Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Христина Михајлоска

Лесна криптографија од перспектива на теоријата на квазигрупи

- Докторски труд -

Скопје, 2017

Ментор: Проф. д-р Данило Глогороски
Department of Telematics, Norwegian University of
Science and Technology, Trondheim, Norway

Коментор: Вонр. проф. д-р Весна Димитрова
Факултет за информатички науки и компјутерско инженерство
УКИМ, Скопје, Македонија

Членови на комисијата:

Проф. д-р Смиле Марковски, претседател
Факултет за информатички науки и компјутерско инженерство
УКИМ, Скопје, Македонија

Проф. д-р Данило Глогороски, ментор
Department of Telematics, Norwegian University of
Science and Technology, Trondheim, Norway

Вонр. проф. д-р Весна Димитрова, коментор
Факултет за информатички науки и компјутерско инженерство
УКИМ, Скопје, Македонија

Вонр. проф. д-р Александра Милева, надворешен член
Факултет за информатика, Универзитет "Гоце Делчев"
Штип, Македонија

Вонр. проф. д-р Анастас Мишев, член
Факултет за информатички науки и компјутерско инженерство
УКИМ, Скопје, Македонија

Апстракт

Во оваа докторска дисертација е даден комплетен дизајн и имплементација на нов алгоритам за автентикациска енкрипција π-Cipher. π-Cipher е еден од кандидатите кои успеаа да се пласираат во вториот круг на натпреварот за автентикациска енкрипција, CAESAR. π-Cipher е паралелен, инкрементален, со меѓу тагови, базиран на нонс шифрувач за автентикациска енкрипција со поврзани податоци. Дизајниран е со специјална намена да може да обезбеди приватност и интегритет на податоци кои користат ограничени ресурси, како и големи податоци во мирување или пренос. Има специјална опција за користење на таен дел од нонсот, што му обезбедува дополнително ниво на робусност и некое средно ниво на сигурност кога нонсот се реискористува. За разлика од стандардните шеми кои користат блок шифрувачи, π-Cipher како криптографска примитива користи пермутациска функција која е базирана на операциите собирање, ротација и логичко ИЛИ. Исто така во градењето на шемата искористени се неколку познати криптографски концепти, како: Енкриптирај-после-Автентицирај композициската шема за автентикациска енкрипција, XOR-MAC шемата за генерирање на таг и двојната сунѓер конструкција. π-Cipher обезбедува неколку нивоа на сигурност од 96 до 256 бита и нуди варијанти со различни должини на зборот. За лесни имплементации е наменета варијантата π16-Cipher096 со должина на збор од 16 бита и сигурност од 96 бита. За стандардни и имплементации со високи перформанси се наменети варијантите со 32 и 64 битни зборови.

Abstract

In this Ph.D. thesis is given a completely new design and implementation of an algorithm for authenticated encryption, π-Cipher. π-Cipher is one of the ciphers that participate in the second round vaof Competition for Authenticated Encryption: Security, Applicability, and Robustness - CAESAR. π-Cipher is parallel, incremental, with intermediate tags, nonce-based authenticated encryption cipher with associated data. It is designed with a special purpose to offer prin vacy and integrity of data that use constrained resources, as well as the big data at rest or in transit. It has a special option for using the secret part of the nonce (SMN), that gives the cipher an extra robustness feature and an intermediate level of nonce-misuse resistance. Instead of the standard schemes that use block ciphers, π-Cipher as a cryptographic primitive is a permutation-based scheme. Its permutation function is based on the operations: addition, rotation, and xor. Also, its design involves several solid cryptographic concepts such as a composition scheme for authenticated encryption, Encrypt-than-MAC, a counter based XOR-MAC scheme, and the duplex sponge construction. π-Cipher is designed for different word sizes and different security levels from 96 to 256 bits. For lightweight implementations, we propose the variant π16-Cipher096 where the word size is 16-bit and security level 96-bit. For standard and high-performance implementations, we propose the variants with 32-bit and 64-bit word sizes.

Благодарност

Сакам да изразам благодарност до ...

Содржина

Вовед **1**

Главни придобивки 3

Преглед на содржината 5

1 Основи на симетричната криптографија **9**

1.1 Блок шифрувачи 10

1.2 Код за автентикација на порака - MAC 13

1.3 Автентикациска енкрипција - AE 15

1.3.1 Композициски шеми за AE 16

1.3.2 Модови на операции кај AE 18

2 π-Cipher **21**

2.1 Нотации 22

2.2 Генерално за шифрувачот 25

2.2.1 Автентикациска енкрипција 26

2.2.2 Декрипција и верификација 34

2.3 π-функцијата 38

2.3.1 За константите на фунцијата π 43

3 Теорија на квазиīру$\overline{u}$и **47**

3.1 Основни $\overline{u}$оими и gефиниции 47

3.1.1 Алгебарска дефиниција на квазигрупи од ред 2^{64}, 2^{128} и 2^{256} 49

3.2 Дефиниција на функција π преку квазигрупна структура 56

4 Сигурносна анализа **59**

4.1 Докажлива сигурност 59
4.2 Формална дефиниција на π-Cipher 60
4.3 Сигурносен модел 61
4.4 Приватност на π-Cipher 66
4.5 Автентичност на π-Cipher 73
4.6 Сигурност на пермутацискатa функција π 78
4.6.1 Тестирање на ефектот на лавина на функцијата π 79
4.7 Криптоанализа на 16 битната варијанта на π-Cipher 85
4.8 Нивоа на сигурност 89

5 Мод на работа на π-Cipher за уреди со ограничена меморија **93**

5.1 Општо за онлајн AEAD шемите 93
5.2 Формална дефиниција 96
5.3 Доказ за сигурност 100

6 Дополнителни карактеристики на π-Cipher **103**

6.1 Блокови со произволно голема должина 103
6.2 Инкрементално својство 105

7 Споредбена анализа **111**

7.1 Споредба на функционални и сигурносни карактеристики 111
7.2 Споредба на перформанси 115
7.2.1 Број на операции 117
7.2.2 Меморија 117
7.2.3 Паралелизам 118
7.2.4 Софтвер 118
7.2.5 Хардвер 121

7.3 Рецензија за π-Cipher од анонимен рецензент на натпреварот CAESAR 121

8 Заклучок **127**

8.1 Преглед 127

8.2 Отворени истражувачки проблеми 130

Листа на слики

1-1 (EtM) Енкрипција-после-MAC 16
1-2 (MtE) MAC-после-енкрипција 17
1-3 (E&M) Енкрипција-и-MAC 18
1-4 Шема на модот на операции CCM кај AEAD [84] 19
1-5 Шема на модот на операции GCM кај AEAD [84] 19
1-6 Шема на модот на операции OCB кај AEAD [84] 20

2-1 Тројната компонента (triplex) 27
2-2 Двете варијанти на тројната компонента - triplex 27
2-3 Иницијализациска фаза 29
2-4 Процесирање на поврзаните податоци AD во a паралелни блокови 30
2-5 Процесирање на тајниот дел од нонсот SMN 32
2-6 Процесирање на оригиналната порака M во m паралелни блокови 33
2-7 Графичка репрезентација на ARX операцијата $*$ 39

3-1 Графичка репрезентација на една рунда од функцијата π. 58

4-1 Шема на π-Cipher со означени состојби s 64
4-2 Ефектот на лавина на операцијата $*$ за $\omega = 16$ 81
4-3 Ефектот на лавина на операцијата $*$ за $\omega = 32$ 81
4-4 Ефектот на лавина за операцијата $*$ за $\omega = 64$ 82
4-5 Ефектот на лавина за една рунда на функцијата π кога $\omega = 16$ ($Min = 120.732, \mathbf{Avg} = \mathbf{127.255}, Max = 128.731$) 82
4-6 Ефектот на лавина за една рунда на функцијата π кога $\omega = 32$ ($Min = 226.063, \mathbf{Avg} = \mathbf{256.765}, Max = 251.472$) 83

4-7 Ефектот на лавина за една рунда на функцијата π кога $\omega = 64$ ($Min = 400.88, \mathbf{Avg} = \mathbf{485.646}, Max = 515.76$) 83
4-8 Ефектот на лавина за една рунда на функцијата π кога $\omega = 16$ и $\mathbf{IS} = 0$ ($Min = 106.0, \mathbf{Avg} = \mathbf{128.238}, Max = 150.0$) 84
4-9 Ефектот на лавина за една рунда на функцијата π кога $\omega = 32$ и $\mathbf{IS} = 0$ ($Min = 201.0, \mathbf{Avg} = \mathbf{251.857}, Max = 292.0$) 84
4-10 Ефектот на лавина за една рунда на функцијата π кога $\omega = 64$ и $\mathbf{IS} = 0$ ($Min = 328.0, \mathbf{Avg} = \mathbf{481.805}, Max = 572.0$) 85
4-11 Функцијата π со една рунда за π16-Cipher096 86
4-12 O_2 ја погодуваме со пробување на сите можни 2^{64} вредности . . . 87
4-13 Сите зелени правоаголници се вредности добиени на единствен начин според дефиницијата на операцијата $*$ според соодветно погодена вредност за O_2 87
4-14 Напад на π16-Cipher096 со две рунди 89

