with temporary keys K_{MLE} and/or K_{MAC}. Instead, the transfer of MK should happen encrypted with a one-time ephemeral key. Note that this is the way MK is transferred during commissioning, where a dedicated Key Encryption/Establishment Key (KEK) is used, see Thread Commissioning White Paper at [Thr16a].

If this is not possible, disabling or limiting the message exchange that allows a node to get the network master key by sending an `MLE Child ID Request` message to its Parent should be considered. Rate limiting the incoming `MLE Parent Request` messages processed by a Parent (Router or REED) significantly slows the attacker, but care should be taken not to expose the network to DoS attacks. We also recommend reducing the K_{MLE} and K_{MAC} default rotation time from 672 hours to 1 hour.

Generic Protocol Level Mitigations. When the cost of protecting the cryptographic implementations of a block cipher is too high, fresh re-keying schemes can be used to prevent side-channel attacks [MSGR10, MPR$^+$11, DEM$^+$16]. These schemes make use of a re-keying function to generate new session keys based on the secret master key and random nonces for every block of message to be encrypted. Although fresh re-keying has the benefit that the re-keying function can be protected against side-channel attacks at a much lower cost than the block cipher [DEMM14], it involves significant changes in the protocol.

Security Certification Scheme. We recommend enforcing a security certification scheme for Thread products in addition to the functional certification scheme currently in place. Although a certification scheme cannot prevent new attacks, the benefits for the security of the whole ecosystems are obvious. A security certification seal increases consumer awareness of possible security issues and attests resistance to known attacks. The major drawbacks are an increase of the overall price and an additional delay before the certified products are available on the market.

8 Conclusion

We conducted the first electromagnetic side-channel vulnerability analysis of the Thread networking stack. We described how different network mechanisms that can be learned by the attacker from the published OpenThread code can be used to create attack vectors for side-channel attacks. We showed that some of the side-channel attack paths are hard or impossible to exploit in the context of Thread. We implemented the most promising attack path that provides complete access to the Thread network. It exploits a chain of network mechanism and side-channel attacks on executions of unprotected implementations of cryptographic algorithms.

The full attack did not succeed in our experimental network of OpenThread nodes. We consider the setting where the last attack step is indirectly prevented by the `MLE Child ID Response` payload size to be insufficient to rely upon. Firstly, it is not in the design of the protocol that the master key is protected by both K_{MLE} and K_{MAC}. Additional protection by K_{MAC} is a side effect. Secondly, a possibility to request the master key having the derived key(s) is questionable security-wise as it subverts the essence of key derivation using HMAC. Hence, we suppose that the full attack may succeed with moderate effort for other implementations of the stack, or by mounting an additional DEMA to recover K_{MAC} as described in [OC16].

The possibility of an arbitrary Thread device to trigger cryptographic operations and responses from a commissioned Thread device at unlimited rate presents a standalone risk of a denial-of-service attack.

The lessons learned from our work can be applied to other IoT systems and protocols as well. Our threat model can be used to better shape the attack surface of future IoT products and prevent issues such as: processing of invalid injected messages, EM leakage, converting temporary keys into master key, and using a single network master key to secure the whole network. In light of our results, designers of future protocols for the IoT should carefully consider the threat of side-channel attacks from the early inception. We recommend to apply tamper resistance for low security level, use hardware or protected cryptographic implementations for moderate security level, and implement protocol mitigations to achieve high security level.

In general, we demonstrated that in the context of a modern IoT network protocol mounting a side channel attack is not trivial. Similar to a modern software exploit, it requires chaining multiple vulnerabilities. Nevertheless, such attacks are feasible. Being perhaps too expensive for settings like smart homes, they pose a relatively higher threat to commercial settings.

Acknowledgments

We thank the anonymous reviewers and Jacques Fournier for their helpful comments, and our contacts in Thread Group for their feedback.

References

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002*, pages 29–45, 2002.

