Bachelor thesis
Computing Science

Radboud University

Multi-user security analysis of three finalists of the NIST ligthweight cryptography competition

Author:
Bart Veldman
s1017975

First supervisor/assessor:
Dr. ir. Bart Mennink
b.mennink@cs.ru.nl

Second supervisor/assessor:
Prof. dr. Joan Daemen
joan@cs.ru.nl

January 12, 2023

Abstract

This thesis investigates the multi-user security of three cryptographic schemes with a focus on key-recovery attacks. The cryptographic schemes are some of the permutation-based finalists of the lightweight cryptography competition organized by the National Institute of Standards and Technology. The attacks are multi-user attacks and allow assessment of the multi-user security of the three cryptographic schemes. The security model used for the attacks comprises of a key-recovery adversary with the ability to choose the nonce and plaintext. No surprising security bound was found nor expected. The multi-user security of the three cryptographic schemes depend on their block length or rate, and on the number of attacked devices and offline generated ciphertexts.

Contents

Chapter 1

Introduction

Internet-of-Things (IoT) devices are being rapidly adopted by the public. They come in many forms; household applications, wearables, and promise to improve our everyday lives. As the name suggests, IoT devices are characterised by communicating, often wirelessly, via internet. However, while they can provide benefits, they also introduce new challenges [4]. IoT devices provide a new attack vector for cyber criminals, due to their connection to the internet and generally poor security. This has lead to an increase in frequency and severity of cyber-attacks. Their security can be broken, allowing an attacker to control the device, or use it to amplify other cyber attacks.

One of the security issues concerns cryptography. A cryptographic scheme describes the order of operations in which a cryptographic functionality is carried out. Possible cryptographic functionalities are encryption, decryption and authentication, often using a private key which is kept secret. Most IoT applications run on resource-constraint devices. Hence, conventional cryptographic schemes can be too computationally heavy to be feasible. This has lead to a demand for alternative schemes which are less computationally expensive, but still provide sufficient security. Luckily there are plenty of such lightweight schemes, but their security is not always clear.

Historically, the U.S. National Institute of Standards and Technology (NIST) has issued standardization processes in which cryptographic schemes are standardized based on their functionality. Previous standardization processes concerned for example block ciphers [6], hash functions [7], and key establishment schemes [8], and more recently NIST has announced a lightweight cryptography standardization process [5]. These processes aim to find a cryptographic scheme which offers the best security and satisfies certain criteria related to the type of scheme. The winner of the standardization process then becomes the "standard"; the recommended cryptographic scheme to implement. With regards to lightweight cryptography and IoT, develop-

ers will have a clear recommendation and a set of guarantees to work with, enabling them to combat possible security issues.

The standardization process works in multiple rounds. Each round the submissions to the NIST standardization process are heavily scrutinized in terms of security and performance. Subsequently each round multiple submissions will be rejected based on findings by the cryptography community. As of right now, the process is in its final round: on March 29, 2021, the ten finalists have been announced. Since the amount of submissions have been reduced significantly, more attention can be put at identifying vulnerabilities in the ten finalists. One area that deserves more attention is vulnerability to multi-user attacks. A cryptographic scheme is vulnerable to multi-user attacks if an attacker has a greater chance of recovering the key by attacking multiple devices, rather than attacking a single device multiple times.

In this thesis, I review three of the ten finalists of the NIST lightweight cryptography standardization process. The aim of these reviews is to investigate their multi-user security by describing a multi-user key-recovery attack for each finalist, enabling further insight into their security. The three finalists are Elephant [1], ASCON [3] and ISAP [2].

Chapter 2, preliminaries, contains the necessary information to be able to comprehend the contents of this thesis. It will allow anyone to grasp the importance and considerations regarding multi-user security of cryptographic schemes. Chapter 3 explores the three finalists. It contains a specification and a multi-user attack for each scheme. The thesis is concluded in Chapter 4.

Chapter 2

Preliminaries

This chapter provides information about the notation (Section 2.1), primitive (Section 2.2), cryptographic functionality (Section 2.3), and security model (Section 2.4) that will be used in the research chapter.