5-1 Алгоритми за енкрипција и декрипција на сегментирана AEAD шема со SMN за π-Cipher 99

6-1 Процесирање на оригиналната порака M во m паралелни блокови кај инкременталниот мод на π-Cipher. Тука $UpdCtr_i$ е бројачот за ажурирање. 107

7-1 Споредба на најдобрите имплементации на секој од втората рунда кандидати на рамката за тестирање Supercop. Поврзаните податоци се 0 бајти, а пораката е 2048 бајти. π-Cipher е означен со pi . . 120
7-2 Позиција на π-Cipher според тестирањата на GMU API хардверските имплементации 122

Листа на табели

2.1 Основни карактеристики на сите варијанти од π-Cipher 22
2.2 Алгоритамски опис на ARX операцијата $*$ за 16 битни зборови. . . 40
2.3 Алгоритамски опис на ARX операцијата $*$ за 32 битни зборови. . . 41
2.4 Алгоритамски опис на ARX операцијата $*$ за 64–битни зборови. . 42
2.5 Генерален алгоритам за една рунда на π-функцијата 43
2.6 Константи за рундите за π16-Cipher 44
2.7 Константи за рундите за π32-Cipher 44
2.8 Константи за рундите за π64-Cipher 44

3.1 Парастрофи на квазигрупната операција $*$ 49
3.2 Вектор колони со константи $\mathbf{C}_1$ и $\mathbf{C}_2$ дадени во хексадецимална нотација за различни вредности на $\omega = 16, 32, 64$. 51
3.3 Два ортогонални Латински квадрати кои се користат за дефинирање на операција собирање по модул $2^\omega, \omega = 16, 32, 64$. 53
3.4 Матрици на соседство дефинирани од Латинските правоаголници $L_{1,1}$ и $L_{2,1}$ каде што елементите се земаат по колони 53
3.5 Латински квадрат поделен на различни блок дизајни за дефинирање на операција XOR на $\omega = 16, 32, 64$ битни зборови. 53
3.6 Матрици на соседство дефинирани од Латинските правоаголници $L_{3,1}$ и $L'_{3,2}$ каде што елементите се земаат по колони 54
3.7 Ротациските вектори кои се користат во пермутациите μ и ν за различни вредности на ω и q. 54

4.1 Листа на нивоа на сигурност кај сите варијанти на π-Cipher . . . 90

6.1 Карактеристика на широки блокови за π64-Cipher256 105
6.2 Инкрементално ажурирање на автентикациска енкрипција со π-Cipher. 108

7.1 Функционални и сигурносни карактеристики на кандидатите во втората рунда на натпреварот CAESAR 116
7.2 Преглед на бројот на операции за π-Cipher 117

Вовед

Живееме во време кога компјутерското пресметување ја постигнува својата трета фаза, во која сеприсутната технологија зазема сè позначајна улога. Ова ново компјутерско бранување исто така познато и како тивка технологија е време кога технологијата станува дел од нашиот секојдневен живот и ги зафаќа сите компјутерски дисциплини. Брзиот развој на секојдневните модерни уреди од една страна ветува многу предности, кои го прават нашиот живот полесен, а од друга страна пак, отвора огромен број прашања поврзани со безбедноста на тие уреди. Да се обезбеди сигурност и приватност кај ваквите уреди е од голема важност. Поради фактот што најголем дел од модерните криптографски алгоритми се дизајнирани за десктоп и серверски апликации, најголем дел од тие алгоритми не би можело да се имплементираат во уредите кои имаат ограничени ресурси или доколку би можело, тогаш нивните перформанси може да се неприфатливо слаби. Ова само по себе ја налага неопходноста на постоење алгоритми кои се фокусирани на безбедноста на малите уреди кои се дел на истражување на лесната криптографија (англ. lightweight cryptography). Лесната криптографија може да се смета како подобласт на криптографијата која се стреми да изнајде решенија посветени за уредите со ограничени ресурси. Во суштината на лесната криптографија е да се направи баланс помеѓу малите перформанси и сигурноста на уредите, односно како може да се постигне повисоко ниво на безбедност, а да се искористи само мала компјутерска моќ. Нејзина цел се широката палета од крајни кориснички уреди кои може да се имплементираат на мали хардверски чипови. Такви уреди се RFID-тагови, сензори во безжичните сензорски мрежи, паметни картички, индустриски микро-контролери итн. Индустријата ја увидела потребата за вакви алгоритми и голем број на криптографи имаат предложено

решенија за лесни проточни шифрувачи [42], [23], блок шифрувачи [17], [8], хеш функции [40], [7], [16] и во најново време шифрувачи за автентикациска енкрипција [45], [19], [39].

Во оваа докторска дисертација е опфатен делот на лесната симетрична криптографија со предлагање на нов дизајн за автентикациска енкрипција кој може да се примени кај системи со ограничени ресурси. Во дизајнот што е предложен во оваа докторска дисертација во еден дел се користат алгебарските структури, наречени квазигрупи.

Интересот за квазигрупите и нивната примена во криптографијата и теоријата на кодирање, датира уште многу одамна. Мотивацијата за употреба на квазигрупите во криптографијата се базира на фактот дека квазигрупите се на некој начин генерализирани пермутации и бројот на квазигрупите од ред n е поголем од $n!(n-1)!\ldots 1!$. Истражувањата покажале дека криптографските примитиви базирани на неасоцијативни структури имаат многу подобри карактеристики од оние базирани на асоцијативни.

Истражувачката група на ФИНКИ од Скопје во соработка со истражувачите од НТНУ од Трондхејм во последниве 15 години интензивно работи на полето на квазигрупите и нивната примена во теоријата на кодирање и криптографијата. Главните истражувања се вршат во насока на дефинирање нови алгоритми за блок и проточни шифрувачи, алгоритми за псевдо генератори на случајни низи, алгоритми за хеш функции, алгоритми со јавен клуч, алгоритми за кодови кои откриваат и поправаат грешки итн, во која што се постигнати значајни успеси. Меѓутоа за дефинирање на секоја од овие примитиви користени се различни типови на квазигрупни трансформации со внимателно избрани соодветни својства и ред на квазигрупите погодни за таа намена. Најинтересни во ова поле за овие истражувачи се квазигрупите од голем ред (кои се карактеризираат со многу добри криптографски својства) со кои се дефинирани разни примитиви. Ако се навратиме на тоа што зборувавме за лесната криптографија, лесно е да се направат големи и гломазни системи кои ќе бидат ултра јаки. Поради тоа што ваквите системи ќе користат многу ресурси и ќе бидат многу скапи, се оди кон откривање на мали алгебарски структури кои во доволна мера ќе бидат сигурни

и лесни за имплементација. За таа цел во оваа докторска дисертација се работи со малите квазигрупи (од ред 4) и се разгледува можноста за нивно искористување во остварување на доволно сигурна комуникација која што ќе ги задоволи основните принципи на лесната криптографија.

Како за почеток, да го наведеме фактот дека со помош на квазигрупи од ред 4 креиравме супституциски табели (англ. s-box) за лесни блок шифрувачи кои ги имаат истите криптографски својства како и оние на познатиот лесен блок шифрувач PRESENT [17]. За разлика од тоа што таму супституциските табели се добиени на начин на интензивно пребарување на сите пермутации од ред 16, ние со помош на квазигрупите од ред 4 понудивме алгоритамска техника на градење јаки супституциски табели. Оваа техника заедно со резултатите е детално опишана во трудот "Construction of Optimal 4-bit S-boxes by Quasigroups of Order 4 ", презентиран на конференцијата SECURWARE 2012 во Рим, Италија каде беше прогласен за најдобар труд [49].

Главни придобивки

Оваа докторска дисертација е резултат на истражувањето добиено во последните пет години во полето на лесната криптографија, дизајнот на симетрични криптографски алгоритми и употребата на квазигрупите во криптографијата. Главната мотивација за овој труд произлезе од повикот за натпревар за нов шифрувач за автентикациска енкрипција кој беше даден во Јануари 2013-та година. Натпреварот CAESAR, ја следи долгата традиција на фокусирани натпревари во симетричната криптографија и има за цел да изнајде портфолио од автентикациски шифрувачи кои ќе бидат предложени за стандардизација и наменети за широка употреба [13]. Токму затоа, најголем дел од истражувањето во оваа дисертација главно е во насока на дизајн на нова фамилија алгоритми за автентикациска енкрипција кои ќе бидат приспособени за користење во лесната и традиционалната криптографија. Притоа дизајниран е нов шифрувач за автентикациска енкрипција, π-Cipher кој доаѓа во четири варијанти во зависност од крајната намена и тоа: π16-Cipher096, π32-Cipher128, π64-Cipher128 и π64-

Cipher256.

Резултатите од истражувањата наведени во дисертацијата се прикажани пред домашната и светската научна јавност на повеќе домашни и меѓународни конференции. Повеќето од нив се веќе објавени во меѓународни и домашни списанија и зборници од конференции. Од оваа докторска дисертација произлезени се следните трудови и научни излагања:

1. Mohamed El-Hadedy, Hristina Mihajloska, Danilo Gligoroski, Amit Kulkarni, Dirk Stroobandt, Kevin Skadron: *A 16-Bit Reconfigurable Encryption Processor for π-Cipher*. IEEE International Parallel and Distributed Processing Symposium Workshops 2016, Chicago, IL, USA, pp: 162-171, Best Paper Award.