[AM14] Ahmad Atamli and Andrew P. Martin. Threat-based security analysis for the internet of things. In Gabriel Ghinita, Razvan Rughinis, and Ahmad-Reza Sadeghi, editors, *2014 International Workshop on Secure Internet of Things, SIoT 2014, Wroclaw, Poland, September 10, 2014*, pages 35–43. IEEE Computer Society, 2014.

[ARM17] ARM MBED. mbed TLS. https://tls.mbed.org/, 2017.

[ASK07] Onur Aciiçmez, Werner Schindler, and Çetin Kaya Koç. Cache based remote timing attack on the AES. In *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007*, pages 271–286, 2007.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004*, pages 16–29, 2004.

[BDC17] Alex Biryukov, Daniel Dinu, and Yann Le Corre. Side-channel attacks meet secure network protocols. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, volume 10355 of *Lecture Notes in Computer Science*, pages 435–454. Springer, 2017.

[CK14] Omar Choudary and Markus G. Kuhn. Template attacks on different devices. In *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014*, pages 179–198, 2014.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002*, pages 13–28, 2002.

[Dar17] DareDevil: A tool to perform (higher-order) correlation power analysis attacks (CPA). https://github.com/SideChannelMarvels/Daredevil, 2017.

[DEM$^+$16] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. Cryptology ePrint Archive, Report 2016/952, 2016. http://eprint.iacr.org/2016/952.

[DEMM14] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, and Florian Mendel. On the security of fresh re-keying to counteract side-channel and fault attacks. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2014.

[dMS10] Giacomo de Meulenaer and François-Xavier Standaert. Stealthy compromise of wireless sensor nodes with power analysis attacks. In Periklis Chatzimisios, Christos V. Verikoukis, Ignacio Santamaría, Massimiliano Laddomada, and Oliver Hoffmann, editors, *Mobile Lightweight Wireless Systems - Second International ICST Conference, MOBILIGHT 2010, Barcelona, Spain, May 10-12, 2010, Revised Selected Papers*, volume 45 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 229–242. Springer, 2010.

[DPN$^+$16] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. Dismantling real-world ECC with horizontal and vertical template attacks. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016*, pages 88–108, 2016.

[Dwo07] Morris J Dworkin. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. NIST Special Publication 800-38C, 2007.

[FJP16] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 636–654, 2016.

[FPR$^+$16] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. Flowfence: Practical data protection for emerging IoT application frameworks. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 531–548, 2016.

[IEE17] IEEE. IEEE Standard for Low-Rate Wireless Networks. https://standards.ieee.org/, 2017.

[Jaf07] Joshua Jaffe. A first-order DPA attack against AES in counter mode with unknown initial counter. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007*, pages 1–13, 2007.

[Jls17] Jlsca: Side-channel toolkit in Julia. https://github.com/Riscure/Jlsca, 2017.

[Kiz09] Ilya Kizhvatov. Side channel analysis of AVR XMEGA crypto engine. In Dimitrios N. Serpanos and Wayne H. Wolf, editors, *Proceedings of the 4th Workshop on Embedded Systems Security, WESS 2009, Grenoble, France, October 15, 2009*. ACM, 2009.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999*, pages 388–397, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996*, pages 104–113, 1996.

[LL16] Citizen Lab and Lookout. Sophisticated, persistent mobile attack against high-value targets on iOS. https://blog.lookout.com/trident-pegasus, 2016.

[LMPT15] Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. Soc it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 620–640, 2015.

[LPR13] Victor Lomné, Emmanuel Prouff, and Thomas Roche. Behind the scene of side channel attacks. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 506–525, 2013.

[Mic17] Microsoft. Internet of Things security architecture. https://docs.microsoft.com/en-us/azure/iot-suite/iot-security-architecture, 2017.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007.

[MPR$^+$11] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2011.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[Nov17] Matt Novak. Hackers Shut Down The Key Card Machine In This Hotel Until a Bitcoin Ransom Was Paid [Corrected]. http://gizmodo.com/hackers-locked-every-room-in-this-hotel-until-a-bitcoin-1791769502, 2017.