2.1 Notation

Symbols	
$\perp$	Error symbol, indicating a failed authentication.
$\mathbb{N}$	Set of all natural numbers.
$\mathbb{Z}$	Set of all integers.
$\mathbb{Z}^+$	Set of all positive integers.
$\{0,1\}^x$	Bitstring of length x.
$\mathbf{Pr}(X)$	Probability of event X.
Operations	
$\lvert x \rvert$	Length of bitstring x in bits.
$x \mid\mid y$	Concatenation of bitstrings x and y.
$x \oplus y$	Xor of bitstrings x and y.
$X \circ Y$	Function composition of X and Y.
$A : x \rightarrow y$	Definition of function A, where x is input and y is output.

2.2 Permutation

Cryptographic schemes often consist of building blocks to create larger schemes. One such building block is a cryptographic primitive, a low-level function. The finalists of the NIST competition that are researched in the next chapter use the same primitive: a permutation. A permutation is a bijection of bits, and since it is a bijective function it is invertible. Let `p` be

a permutation and $b \in \mathbb{Z}^+$, then:

$$\mathtt{p} : \{0,1\}^b \to \{0,1\}^b.$$

The permutation takes as input a bit string of length b, and outputs a bit string of length b.

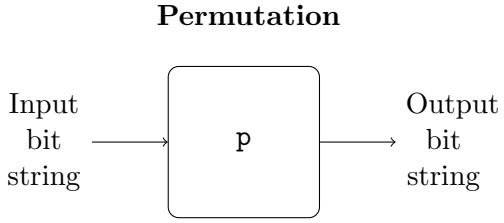

Figure 2.1: High-level overview of a permutation

2.3 Authenticated Encryption with Associated Data

The NIST Lightweight Cryptography competition aims to set a standard for authenticated encryption (AE) [9]. AE is a functionality that offers confidentiality and authenticity. Encryption ensures that no third party can see the contents of the data, ensuring confidentiality, while the tag provides authentication; if the tag is valid, the data is the exact data sent by the author.

AE consists of two algorithms: encryption and decryption, depicted in Figure 2.2. The encryption algorithm takes a key, a nonce, the associated data and the plaintext as input, and outputs a ciphertext and a tag. The decryption algorithm also takes a key, a nonce and the associated data, as well as a ciphertext and corresponding tag. If the tag proves to be a valid tag it will output the decrypted ciphertext: the plaintext. If the tag proves to be invalid it will output an error instead.

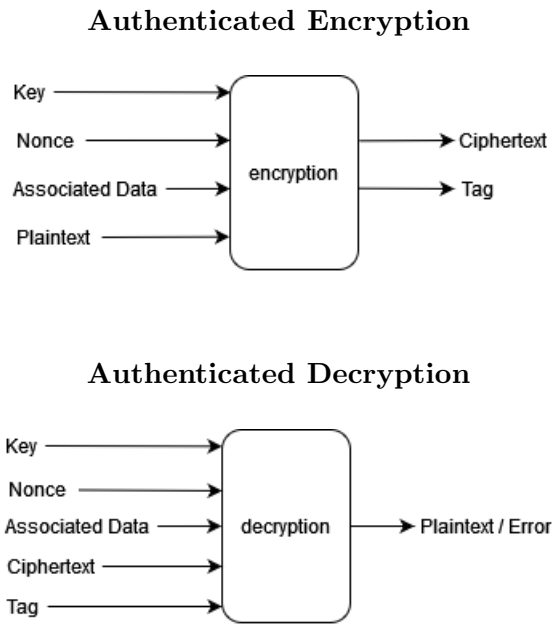

Figure 2.2: High-level overview of AE

2.4 Security model

Key-recovery adversary

The adversary $\mathcal{A}$ engages in a key-recovery game. A set of μ keys $\mathbf{K} = \{K_1, \ldots, K_\mu\}$ is randomly generated and $\mathcal{A}$ wins the game if it correctly recovers one of these keys. The adversary has access to online oracles E_{K_1}, $\ldots$, E_{K_μ} and offline oracles $\mathtt{p}$, $\mathtt{p}^{-1}$, and can query q construction queries to the online oracles and p primitive queries to the offline oracles. Construction queries return a ciphertext C, whereas primitive queries return the result of a permutation. $\mathcal{A}$ will use these queries to recover a key and submit (K, i). If $K_i = K$ holds, $\mathcal{A}$ has found a key and wins the game:

$$\mathbf{Adv}^{\mathrm{keyrec}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\{E_{K_1},\ldots,E_{K_\mu},\mathtt{p},\mathtt{p}^{-1}\}} = (K, i) \,:\, K_i = K)$$

Chapter 3

Research

This chapter deals with the attacks on three of the NIST finalists. It is split into three sections, one for each of the finalists. Each section contains a specification where the relevant parts of the cryptographic scheme are specified. Following the specification is a brief explanation of the security model used when creating the attack. The attack itself concludes each section.

3.1 Elephant

3.1.1 Specification

A full specification of Elephant v2 is available at [1]. A part of the specification, as presented in [1], is shown here.

Let $k, m, n, t \in \mathbb{N}$ with $k, m, t \leq n$. k, m, t, and n are the lengths of the key, nonce, tag, and block, respectively. Let $\mathtt{p} : \{0,1\}^n \to \{0,1\}^n$ be an n-bit permutation, and $\varphi_1 : \{0,1\}^n \to \{0,1\}^n$ be an LSFR. Define $\varphi_2 = \varphi_1 \oplus \mathtt{id}$, where $\mathtt{id}$ is the identity function. Define the function $\mathtt{mask} : \{0,1\}^k \times \mathbb{N}^2 \to \{0,1\}^n$ as follows:

$$\mathtt{mask}_K^{a,b} = \mathtt{mask}(K, a, b) = \varphi_2^b \circ \varphi_1^a \circ \mathtt{p}(K \mid\mid 0^{n-k}).$$

The encryption algorithm gets as input a key $K \in \{0,1\}^k$, nonce $N \in \{0,1\}^m$, associated data $A \in \{0,1\}^*$, and a plaintext $P \in \{0,1\}^*$, and it outputs ciphertext $C \in \{0,1\}^{|P|}$, and a tag $T \in \{0,1\}^t$. Encryption is illustrated in Figure 3.1, for the specific case where $|P| = n$.

The decryption algorithm gets as input a key $K \in \{0,1\}^k$, and a nonce $N \in \{0,1\}^m$, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^t$, and it outputs a plaintext $P \in \{0,1\}^{|C|}$ if the tag is correct, or a dedicated $\perp$-sign otherwise.

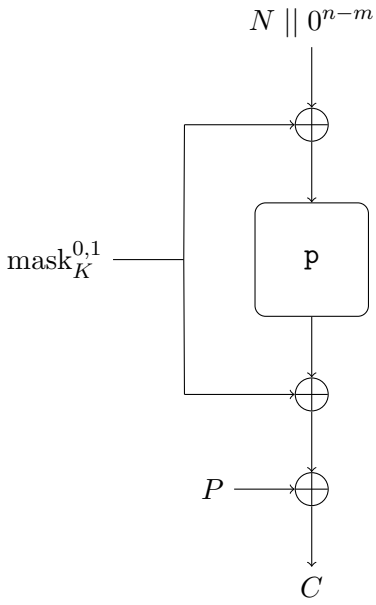

Figure 3.1: Encryption in `Elephant` v2 where $|P| = n$.