2. Simona Samardjiska, Hristina Mihajloska: *Towards Cryptanalysis of ARX based ciphers*. The 13th Conference for Informatics and Information Technology (CIIT 2016), Bitola, Macedonia.

3. Hristina Mihajloska, Danilo Gligoroski, Simona Samardjiska: *Reviving the Idea of Incremental Cryptography for the Zettabyte Era Use Case: Incremental Hash Functions Based on SHA-3*. Open Problems in Network Security - iNetSeC 2015, Zurich, Switzerland, LNCS Springer 2016, pp: 97-111.

4. Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, Rune E. Jensen, Daniel Otte: *π-Cipher v2*. Cryptographic competitions: CAESAR, 2015.

5. Hristina Mihajloska, Mohamed El Hadedy, Danilo Gligoroski, Kevin Skadron: *Lightweight version of π-Cipher*. NIST Lightweight Cryptography Workshop 2015.

6. Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, Rune E. Jensen: *π-Cipher: Authenticated Encryption for Big Data*. Secure IT Systems - 19th Nordic Conference, NordSec 2014, Tromsø, Norway, LNCS Springer 2014, pp: 110-128.

7. Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, Rune E. Jensen: *π-Cipher v1*. Cryptographic competitions: CAESAR, 2014.

8. Hristina Mihajloska: *The Hardware Performance of Authenticated Encryption Modes*. The 10th Conference for Informatics and Information Technology (CIIT 2013), Bitola, Macedonia, pp: 201-204.

9. Susanne Engels, Elif Bilge Kavun, Christof Paar, Tolga Yalçin, Hristina Mihajloska: *A Non-Linear/Linear Instruction Set Extension for Lightweight Ciphers*. 21st IEEE Symposium on Computer Arithmetic 2013, Austin, TX, USA, pp: 67-75.

Преглед на содржината

Докторската дисертација е поделена во 7 глави и заклучок.

Во првата глава се дадени основите на симетричната криптографија. Почнувајќи од блок шифрувачи, преку кодови за автентикација на порака па сè до автентикациска енкрипција што всушност е ѝ суштината на оваа докторска дисертација.

Во втората глава даден е детален опис на новата шема за автентикациска енкрипција π-Cipher. Воедно π-Cipher е еден од кандидатите во втората рунда на натпреварот за автентикациска енкрипција CAESAR. Конструиран е алгоритам кој може да се истакне со своите посебни карактеристики со кои се издвојува од другите кандидати на натпреварот. π-Cipher е паралелен, инкрементален, онлајн, сигурносно докажан, базиран на нонс шифрувач за автентикациска енкрипција со поврзани податоци (англ. associated data). Во оваа глава во детали се претставени функциите за енкрипција и автентикација и декрипција и верификација на податоците. Главна улога во конструкција на дизајнот има пермутацискатa функција π, за чиј опис е издвоена посебна подглава во рамките на оваа глава.

Во третата глава дадена е математичката позадина за градење на една јака крипто примитива како што во случајот е пермутацискатa функција π. Всушност функцијата π се темели на теоријата на квазигрупи која во последните 15 години

интензивно е испитувана за потребите на криптографијата од страна на домашни и странски научници. Дадена е алгебарска дефиниција на квазигрупи од ред 2^{64}, 2^{128} и 2^{256}, со кои се изградени трите варијанти за должина на зборови на π-Cipher (π16-Cipher, π32-Cipher, π64-Cipher соодветно). Во оваа глава дадена е ѝ алгебарската дефиниција на функцијата π преку квазигрупи и квазигрупни стринг трансформации.

Секој алгоритам во криптографијата покрај тоа што треба да има практична потврда за неговото функционирање, истиот треба да биде и теоретски докажан како сигурен алгоритам. За таа цел се користи криптографската парадигма - докажлива сигурност преку која се прави ригорозна анализа и проценка на сигурноста на криптографските шеми. Детална сигурносна анализа на нашиот дизајн е дадена во четвртата глава. Во неа е прикажана формалната дефиниција за π-Cipher и поставен е сигурносниот модел за негово докажување. Исто така прикажана и потврдена со анализа е ѝ сигурноста на пермутацискатa функција π. Во истата глава опишан е ѝ обид за разбивање на сигурноста на нашиот дизајн од страна на трети лица чија цел беше лесната варијанта на нашиот дизајн, π16-Cipher096.

Во петтата глава даден е опис на мод на работа на π-Cipher за уреди со ограничена меморија. Во оваа глава се фокусираме на побарувањата на лесната криптографија со детален опис на новиот мод на операции на шемата π-Cipher кој поддржува меѓу тагови. Со овој мод, овозможено е π-Cipher да изврши верификација на тагот за големи пораки дури и на уреди кои имаат ограничена меморија и без притоа да се направи ослободување на неверифициран дел од декриптираниот шифриран текст. Како и самата шема и овој мод е паралелен, поддржува онлајн енкрипција и декрипција и дава некое средно ниво на отпорност на шемата при реискористување на нонсот, како дополнителна робустна карактеристика на шифрувачот.

Како што веќе напоменавме, нашиот дизајн има дополнителни карактеристики со кои се издвојува од другите кандидати на натпреварот. Во шестата глава наведени се во детали овие карактеристики. Имено станува збор за лесната прилагодливост на шифрувачот кон големината на блокот на пораката што треба

да се обработува. Оваа карактеристика со која блоковите имаат произволна големина му дава на шифрувачот можност за адаптација во системи кои работат со големи податоци и каде блокот на пораката што треба да се енкриптира достигнува и до големина на еден диск сектор/страница. Исто така претставено е ѝ инкременталното својство на шифрувачот. Во контекст на тоа даден е посебен алгоритам со кој π-Cipher работи во инкреметален мод.

Во седмата глава која всушност е ѝ последна, дадена е споредбена анализа на нашиот дизајн со останатите кандидати на натпреварот. Во оваа глава се претставени предностите и недостатоците на функционалните и сигурносните карактеристики на шемата на π-Cipher во однос на досегашниот стандард за автентикациска енкрипција AES-GCM. Исто така е претставена детална анализа на софтверската и хардверската имплементација на π-Cipher во споредба со останатите кандидати на натпреварот со што може јасно да се види каде е местото што го заслужува овој дизајн во целосната слика на кандидати. Дадена е комплетна рецензија за нашиот шифрувач од страна на член од стручната комисија на CAESAR натпреварот, од која се гледа дека ѝ комисијата смета дека нашиот дизајн е солиден и не е разбиен, но заради екстра карактеристиките што ги има цената е наплатена во перформанси.

На крај е даден заклучок и насоки за понатамошните истражувања во оваа област.

Глава 1

Основи на симетричната криптографија

Криптологијата е наука која што во себе вклучува две интерактивни страни: криптографија и криптоанализа. Додека криптографијата се занимава со дизајнирање на механизми кои што обезбедуваат конкретни безбедносни критериуми, криптоанализата се обидува да ги разбие тие бебедносни механизми. Множеството безбедносни критериуми, зависи од тоа каде тие се применуваат, но во секој случај мора да ги запазат основните принципи како доверливост, податочен интегритет, автентикација и неодрекување, за да се обезбеди безбедна комуникација.

Симетричен криптографски алгоритам е метод на трансформација на најмалку два влезни параметри (клуч и порака - оригинален текст) и нивно пресликување во еден излезен параметар (шифриран текст). Ваквите алгоритми се нарекуваат симетрични, затоа што тие побаруваат тајниот клуч да биде познат за сите страни кои се вклучени во комуникацијата.

Генералиите за секој симетричен алгоритам се тоа што двете засегнати страни - испраќачот, познат во криптографијата како Алиса и примачот, познат во криптографијата како Боб, сакаат да комуницираат преку несигурен канал без да дозволат некој што ги прислушкува (во случајов Ева), да открие било каква информација од нивната комуникација. Секогаш се претпоставува дека претход-

но Алиса и Боб успеале да направат размена на тајниот клуч преку некој сигурен канал.

Генерално, еден симетричен криптосистем се состои од петорката $(\mathcal{K}, \mathcal{P}, \mathcal{C}, \mathscr{E}, \mathscr{D})$, каде што $\mathcal{K}$ е просторот на тајни клучеви, $\mathcal{P}$ е конечно множество на пораки (информацијата која што Алиса ја испраќа до Боб), $\mathcal{C}$ е конечно множество на шифрирани пораки (информацијата која што Боб ја прима), $\mathscr{E}$ е конечно множество од детерминистички енкриптирачки алгоритми, $\mathscr{D}$ е конечно множество од детерминистички декриптирачки алгоритми. За секој клуч $K \in \mathcal{K}$, постои енкриптирачки алгоритам $\mathcal{E}_K \in \mathscr{E}$ така што, $\mathcal{E}_K(M) = C$ и соодветен декриптирачки алгоритам $\mathcal{D}_K(C) = M$, односно $\mathcal{D}_K(\mathcal{E}_K(M)) = M$, каде што $M \in \mathcal{M}$ и $C \in \mathcal{C}$.