[OC15] Colin O'Flynn and Zhizhang Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *J. Cryptographic Engineering*, 5(1):53–69, 2015.

[OC16] Colin O'Flynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15.4 nodes. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016*, pages 55–70, 2016.

[O'F17] Colin O'Flynn. Chipwhisperer. https://newae.com/tools/chipwhisperer/, 2017.

[Ope16] OpenThread. https://github.com/openthread/openthread, 2016.

[Pic17] Pico Technology. https://www.picotech.com/products/oscilloscope, 2017.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[RRKS14] Michael Rushanan, Aviel D. Rubin, Denis Foo Kune, and Colleen M. Swanson. Sok: Security and privacy in implantable medical devices and body area networks. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 524–539, 2014.

[RSWO17] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 195–212. IEEE Computer Society, 2017.

[SDK$^+$13] Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Part I*, pages 147–164, 2013.

[SOG13] SOG-IS. Joint Interpretation Library – Application of Attack Potential to Smartcards, January 2013. http://www.sogis.org/uk/supporting_doc_en.html.

[Tel16] Teledyne LeCroy. WaveRunner 625Zi. http://teledynelecroy.com/, 2016.

[Tex15] Texas Instruments. CC2538. http://www.ti.com/lit/ds/symlink/cc2538.pdf, 2015.

[Thr16a] Thread Group. Resources. https://www.threadgroup.org/ourresources, 2016.

[Thr16b] Thread Group. Thread. https://www.threadgroup.org/, 2016.

[Thr16c] Thread Group. Thread Group Broadens Focus to Encompass the Places Where People Live and Work with Expansion Into Commercial Building Space, Nov 2016. http://threadgroup.org/news-events/press-releases/.

[Thr17] Thread Group. Thread Certified Products. http://threadgroup.org/technology/ourtechnology#certifiedproducts, 2017.

A Additional Attack Paths

Attack on Loading the Security Material. As we have shown in Section 2, template attacks are powerful side-channel attacks that can recover sensitive values using very few traces. Thus, an attacker can purchase a device similar to the one to be attacked to create EM profiles. Then, she can force the targeted Thread device to reset such that she can observe the EM emanations corresponding to the loading of the network parameters from non-volatile memory. In particular, the attacker is interested to capture the loading of the network master key MK and commissioning key CK. Though powerful, template attacks depend on the quality of the templates made in the profiling phase. Thus, we did not investigate this attack vector further.

Elliptic Curve Implementations. The execution of elliptic curve computations might be vulnerable to timing [Koc96] or Simple Power Analysis (SPA) [KJJ99] attacks if not properly implemented. The OpenThread implementation of the Thread networking stack relies on mbed TLS [ARM17] for cryptographic services. Recently, Dugardin *et al.* showed that the point blinding countermeasure must be activated in mbed TLS for elliptic curve implementations to prevent horizontal and vertical template attacks [DPN$^+$16]. We did not investigate into this direction further.

B Quantification of the Attack Effort

There is no standard procedure to quantify the attacker's potential to perform an attack on an IoT device. Thus, we use an adaptation of the rating for smart cards from the Joint Interpretation Library [SOG13] to rate our attack. The rating procedure interprets the Common Criteria methodology based on smart card evaluation experience gained by the industry. It is used in practice by testing laboratories to quantify the resistance of smart cards to various attacks, including protocol and side-channel attacks. The aforementioned procedure distinguishes two independent phases for an attack: identification and exploitation. The identification phase refers to the demonstration of the attack, while the exploitation phase considers the impact of the attack when all necessary tools are readily available from the identification phase.

Table 3: Attack rating using an adaptation of the rating for smart cards from Joint Interpretation Library [SOG13].