3.1.2 Attack

Using the nonce $N = 0^n$ and plaintext $P = 0^n$, the adversary queries ciphertexts $C_1, C_2, \ldots, C_\mu$, which takes μ construction queries. Each C is queried by the online oracles $E_{K_1}, \ldots, E_{K_\mu}$ and generated as:

$$C_i = \texttt{mask}_{K_i}^{0,1} + \texttt{p}(\texttt{mask}_{K_i}^{0,1} + 0^n)$$

for $i \in \{1, \ldots, \mu\}$, where

$$\texttt{mask}_{K_i}^{0,1} = \varphi_2^1 \circ \varphi_1^0 \circ \texttt{p}(K_i \mid\mid 0^{n-k}).$$

The adversary generates ciphertexts $C'_1, C'_2, \ldots, C'_q$ offline, using a different key guess $L_j \in \{0,1\}^k$ for $j \in \{1, \ldots, q\}$ for each ciphertext, but the same $N = P = 0^n$ as the online queries. Generating the ciphertexts takes $2q$ primitive queries, since the permutation `p` is used twice for each ciphertext. Each C' is generated as:

$$C'_j = \texttt{mask}_{L_j}^{0,1} + \texttt{p}(\texttt{mask}_{L_j}^{0,1} + 0^n)$$

for $j \in \{1, \ldots, q\}$, where

$$\texttt{mask}_{L_j}^{0,1} = \varphi_2^1 \circ \varphi_1^0 \circ \texttt{p}(L_j \mid\mid 0^{n-k}).$$

The adversary looks for two ciphertexts, C_i and C'_j, for which $C_i = C'_j$ holds,

for some $i, j \in \mathbb{N}$ with $i \leq \mu$ and $j \leq q$. This means both ciphertexts are likely to be generated using the same key. The probability of this event occurring is $\frac{\mu \cdot q}{2^k}$. The adversary outputs (L_j, i), and wins if $K_i = L_j$. If no collision is found the adversary loses. However, there is a possibility of having a false positive, where $C_i = C'_j$ even though $K_i \neq L_j$, with a probability of $\frac{1}{2^n - 1}$. Thus, there exists an adversary for $q = \frac{2^k}{\mu}$ such that $\mathbf{Adv}^{\text{keyrec}} \geq 1 - \frac{1}{2^n}$:

$$\begin{aligned}
\mathbf{Adv}^{\text{keyrec}}(\mathcal{A}) &= \mathbf{Pr}(\mathcal{A}^{\{E_{K_1}, \ldots, E_{K_\mu}, \mathtt{p}, \mathtt{p}^{-1}\}} = (L_j, i) \ : \ K_i = L_j) \\
&= \mathbf{Pr}((K_i = L_j) \wedge (\exists s, t : C_s = C'_t)) + \mathbf{Pr}((K_i = L_j) \wedge (\forall s, t : C_s \neq C'_t)) \\
&\geq \mathbf{Pr}((K_i = L_j) \wedge (\exists s, t : C_s = C'_t)) \\
&= \mathbf{Pr}(K_i = L_j \mid \exists s, t : C_s = C'_t)) \cdot \mathbf{Pr}(\exists s, t : C_s = C'_t) \\
&\geq (1 - \mathbf{Pr}(K_i \neq L_j \mid \exists s, t : C_s = C'_t)) \cdot \frac{\mu \cdot q}{2^n} \\
&\geq \left(1 - \frac{1}{2^n}\right) \cdot \frac{\mu \cdot q}{2^n}.
\end{aligned}$$

3.2 ASCON

3.2.1 Specification

A full specification of ASCON v1.2 is available at [3]. A part of the specification, as presented in [3], is shown here.

Let $k, r, c, a, b \in \mathbb{N}$. k is the length of the key, r the rate, c the capacity, a and b the number of rounds for permutations $\mathtt{p}^a$ and $\mathtt{p}^b$, where $\mathtt{p}^a : \{0,1\}^{320} \to \{0,1\}^{320}$ and $\mathtt{p}^b : \{0,1\}^{320} \to \{0,1\}^{320}$ are 320-bit permutations, and $c = 320 - r$. Define the initial value as:

$$IV_{k,r,a,b} = k \,||\, r \,||\, a \,||\, b \,||\, 0^{160-k}.$$

The encryption algorithm gets as input key $K \in \{0,1\}^k$, nonce $N \in \{0,1\}^{128}$, associated data $A \in \{0,1\}^*$, and plaintext $P \in \{0,1\}^*$. It outputs ciphertext $C \in \{0,1\}^{|P|}$, and a tag $T \in \{0,1\}^{128}$, as illustrated in Figure 3.2.