Со помош на симетричните криптографски алгоритми може да се задоволат првите три основни принципи од критериумите за безбедност и тоа: доверливост, интегритет и автентикација на пораката. Доверливоста се обезбедува со користење на енкрипциски алгоритми (блок шифрувачи и проточни шифрувачи), додека пак интегритетот и автентикацијата со користење на кодовите за автентикација на пораки.

1.1 Блок шифрувачи

Блок шифрувач е симетричен криптографски алгоритам кој што оперира над блокови од порака со фиксна големина. Притоа, блок шифрувачот, пресликува блок со големина од n бита од оригиналната порака, во блок од шифрирана порака со иста големина, каде што пресликувањето зависи од тајниот клуч. Блок шифрувачите се инверзибилни, што значи дека постои инверзна функција (функција за декрипција) чиј што влез е блок од шифриран текст, а излез е единствен блок од оригиналната порака. Најчесто големината на блоковите кај стандардните блок шифрувачи е 128 бита. За да се изврши криптирање на податоците, потребен е таен клуч. Големината на тајниот клуч може да биде 128 или 256 бита зависно од намената и бараната сигурност на блок шифрувачот. Стандард за блок шифрувач е веќе познатиот AES блок шифрувач [2] кој што веќе 15 години

опстојува без притоа никој да го разбие. Овој шифрувач користи блокови со големина од 128 бита и три различни должини за клуч во зависност од намената 128, 192 и 256 бита.

Во литературата, познати се два начини на дизајн на блок шифрувачи и тоа: супституциско-пермутациски мрежи и Фејстелови мрежи [58] [68]. Кај првиот начин, блок шифрувачот прима на влез блок од оригиналната порака и клуч. Истите ги проследува низ неколку рунди составени од супституциски дел и пермутациски дел, за на крај да произведе блок од шифрирана порака. Супституцискиот дел е одговорен за нелинеарно мешање на влезните битови, додека пак пермутацискиот дел е линеарен дел и создава дифузија на битовите добиени од супституцискиот дел. Главен двигател на сигурноста на ваквиот тип на блок шифрувачи е супституцискиот дел, каде одговорна улога имаат така наречените супституциски кутии (англ. S-boxes). Во фазата на декрипција потребно е да се искористат инверзни функции за супституција (инверзни супституциски кутии) и пермутација. Кај вториот начин, Фејстеловата мрежа, влезниот блок од оригиналната порака кој треба да се енкриптира се дели на два подеднакви дела. На првата половина се извршува определена рундовска функција со користење на потклуч и потоа излезот се XOR-ира (се применува логичката операција *исклучиво ИЛИ*) со втората половина. Потоа двете половини си ги заменуваат местата и постапката продолжува да се повторува колку што има број на рунди. Кај овој начин во фазата на декрипција не се употребува инверзна од дадената рундовска функција.

Блок шифрувачите сами за себе овозможуваат енкрипција само на порака со еден блок. Доколку пораката е поголема, односно се состои од два или повеќе блокови, тогаш блок шифрувачот мора да се користи со некој даден мод на операција за да може да се изврши енкриптирање на целата порака. Според тоа, мод на операција на блок шифрувач е алгоритам кој што опишува како последователно да се примени блок шифрувачот, за на сигурен начин да се обработи порака со должина поголема од еден блок. Најчесто модовите на операции на блок шифрувачи корстат дополнителни иницијализациски вектори IV (не повторувачки) кои имаат улога да овозможат добивање на различни шифрирани

пораки дури и кога влезната порака е иста [68]. Познати модови на операции кај блок шифрувачи се:

- ECB е скратеница од Electronic CodeBook [75]. Овој мод е наједноставен, а во исто време ѝ најнебезбеден. Притоа пораката се дели на блокови и секој блок посебно се енкриптира.

- CBC е скратеница од Cipher Block Chaining [75]. Овој мод работи на принцип, при што на секој блок од оригиналната порака се извршува операцијата XOR со претходниот шифриран блок без притоа истиот да се енкриптира. На овој начин секој шифриран блок од пораката зависи на некој начин од сите до тогаш процесирани блокови од оригиналната порака. Заради зголемување на нивото на сигурност, овој мод мора да се користи со иницијализациски вектор кој се процесира со првиот блок.

- OFB е скратеница од Output Feedback [75]. Овој мод го претвара блок шифрувачот во синхрон проточен шифрувач. Генерира блокови од проточни клучеви. Шифрираната порака се генерира на тој начин што врз блоковите од проточниот клуч и блоковите од оригиналната порака се извршува операцијата XOR. Како што е случајот кај другите проточни шифрувачи и тука со промена на еден бит во шифрираната порака, ќе се предизвика промена на бит на истата позиција и во оригиналната порака. Кај овој мод, секој нареден блок зависи од претходните, па затоа паралелизација при имплементација не може да се примени.

- CTR е скратеница од Counter [76]. Овој мод исто така го претвара блок шифрувачот во проточен шифрувач. Идејата тука е дека проточниот клуч се генерира со енкрипција на последователните вредности од бројачот (секвенца која што загарантирано нема да се повтори за доволно голема периода). CTR модот дозволува паралелизам и е еден од најшироко употребуваните модови за блок шифрувачи. Истиот е стандардизиран од страна на NIST во 2001 година и додаден кон листата на модови на блок шифрувачи.

1.2 Код за автентикација на порака - MAC

Код за автентикација на порака или MAC (англ. Message Authentication code), исто така познат како хеш функција со клуч или сума за проверка (англ. checksum), е широко употребуван крипто алгоритам во секојдневната пракса. Во однос на сигурносните начела овој алгоритам обезбедуба интегритет, односно автентичност на пораката, така што се спречува манипулирање со неа. Со оглед на тоа што овој алгоритам спаѓа во групата на симетрични алгоритми, не обезбедува неодрекување, односно гаранција за тоа кој е вистинскиот испраќач на пораката. Симетричниот клуч не е поврзан со конкретен испраќач/примач, туку со двете страни заедно. Според тоа одлуката за испраќањето на порака во случај на спор не може да се определи.

Во пракса, кодовите за автентикација на пораки се конструирани на два различни начини, од блок шифрувачи и од хеш функции. И во двата случаи кодовите за автентикација на пораки се карактеризираат со ефикасност и брзина на извршување.

MAC е функција која што како влез прима два аргументи, клуч K со фиксна должина и порака M со произволна должина, а како излез дава MAC вредност со фиксна должина. Математичката нотација за ова е $\mathbf{MAC}(K, M)$ или $\mathbf{MAC}_K(M)$. За да се автентицира пораката, испраќачот - Алиса ја испраќа не само пораката M до примачот - Боб, туку и MAC кодот пресметан за таа порака.

Техника за конструкција на MAC од блок шифрувач е CBC MAC [85]. Оваа техника произлегува од модот на операции за блок шифрувачи CBC, со таа разлика што сега меѓувредностите (шифрираниот текст) не се ослободуваат како резултат. Во овој случај се формира синџир од поврзани вредности меѓу блоковите што резултира со сигурна контролна сума на крај на пораката. Доколку се промени барем еден бит во некој од блоковите на оригиналната порака, тоа ќе предизвика да последниот енкриптиран блок се смени на начин што не може лесно да се предвиди без притоа да се знае тајниот клуч. Меѓутоа CBC MAC е сигурен само за пораки со фиксна должина. За пораки со произволна должина се користи неговата варијација CMAC (Cipher-based MAC) или OMAC1(One-key MAC) што претставува NIST предлог од 2005 година [77].

Опција за дизајн на MAC е да се користи криптографска хеш функција како градбена единка. Една таква конструкција е HMAC [71] која што стана многу популарна во праксата во последната декада. Како пример, HMAC се користи во интернет сигурносните протоколи, како што се TLS и IPSec. Основната идеја зад сите хеш-базирани кодови за автентикација на порака е дека клучот се хешира заедно со пораката. Постојат две општи конструкции и тоа:

$$MAC_K(M) = h(K||M)$$

наречена *MAC со тајен суфикс* и другата:

$$MAC_K(M) = h(M||K)$$

наречена *MAC со тајен префикс*, каде што h се однесува на безбедна криптографска хеш функција (MD5, SHA-1, SHA-2, SHA-3), а симболот $\|$ означува конкатенација на стрингови. Во двата случаи се покажало дека ваквите конструкции имаат слабости и не се доволно сигурни. Во 1996 година, Беларе и тимот, [71] покажале дека HMAC може да биде сигурна со користење на внатрешна и надворешна хеш функција. Оваа конструкција се смета за основна дефиниција на HMAC и е позната како NIST стандард од 2008 година [79]. Функцијата HMAC користи клуч K, порака M и хеш функција h за да произведе хеш базиран код за автентикација на пораки и тоа на следниот начин:

$$HMAC_K(M) = h((K \oplus opad)||h((K \oplus ipad)||M))),$$

каде што *opad* и *ipad* се соодветно функции за надворешно и внатрешно пополнување (проширување) на клучот K до определен број битови.

Криптографската јачина на HMAC се базира на јачината на основната хеш функција, големината на хеш излезот како и големината на тајниот клуч кој притоа се користи.