Factors	**Identification**		**Exploitation**	
	Rating	**Score**	**Rating**	**Score**
Elapsed time	more than 1 month	5	less than 1 day	3
Expertise	expert	5	proficient	2
Knowledge of TOE	public	0	public	0
Access to TOE	less than 10	0	less than 10	0
Equipment	specialized	3	specialized – standard	3
Open samples	public/not required	0	–	–
Sum		13		8
Total	**21 (enhanced-basic)**			

Next, we briefly introduce the factors considered by the rating methodology. The *elapsed time* defines the time required by the attacker to mount the attack from the moment she has access to the target. The *expertise* reflects the knowledge the attacker should have to mount the attack. The *knowledge of the target of evaluation (TOE)* indicates the level of access to the specifications. In the case of our attack, although the access to the official specifications is restricted, relevant information can be inferred from the open source implementation placed in the public domain. The number of different devices on which the attacker needs to perform the attack is captured by the *access to TOE* factor. The technical resources required for the attack are comprised in the *equipment* factor. Finally, the *open samples* factor measures to which extent the attacker is able to modify the software running on the target device. For more details, we refer the reader to [SOG13].

The individual scores for each factor are given in Table 3 for an implementation where the last step of the attack is feasible. The final score of 21 classifies our attack as an enhanced-basic attack. The rating places our attack in between basic attacks that are easy to perform and enhanced attacks that require an advanced effort.

C Details on Attack Target and EM Probe Performance

During our attack, the CC2538 target was running the default OpenThread configuration. This configuration uses the T-tables based software AES implementation of mbedTLS. The source code can be seen at https://github.com/ARMmbed/mbedtls/blob/development/library/aes.c. The CC2538 target has an ARM Cortex-M3 core and was clocked at 32 MHz.

For the acquisition, we used the best EM probe position we could identify to reveal a relevant pattern in the side channel traces. The relevant part (i.e. the first three rounds of AES) of an EM trace is presented in Figure 6; it was obtained with Langer RF-K 7-4 probe[4].

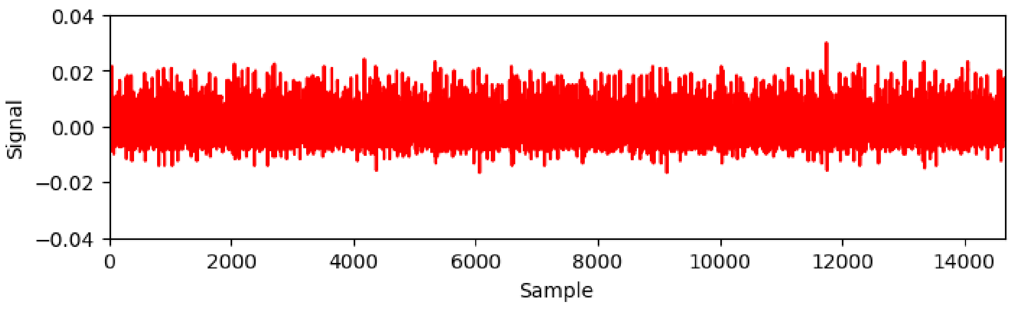


Figure 6: The relevant part (i.e. the first three rounds of AES) of an acquired EM trace.

It is difficult to provide an exact comparison with existing attacks on similar devices, as many factors impact the attack complexity, including the mode of operation of the cipher and the use of peripherals, most notably the radio (which was used in our setting). However we believe our attack matches published results in terms of the ballpark figures. Namely, experiments on the TI AM335x SoC clocked at 1 GHz reported by Longo *et al.* at CHES 2015 [LMPT15]. They show that a DEMA attack on a software implementation of the AES (in CBC mode) using T-tables requires 3,000 to 10,000 traces. Another reference point is provided by Biryukov *et al.* [BDC17], where a DEMA attack on a Cortex-M3 clocked at 8 MHz and running a software implementation of the AES based on T-tables (in ECB mode with limited input control) requires less than 1,600 traces. In both references, no on-chip radio was used.

We exemplarily show the evolution of the guessing entropy for the second key byte in Figure 7. For this key byte 2,000 EM traces are enough to recover the correct key. In Figure 8 we give the correlation of all candidates for the same key byte when using 3,000 traces; the correlation of the correct key is shown in red.