The decryption algorithm gets as input key K, nonce N, associated data A, ciphertext C and tag T. It outputs either plaintext P if the tag is correct, or a $\perp$-sign otherwise.

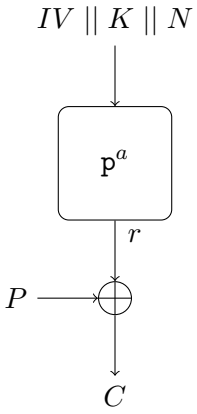

Figure 3.2: Encryption in `ASCON v1.2`, where $|P| = r$ and $A = \emptyset$.

3.2.2 Attack

Using the nonce $N = 0^{128}$ and plaintext $P = 0^r$, the adversary queries ciphertexts $C_1, C_2, \ldots, C_\mu$, which takes μ construction queries. Each C is queried by the online oracles $E_{K_1}, \ldots, E_{K_\mu}$ and generated as:

$$C_i = \mathtt{p}^a(IV \,||\, K_i \,||\, 0^{128})$$

for $i \in \{1, \ldots, \mu\}$.

The adversary generates ciphertexts $C'_1, C'_2, \ldots, C'_q$ offline, using a different key guess $L_j \in \{0,1\}^k$ for $j \in \{1, \ldots, q\}$ for each ciphertext, but the same $N = 0^{128}$ and $P = 0^r$ as the online queries. Generating the ciphertexts takes q primitive queries. Each C' is generated as:

$$C_j = \mathtt{p}^a(IV \,||\, K_j \,||\, 0^{128})$$

for $j \in \{1, \ldots, q\}$.

The adversary looks for two ciphertexts, C_i and C'_j, for which $C_i = C'_j$ holds, for some $i, j \in \mathbb{N}$ with $i \leq \mu$ and $j \leq q$. The probability of these ciphertexts being created by the same key is $\frac{\mu \cdot q}{2^k}$. The adversary outputs (L_j, i) and wins if $K_i = L_j$. If no collision is found the adversary loses. However, there is a possibility of having a false positive, where $C_i = C'_j$, but $K_i \neq L_j$, with a probability of $\frac{1}{2^r - 1}$. Thus, there exists an adversary for

$q = \frac{2^k}{\mu}$ such that $\mathbf{Adv}^{\text{keyrec}} \geq 1 - \frac{1}{2^r}$:

$$\begin{aligned}
\mathbf{Adv}^{\text{keyrec}}(\mathcal{A}) &= \mathbf{Pr}(\mathcal{A}^{\{E_{K_1},\ldots,E_{K_\mu},\mathtt{p},\mathtt{p}^{-1}\}} = (L_j, i) \,:\, K_i = L_j) \\
&= \mathbf{Pr}((K_i = L_j) \wedge (\exists s,t : C_s = C'_t)) + \mathbf{Pr}((K_i = L_j) \wedge (\forall s,t : C_s \neq C'_t)) \\
&\geq \mathbf{Pr}((K_i = L_j) \wedge (\exists s,t : C_s = C'_t)) \\
&= \mathbf{Pr}(K_i = L_j \mid \exists s,t : C_s = C'_t)) \cdot \mathbf{Pr}(\exists s,t : C_s = C'_t) \\
&\geq (1 - \mathbf{Pr}(K_i \neq L_j \mid \exists s,t : C_s = C'_t)) \cdot \frac{\mu \cdot q}{2^r} \\
&\geq \left(1 - \frac{1}{2^r}\right) \cdot \frac{\mu \cdot q}{2^r}.
\end{aligned}$$

3.3 ISAP

3.3.1 Specification

A full specification of ISAP is available at [2]. A part of the specification, as presented in [2], is shown here.