1.3 Автентикациска енкрипција - AE

Најчесто, во пракса, кога две страни комуницираат преку мрежа потребно е да бидат задоволени ѝ двете главни цели на симетричната криптографија, односно, доверливост и интегритет. Автентикацискатa енкрипција (AE) примарно е комбинација од енкрипција и автентикација и овозможува истовремено доверливост и интегритет на податоците кои се енкапсулирани.

Секоја симетрична шема, која што овозможува автентикациска енкрипција прима на влез оригинален текст и таен клуч, а на излез дава шифриран текст и краток автентикациски таг. Тагот се смета како контролна-сума на пораката и се користи за да се провери дали е примен коректниот шифриран текст од испраќачот преку верификациски алгоритам. Покрај тајниот клуч често се користи и дополнителен јавен податок - *нонс* (англ. nonce = number used once) кој претставува број што се употребува еднократно и може да биде некоја случајна вредност или бројач. Овој податок се користи по еднаш во секоја енкрипциска серија и овозможува да не се дозволи повторување на секвенци во енкриптираниот текст.

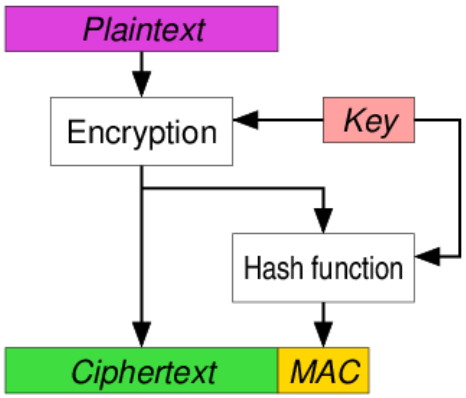
Во многу апликации постојат дополнителни податоци кои треба да бидат автентицирани, но треба да останат некриптирани. Како пример, може да се спомена мрежните пакети кои се состојат од порака и заглавие. Притоа, пораката мора да се енкриптира и во исто време автентицира, додека пак заглавието треба да остане во онаа форма во која е зададено (без енкрипција, само автентикација). Причината за ова е дека рутерите кои ги примаат и пренасочуваат пораките, мора да бидат во можност да го прочитаат заглавието на пакетот кој што пристигнува, за соодветно да може да го препратат понатака. Според овој практичен пример, многу од денешните шеми покрај оригиналниот текст имаат и поврзани податоци (англ. associated data) кои претставуваат влез во AE алгоритамот. Вакви те шеми се именуваат како автентикациска енкрипција со поврзани податоци и за оваа шема се користи теминот AEAD (англ. Authenticate Encryption with Associated Data), кој што е за прв пат фомрализиран од страна на Рогавеј [87].

1.3.1 Композициски шеми за AE

Секоја AE шема е едноставно комбинација од енкрипциски алгоритам и автентикациски алгоритам. Постојат три типови на композициски шеми кои нудат автентикациска енкрипција. Тие се разликуваат по тоа како овие два алгоритми се искомбинирани.

1. **Енкрипција-после-MAC (англ. Encrypt-then-Mac EtM)**

   Во оваа шема пораката прво се енкриптира, а после тоа тагот се пресметува со код за автентикација на порака MAC над добиениот шифриран текст. На страната на примачот, прво се прави верификација на добиениот таг, па доколку тие се поклопуваат пораката се декриптира. Во секој друг случај се враќа симбол за невалидна порака $\perp$. Слика 1-1 ја прикажува операцијата на EtM композициската шема за AE.

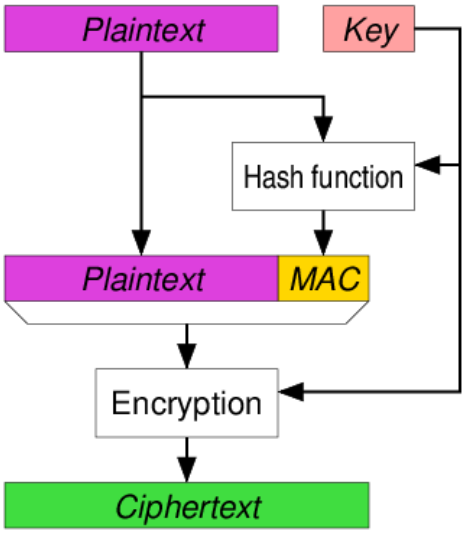


Слика 1-1: (EtM) Енкрипција-после-MAC

2. **MAC-после-Енкрипција (англ. MAC-then-Encrypt MtE)**

   Во оваа шема, прво се пресметува MAC над оригиналната порака. Добиениот таг се додава на оригиналната порака и заедно се енкриптираат, добивајќи проширен шифриран текст. На страната на примачот прво се извршува операцијата декрипција, за да се добие пораката и тагот, за потоа истиот да се верифицира. Доколку тагот е успешно верифициран, тогаш

декриптираниот текст се враќа, инаку симболот $\perp$. Слика 1-2 ја прикажува операцијата на MtE композициската шема за AE.

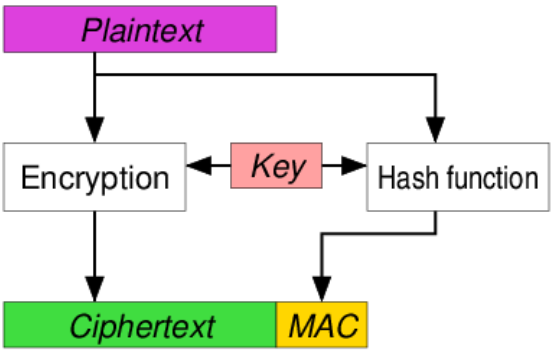


Слика 1-2: (MtE) MAC-после-енкрипција

3. **Енкрипција-и-MAC (англ. Encrypt-and-MAC E&M)**

   Во оваа шема оригиналната порака се енкриптира за да се добие шифрираниот текст, а исто така тагот се пресметува над оригиналната порака. Шифрираниот текст и тагот се праќаат до примачот. На негова страна прво се извршува операцијата декрипција, за да се добие декриптираната порака за која се пресметува MAC. Добиената вредност и тагот се споредуваат за верификација, доколку е успешна споредбата се враќа декриптираната порака, инаку се враќа симболот $\perp$. Слика 1-3 ја прикажува операцијата на E&M композициската шема за AE.

Во 2000 година, Беларе и Нампремпр [12] ги анализирале овие композициски шеми и покажале дека ако се користи енкрипциски алгоритам кој е отпорен на нападот со избран оригинален текст и код за автентикација на порака отпорен на нападот за фалсификат на оригинален текст, само шемата Енкрипција-после-MAC гарантира дека резултантниот дизајн ќе биде отпорен на нападот со избран



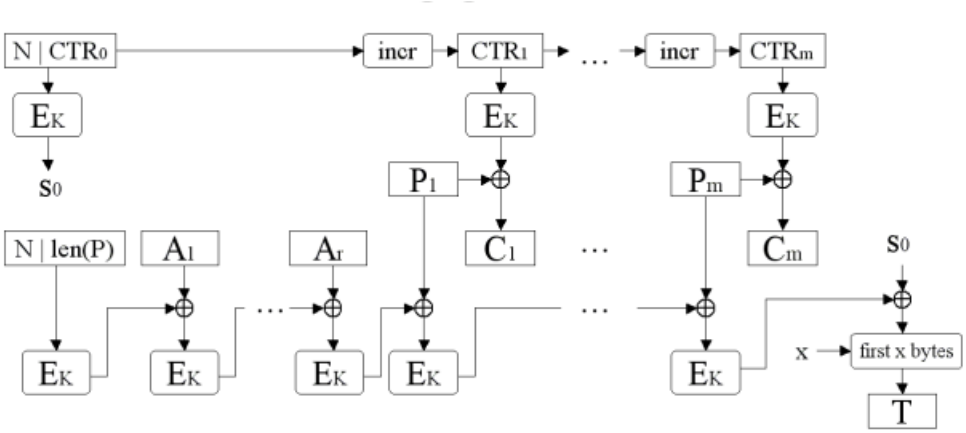
Слика 1-3: (E&M) Енкрипција-и-MAC

шифриран текст. Значи за било кои две криптографски примитиви кои ги задоволуваат наведените сигурносни побарувања, EtM секогаш дава како резултат сигурна автентикациско енкрипциска шема [59]. Оваа шема е најупотребувана за конструкција на AE/AEAD примитиви и токму затоа во понатамошното излагање, секое референцирање кон AE/AEAD би значело AE/AEAD со EtM како композициска шема.