While we have spent some effort on the optimization of the measurement setup, the number of traces in the side-channel part of our attack may lend itself to further reduction. We did not aim to over-optimize the side-channel setup as we believe the side-channel aspect was already significantly studied in the literature, in particular in the above-mentioned works. Instead, the focus of this work is to show the possibility of the full-stack attack.

There is a possibility to use the hardware AES accelerator of CC2538 by configuring mbedTLS as described in https://openthread.io/guides/porting/implement_

[4]RF-K 7-4 H-Field Probe: 30 MHz up to 1 GHz, https://www.langer-emv.de/en/product/rf-passive-30-mhz-3-ghz/35/rf-k-7-4-h-field-probe-30-mhz-up-to-1-ghz/9

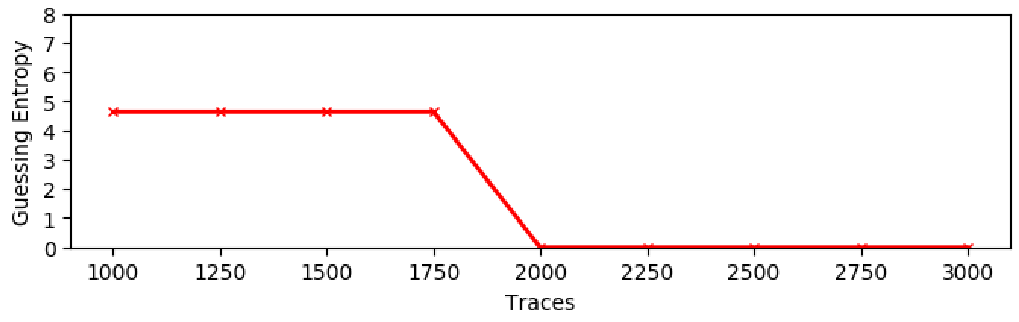


Figure 7: Evolution of the guessing entropy of the second key byte for different number of traces.

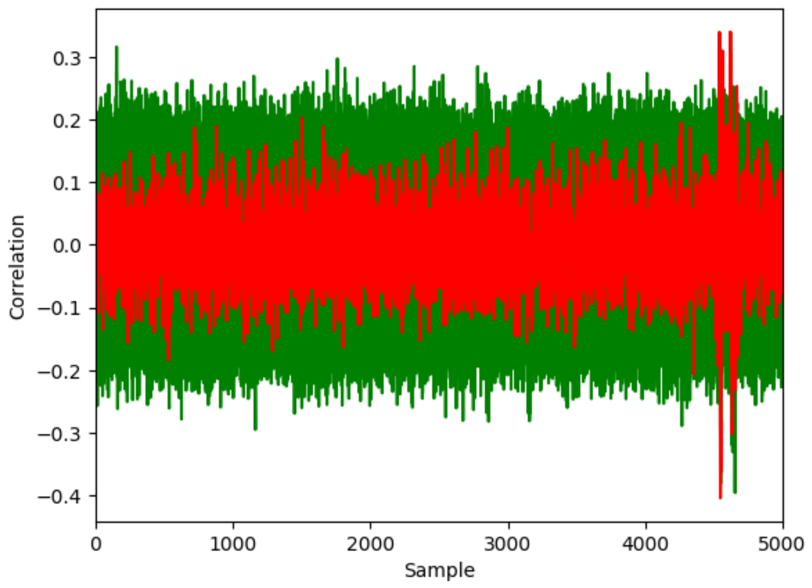


Figure 8: Correlation of all candidates for the second key byte. The correlation of the correct key is shown in red.

`advanced_features`. Other platforms (and other Thread stacks) may also use hardware accelerators. Based on the published experiments, we expect that switching to the hardware AES accelerator will significantly increase the attack complexity. In particular, finding the relevant pattern for aligning the traces may become challenging. In [LMPT15], it took 500,000 EM traces and a meticulous alignment procedure to successfully attack a HW AES accelerator. This is why we recommend the use of HW crypto implementations as a moderate-level countermeasure for the connected device setting.