Let $k, n, r, r_B, s_H, s_B, s_E, s_K \in \mathbb{N}$. k is the length of the key and nonce, n the length of the permutation, r and r_B the rate in encryption and re-keying, respectively, where $r = n - 2k$ and $r_B = 1$. s_H, s_B, s_E and s_K specify the number of rounds the permutation is evaluated. Let $\mathtt{p}_E$, $\mathtt{p}_K$, $\mathtt{p}_B : \{0,1\}^n \to \{0,1\}^n$ be n-bit permutations. ISAP makes use of a re-keying function `IsapRk`. It takes as input a flag $f \in \{\mathtt{enc}, \mathtt{mac}\}$, a key K and nonce N, both of length k bits, and outputs subkey K^* of $n - k$ bits, as illustrated in Figure 3.4 for the specific case where $N = 0^k$ and $f = \mathtt{enc}$. Define the initial value of `IsapRk` as:

$$IV_{KE} = 3 \,||\, k \,||\, r \,||\, r_B \,||\, s_H \,||\, s_B \,||\, s_E \,||\, s_K \,||\, 0^*$$

The encryption algorithm gets as input k-bit key K and nonce N, and plaintext $P \in \{0,1\}^*$. It outputs ciphertext $C \in \{0,1\}^{|P|}$, as illustrated in Figure 3.3. The decryption algorithm is identical to encryption with the plaintext P and ciphertext C swapped. Authentication is done separately by the `IsapMac` function.

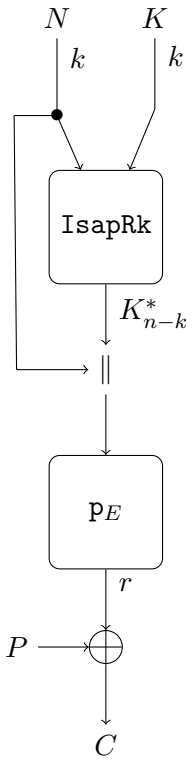

Figure 3.3: Encryption in ISAP v2.0, where $|P| = r$

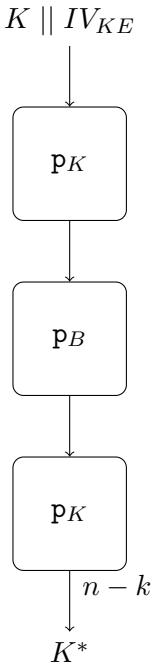

Figure 3.4: Re-Keying with IsapRk, where $N = Y = 0^k$ and $f =$ enc

3.3.2 Attack

Using the nonce $N = 0^k$ and plaintext $P = 0^r$, the adversary queries ciphertexts $C_1, C_2, \ldots, C_\mu$, which takes μ construction queries. Each C is queried by the online oracles $E_{K_1}, \ldots, E_{K_\mu}$ and generated as:

$$C_i = \mathtt{p}_E(0^k \mid\mid \mathtt{IsapRk}(0^k \mid\mid K_i^*))$$

for $i \in \{1, \ldots, \mu\}$, where

$$K_i^* = \mathtt{IsapRk}(0^k \mid\mid K_i).$$

The adversary generates ciphertexts $C'_1, C'_2, \ldots, C'_q$ offline, using a different key guess $L_j \in \{0,1\}^k$ for $j \in \{1, \ldots, q\}$ for each ciphertext, but the same $N = 0^k$ and $P = 0^r$ as the online queries. Generating the ciphertexts takes q primitives queries. Each C' is generated as:

$$C'_j = \mathtt{p}_E(0^k \mid\mid \mathtt{IsapRk}(0^k \mid\mid L_j))$$

for $j \in \{1, \ldots, q\}$.

The adversary looks for two ciphertexts, C_i and C'_j, for which $C_i = C'_j$ holds, for some $i, j \in \mathbb{N}$ with $i \leq \mu$ and $j \leq q$. The probability of these ciphertexts being created by the same key is $\frac{\mu \cdot q}{2^k}$. The adversary outputs (L_j, i) and wins if $K_i = L_j$. If no collision is found the adversary loses. However, there is a possibility of a false positive, where $C_i = C'_j$, but $K_i \neq L_j$, with a probability of $\frac{1}{2^r - 1}$. Thus, there exists an adversary for $q = \frac{2^k}{\mu}$ such that $\mathbf{Adv}^{\text{keyrec}} \geq 1 - \frac{1}{2^r}$:

$$
\begin{aligned}
\mathbf{Adv}^{\text{keyrec}}(\mathcal{A}) &= \mathbf{Pr}(\mathcal{A}^{\{E_{K_1}, \ldots, E_{K_\mu}, \mathtt{P}, \mathtt{P}^{-1}\}} = (L_j, i) \ : \ K_i = L_j) \\
&= \mathbf{Pr}((K_i = L_j) \wedge (\exists s, t : C_s = C'_t)) + \mathbf{Pr}((K_i = L_j) \wedge (\forall s, t : C_s \neq C'_t)) \\
&\geq \mathbf{Pr}((K_i = L_j) \wedge (\exists s, t : C_s = C'_t)) \\
&= \mathbf{Pr}(K_i = L_j \mid \exists s, t : C_s = C'_t)) \cdot \mathbf{Pr}(\exists s, t : C_s = C'_t) \\
&\geq (1 - \mathbf{Pr}(K_i \neq L_j \mid \exists s, t : C_s = C'_t)) \cdot \frac{\mu \cdot q}{2^r} \\
&\geq \left(1 - \frac{1}{2^r}\right) \cdot \frac{\mu \cdot q}{2^r}.
\end{aligned}
$$

Chapter 4

Conclusions

In the previous chapter an attack was created for three finalists, resulting in three security bounds. The security bounds allow an assessment of the cryptographic schemes concerning their multi-user security. The security bounds show similarities, since they largely depend on the same variables. For `Elephant` the following attack success probability was found:

$$\left(1 - \frac{1}{2^n}\right) \cdot \frac{\mu \cdot q}{2^n}$$

For both `ASCON` and `ISAP` the following attack success probability was found:

$$\left(1 - \frac{1}{2^r}\right) \cdot \frac{\mu \cdot q}{2^r}$$

All three probabilities are dependent on the values of μ and q. Translated to a real-life scenario this means the security of the cryptographic schemes, in terms of a multi-user attack, depend on the amount of devices being attacked and the amount of ciphertexts the attacker creates locally. A higher amount of devices and ciphertexts results in a larger chance of a key recovery.

The probability of `Elephant` also depends on the value of n, the block length. A higher block length means the ciphertext will be longer and it becomes harder to find collisions. Since a collision is necessary for a key recovery, a higher block length results in a more difficult attack. This increase is offset by a lower chance of finding a false positive, though this is insignificant.

The probabilities of `ASCON` and `ISAP` depend on the value of r, the rate. The effects are similar to the value of n for `Elephant` since r determines the length of the ciphertext. Finding a collision becomes harder resulting in a more difficult attack, regardless of the lower chance of encountering a false positive.

Bibliography

[1] Dobraunig C. Mennink B. Beyne T., Chen Y.L. Elephant v2, 2021. Submission to NIST.

[2] Eichlseder M. Mangard S. Mendel F. Mennink B. Primas R. Unterluggauer T. Dobraunig, C. Isap v2.0, 2021. Submission to NIST.

[3] Mendel F. Schläffer M. Dobraunig C., Eichlseder M. Ascon v1.2, 2021. Submission to NIST.

[4] Amina Harit, Abdellah Ezzati, and Rachid Elharti. Internet of things security: Challenges and perspectives. In *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, ICC '17, page 8, New York, NY, USA, 2017. Association for Computing Machinery.

[5] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the lightweight cryptography standardization process, 2018.

[6] National Institute of Standards and Technology. Block cipher techniques. `https://csrc.nist.gov/projects/block-cipher-techniques`, 2020.

[7] National Institute of Standards and Technology. Hash functions. `https://csrc.nist.gov/projects/hash-functions`, 2022.

[8] National Institute of Standards and Technology. Key establishment. `https://csrc.nist.gov/Projects/key-management/key-establishment`, 2022.

[9] National Institute of Standards and Technology. Lightweight cryptography. `https://csrc.nist.gov/projects/lightweight-cryptography`, 2022.