1.3.2 Модови на операции кај AE

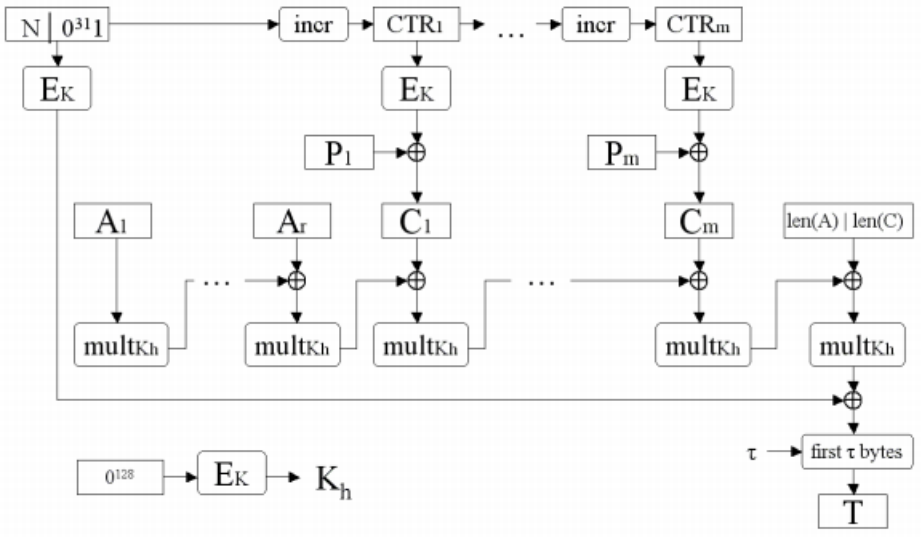
CCM е скратеница од Counter with CBC-MAC [101]. Овој мод на операциии всушност ги комбинира модот на операции со бројач за енкрипција на податоците (CTR mode) и CBC-MAC шемата за автентикација на истите. Имено, тој користи еден таен клуч во двете фази на автентикација и енкрипција. Исто така дозволува и обработка (автентикација) на поврзани податоци. CCM модот е дизајниран да работи со блокови од 128 бита, не е паралелен, не користи декрипциска функција, но побарува два повици на основната крипто примитива за да се овозможи енкрипција и автентикација на податоците. Не постои патент за овој тип на мод на операции за AE. Шемата за овој мод е дадена на Слика 1-4.

GCM е скратеница од Galois Counter Mode [67]. Овој мод користи мод на операции со бројач во фазата на енкрипција, додека пак автентикацискатa фаза е овозможена со множење во бинарното поле на Галоа, $GF(2^{128})$. Кај овој мод



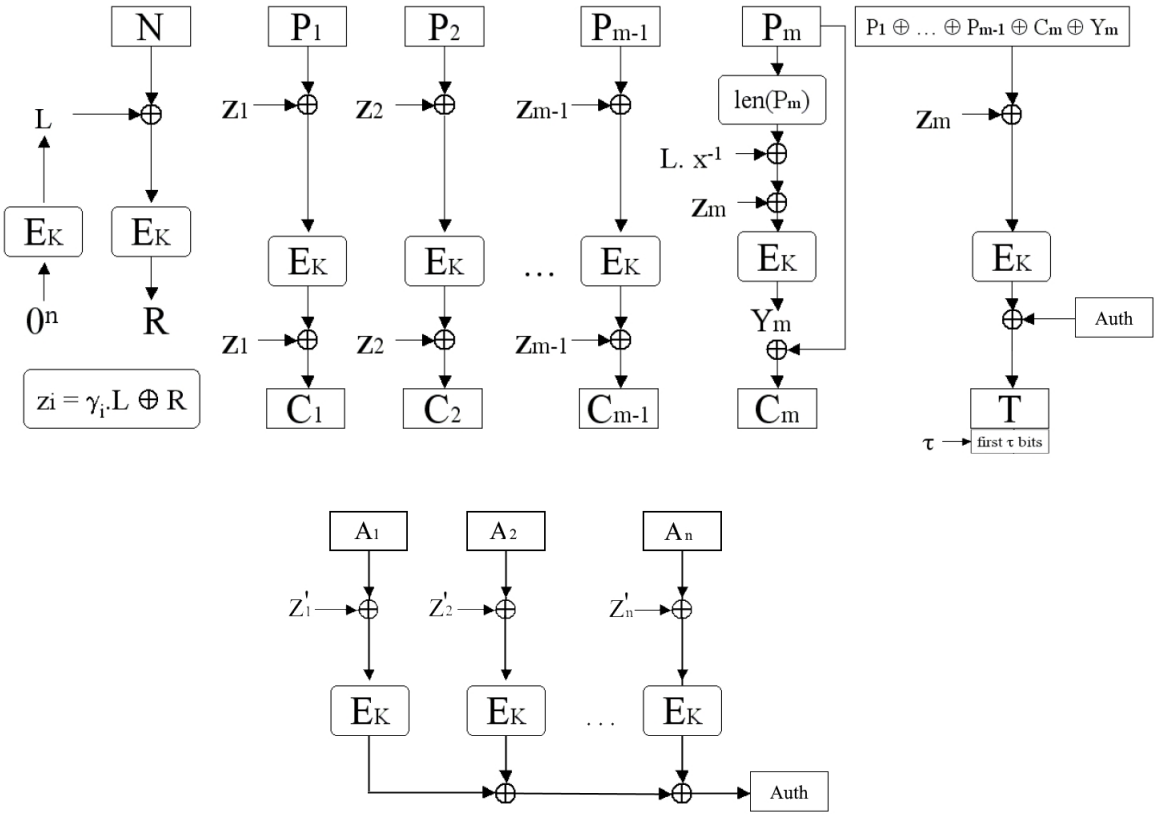
Слика 1-4: Шема на модот на операции CCM кај AEAD [84]

само еден повик на енкрипциска функција е доволен, не бара посебна функција за декрипција, овозможува паралелизам, автентикација на поврзани податоци, флексибилен е и ефикасен и што е многу важно не е патентиран. Шемата за овој мод е дадена на Слика 1-5.



Слика 1-5: Шема на модот на операции GCM кај AEAD [84]

OCB е скратеница од Offset CodeBook мод [88]. Базиран е на целосно паралелизирачкиот алгоритам IAPM [56]. И кај овој мод се користи множење во полето на Галоа, но на поедноставен и побрз начин отколку кај GCM. OCB



Слика 1-6: Шема на модот на операции OCB кај AEAD [84]

е без двоумење најбрзата и најоптимизираната шема за AE која што има можност за автентикација на поврзани податоци. Автентикацијата се прави независно од енкрипцијата и вредноста од автентикацијата на поврзаните податоци се додава на крај пред производството на тагот. Можноста за комплетната паралелизација се компензира со тоа што овој мод ги користи двете функции и за енкрипција и за декрипција на основната крипто примитива. Сите варијанти на OCB модот, OCB1, OCB2 и OCB3 подлежат на патент. Шемата за овој мод е дадена на Слика 1-6.

Глава 2

π-Cipher

π-Cipher е еден од кандидатите во втората рунда на натпреварот за автентикациска енкрипција CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness). Следејќи ја традицијата на успешни натпревари во симетричната криптографија, натпреварот CAESAR има за цел да изнајде не еден туку повеќе шифрувачи кои ќе понудат:

1. предности над веќе постоечкиот стандард AES-GCM [78],

2. широка распространетост и приспособливост кон различни уреди.

Првиот повик за натпреварот беше отворен во јануари 2013 година. Влез во натпреварот, во првата рунда добија 56 шифрувачи, предложени од различни тимови од целиот свет. Во текот на оваа рунда, 9 шифрувачи беа разбиени од страна на тимовите за криптоанализа. Во јули, 2015 година беа избрани шифрувачите кои продолжуваат понатака, односно во втората рунда на натпреварот. На овој избор, 28 кандидати ја добиа можноста да ја продолжат својата трка за финалето, меѓу кои се најде и нашиот дизајн π-Cipher [22].

π-Cipher е паралелен, инкрементален, сигурносно докажан, базиран на нонс шифрувач за автентикациска енкрипција со поврзани податоци кој што овозможува одредено ниво на отпорност кон втора слика на тагот. Дизајниран е од тим кој вклучува седум членови и тоа: Данило Глигороски (професор на Универзитетот во Трондхејм - Норвешка, NTNU), Христина Михајлоска (студент-

докторант на Универзитетот во Скопје - Македонија, УКИМ), Симона Самарџиска (доцент на Универзитетот во Скопје-Македонија, УКИМ), Хокон Јакобсен (студент-докторант на Универзитетот во Трондхејм - Норвешка, NTNU), Руне Јенсен (студент-докторант на Универзитетот во Трондхејм - Норвешка, NTNU), Мохамед Ел-Хадеди (истражувач на Универзитетот во Илиноис - САД) и Даниел Оте (студент на Универзитетот во Бохум - Германија, RUB).

Шифрувачот π-Cipher е деизјаниран за различни големини на зборови и различно ниво на сигурност. Предложените варијанти во официјалната документација на овој шифрувач [22] [95] се: π16-Cipher096, π32-Cipher128, π64-Cipher128 и π64-Cipher256 и истите се дадени во Табела 2.1 заедно со пропратните параметри.

	Збор ω (во битови)	$klen$ (во битови)	PMN (во битови)	SMN (во битови)	b (во битови)	N	$rate$ (во битови)	Tag T (во битови)	R
π16-Cipher096	16	96	32	0 or 128	256	4	128	$\leqslant 128$	3
π32-Cipher128	32	128	128	0 or 256	512	4	256	$\leqslant 256$	3
π64-Cipher128	64	128	128	0 or 512	1024	4	512	$\leqslant 512$	3
π64-Cipher256	64	256	128	0 or 512	1024	4	512	$\leqslant 512$	3

Табела 2.1: Основни карактеристики на сите варијанти од π-Cipher

2.1 Нотации

Во продолжение ќе дадеме опис на сите параметри и променливи кои се користат при спецификацијата на шифрувачот π-Cipher:

$\pi\omega$-Cipher$n$	AEAD шифрувач дефиниран со ω-бит зборови и n-бит сигурност.
$\omega = 16, 32, 64$	Должина на бинарен збор во битови користен во π-Cipher.
π функција	Главната пермутациска функција на шифрувачот.
M	Оригинална порака. Важи, $M = M_1 \|\| M_2 \|\| \ldots \|\| M_m$.
$mlen$	Големина на пораката $< 2^{64} - 1$ бајти.

m	Број на блокови на пораката.
K	Таен клуч.
$klen$	Должина на клучот K во бајти. Може да биде 12 бајти (96 бита), 16 бајти (128 бита) или 32 бајти (256 бита).
AD	Поврзани податоци (англ. associated data). Овие податоци не се енкриптираат, ниту декриптираат.
$adlen$	Големина на поврзаните податоци $< 2^{64} - 1$ бајти.
a	Број на блокови на поврзаните податоци. Важи, $AD = AD_1 \| \| AD_2 \| \| \ldots \| \| AD_a$.
IS	Внатрешна состојба, бијективно трансформирана од страна на π функцијата. Во оваа дисертација, кога се користи IS како заедничка интерна состојба за паралелните пресметувања, ќе ја користиме скратеницата CIS (*Common Internal State*).
I_i	Делче од интерната состојба IS. Ова делче е претставено како 4-торка од ω–бит зборови. $I_i = (I_{i1}, I_{i2}, I_{i3}, I_{i4})$.
N	Интерната состојба IS е поделена на N делчиња со големина од $4 \times \omega$ бита. N е секогаш парен број и важи $N \geqslant 4$. Значи, $IS = (I_1, I_2, \ldots, I_N)$.
b	Големина на IS во битови. Овој параметар е ограничен со следнава релација: $b = N \times 4 \times \omega$.
$rate$	Ратата на IS (гледано од аспект на сунѓер-конструкциите). $IS_{rate} = I_1 \ \| \ I_3 \ \| \ \ldots \ \| \ I_{N-1}$.
r	Големина на IS_{rate} во битови.

$capacity$	Капацитет на IS (гледано од аспект на сунѓер-конструкциите). $IS_{capaity} = I_2 \ \| \ I_4 \ \| \ \ldots \ \| \ I_N$.
c	Големина на $IS_{capacity}$ во битови.
$\|\|\|\|$	Оператор за наизменично спојување сè со цел правилно да се означи спојувањето на IS_{rate} и $IS_{capaity}$ кои ја формираат IS i.e., $IS = IS_{rate} \ \|\|\|\| \ IS_{capaity}$.
PMN	Јавен број на пораката (англ. Public message number). Должината $\|PMN\|$ во битови е ограничена со следнава релација: $8 \times klen + \|PMN\| + 8 \leqslant b$.
SMN	Таен број на пораката (англ. Secret message number). Должината $\|SMN\|$ во битови е ограничена со следнава релација: $\|SMN\| = 0$ or $\|SMN\| = r$.
$NONCE$	$NONCE = (PMN, SMN)$.
C	Шифриран текст.
$clen$	Големина на шифрираниот текст во бајти, каде $clen = mlen + \|SMN\| + tlen$ (во овој случај $\|SMN\|$ е дадено во бајти).
T	Автентикацискиот таг на пораката, нонсот и поврзаните податоци.
$tlen$	Должина на автентикацискиот таг во бајти. Ограничен е со следната зависност: $tlen \leqslant \frac{r}{8}$.
R	Променлив параметар кој означува број на рунди во π-функцијата.
ctr	64-бит внатрешен бројач. Се иницијализира од првите 64 бита од $IS_{capacity}$ на крај на Иницијализацискатa фаза.

$\boxplus_d$	Операција на собирање по компоненти на два d-димензионални вектори од ω-бит зборови во $(\mathbb{Z}_{2^\omega})^d$.

2.2 Генерално за шифрувачот

Инспирацијата за дизајнот на π-Cipher потекнува од неколку познати и веќе докажани концепти во криптографијата и тоа:

1. π-Cipher користи композициска шема за автентикациска енкрипција, Енкрипција-после-MAC (EtM).

2. Модот на операции е базиран на модот на операции со бројач за енкрипција на податоците (CTR mode) и шемата XOR MAC за автентикација [10]. Поради тоа што се покажало дека шемата XOR MAC не е доволно сигурна [100] и сé со цел да се постигне отпорност кон втора слика на тагот, наместо операцијата XOR, во пресметувањето на финалниот таг ние ја искористивме операцијата сума по компоненти во полето $(\mathbb{Z}_{2^\omega})^d$.

3. Наспроти стандардниот начин на користење на блок шифрувач како основна криптографска примитива, нашиот дизајн е базиран на пермутациска функција. Идејата за пермутациско базиран шифрувач за автентикациска енкрипција доаѓа од двојната сунѓер-конструкција предложена и докажана од страна на Бертони и тимот во трудот [14]. Оваа конструкција користи шема со две поминувања, каде првиот пат податоците се енкриптираат, а на второто поминување податоците се автентицираат.

4. Пермутацискатa функција π, која е основа на нашиот дизајн е базирана на ARX (Addition, Rotation and XOR) операциите, кои се постигнати со користење на алгебарските структури, квазигрупи.

Со правилна комбинација на овие основни концепти како и со додавање нови функционалности, π-Cipher нуди робустност како една од главните цели на нат-преварот. Имено, двојната сунѓер-конструкција е искористена во паралелен и

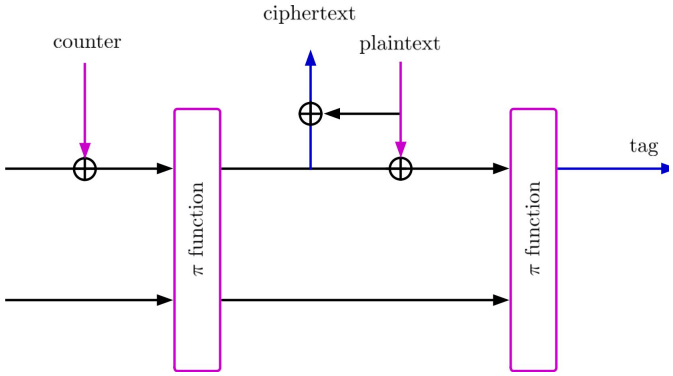
инкрементален мод со користење на бројач. Ново воведената операција, сума по компоненти во полето $(\mathbb{Z}_{2^\omega})^d$ овозможува тагот да биде единствен за секоја порака, односно да не може на едноставен начин да се направи втора слика за тагот. Со умешно вметнување на тајниот број (SMN) во дизајнот на π-Cipher, постигнавме некое средно ниво на сигурност при реискористување на нонсот за пораката. Исто така, употребата на алгебарските структури, квазигрупи при дизајнирањето на π-функцијата ни овозможува флексибилност во изборот на должината на блоковите за обработка на пораката. Во продолжение, ќе дадеме детално објаснување на двете главни процедури во работата на π-Cipher, односно на кој начин се врши енкрипција и автентикација на пораките од страната на испраќачот и на кој начин истите се верифицираат и декриптираат од страната на примачот.

2.2.1 Автентикациска енкрипција

Процедурата за енкрипција/автентикација на π-Cipher прима како параметри, клуч K со фиксна должина од $klen$ бајти, порака M со должина $mlen$ бајти и поврзани податоци AD со должина $adlen$ бајти. Шифрувачот користи јавен број PMN и таен број SMN. Излезот од процедурата за енкрипција/автентикација е шифриран текст C со должина од $clen$ бајти и соодветен таг T со фиксна должина од $tlen$ бајти. Должината $clen$ на шифрираниот текст C се пресметува како сума од должината на пораката, автентикацискиот таг и енкриптираниот таен број.

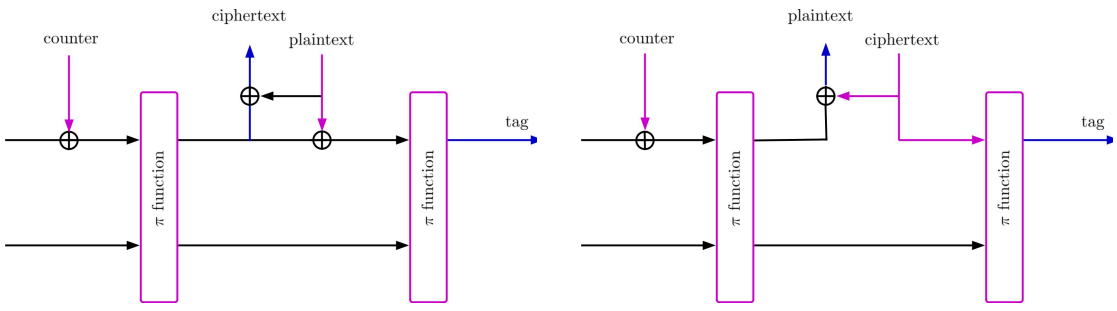
Главната градбена единка на операцијата енкриптирај-потоа-автентицирај, како и верифицирај-потоа-декриптирај, е нашата нова конструкција поврзана со двојната сунѓер-конструкција, наречена тројна компонента (англ. triplex component). Оваа компонента ја користи пермутацискатa функција π два пати, го инјектира бројачот во внатрешната состојба и ја енкриптира пораката. Излезот од оваа компонента е секогаш таг. Опционално, после првиот повик на функцијата π, може на излез да даде излезен стринг, кој може да биде шифриран блок или блок од пораката. Генералната скица на тројната компонента е претставена на Слика 2-1.

Поради разликата во процедурите за енкрипција/автентикација и декрипци-



Слика 2-1: Тројната компонента (triplex)

ја/верификација, постојат две различни варијанти на тројната компонента. Ги обележуваме како *e-triplex* за фазата на енкрипција и *d-triplex* за фазата на декрипција. Единствената разлика во овие компоненти е како тие го третираат влезниот стринг после првиот повик на π-функцијата. Во првата варијанта, влезниот стринг (оригиналната порака) се XOR-ира со моменталната внатрешна состојба од функцијата π и резултатот продолжува како влез во вториот повик на функцијата π. Кај варијантата *d-triplex*, влезниот стринг (шифрираната порака) директно се инјектира како дел од интерната состојба на π-функцијата и така новодобиената внатрешна состојба се проследува како влез на вториот повик на π-функцијата. Графичките репрезентации на овие две варијанти на тројната компонента се приложени на Слика 2-2.



(а) Тројната компонента за енкрипција (e-triplex)

(б) Тројната компонента за декрипција (d-triplex)

Слика 2-2: Двете варијанти на тројната компонента - triplex

Операцијата енкриптирај-потоа-автентицирај на π-Cipher може да се опише

во четири фази и тоа:

Фаза 1. Иницијализација. Во оваа фаза се врши спојување на клучот и јавниот број. Поради тоа што големината на внатрешната состојба IS на пермутацискатa финкција е b-бита, потребно е да се изврши пополнување на споениот стринг до оваа големина. Затоа, на крај на споениот стринг се додава 1 (во вид на бајт) и потоа се додаваат 0-и (во вид на бајти). Должината на вака добиениот стринг мора да биде еднаква на должината на внатрешната состојба на функцијата π изразена во бајти. Со други заборови, внатрешната состојба се иницијализира со $K||PMN||10^*$, каде што $|K||PMN||10^*| = |IS|$. Понатака, внатрешната состојба IS се ажурира со примена на функцијата π. Поради фактот што π-Cipher работи во паралелен мод, тој треба да има иницијална вредност за состојбата на паралелните нишки. Оваа состојба е заедничка внатрешна состојба за сите π функции на паралелните нишки и ја обележуваме со CIS. Истата ја иницијализираме на следниов начин:

$$CIS \leftarrow \pi(K||PMN||10^*).$$

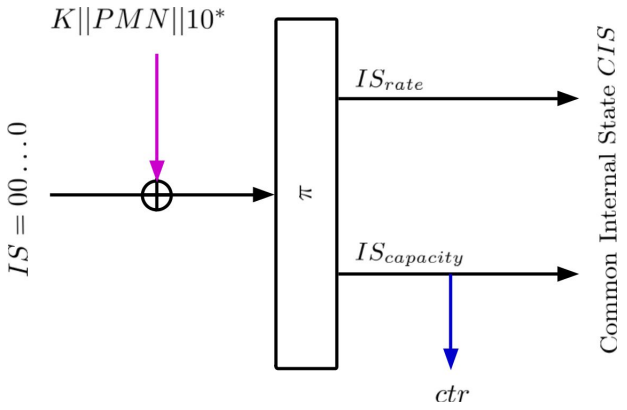
Следниот чекор во оваа фаза е иницијализација на бројачот ctr. Со оглед на тоа што внатрешната состојба се репрезентира со јавен и таен дел наречени рата и капацитет соодветно (овие поими се карактеристични за сунѓер-базираните конструкции [15]), ние ќе го иницијализираме бројачот со вредноста на првите 64 бита (во репрезентацијата little endian) од капацитетот (тајниот дел) на CIS.

Графичката репрезентација на фазата Иницијализација е дадена на Слика 2-3.

Фаза 2. Процесирање на поврзаните податоци. Поврзаните податоци AD се процесираат паралелно по блокови, користејќи ги e-triplex компонентите. Правилото за пополнување кај поврзаните податоци е следно:

$$Pad(AD) = AD_1||AD_2||\ldots||AD_a||10^*,$$

каде што 1 и 0 претставуваат бајти. Притоа, треба да се напомене дека, ако



Слика 2-3: Иницијализациска фаза

AD има должина која што е делива со големината на блокот, тогаш бројот на процесираните блокови од поврзаните податоци се зголемува за 1, значи $a \leftarrow a + 1$.

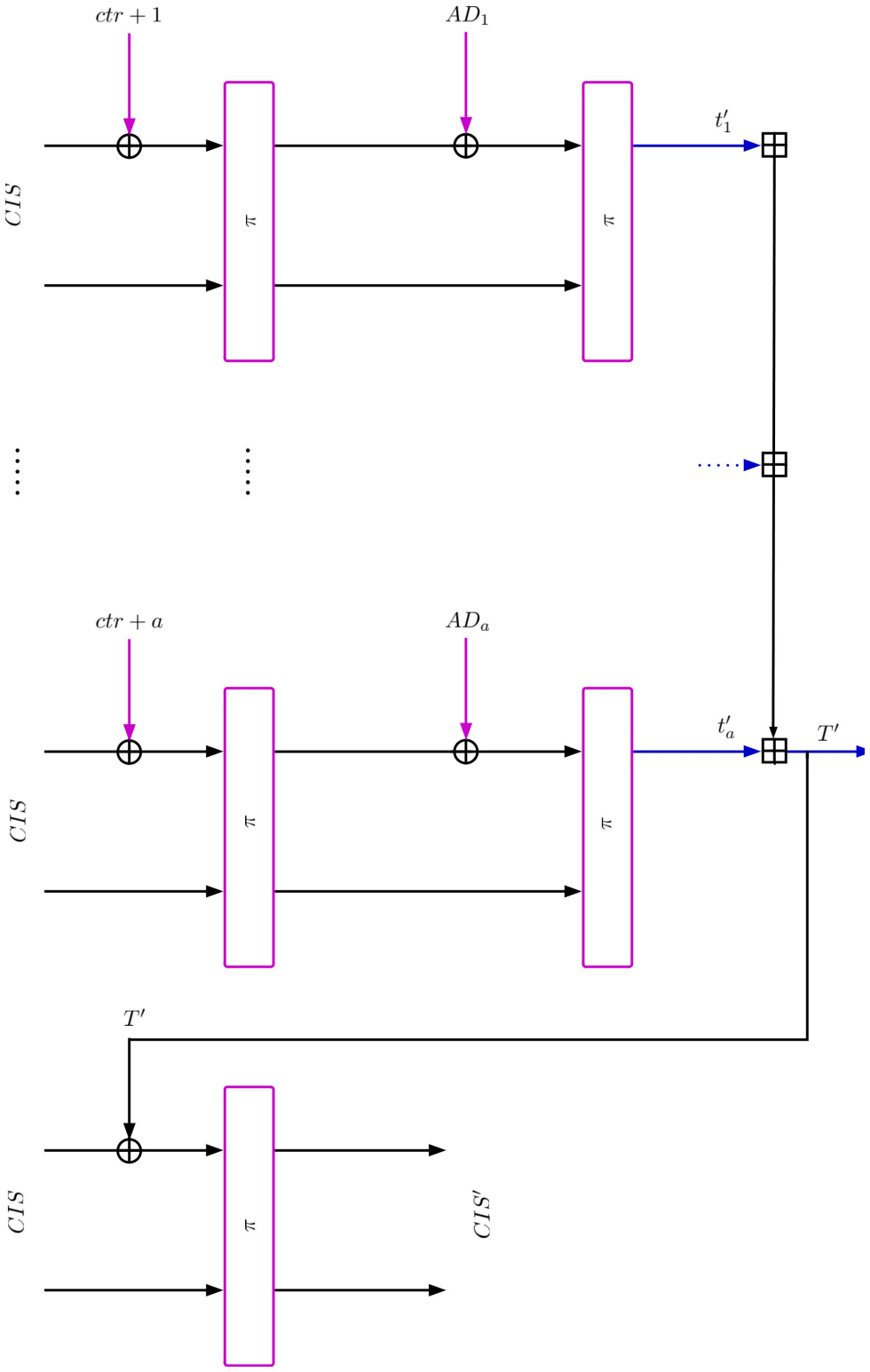
Кон секој блок AD_i, доделуваме единствен бројач пресметан како сума од иницијаната вредност на бројачот и редниот број i на блокот што се процесира. Влез во секоја e-triplex компонента е CIS, $ctr + i$ и AD_i, а излезот е меѓу таг t'_i.

Тагот T' за поврзаните податоци се пресметува како сума по компоненти $\boxplus_d$ од a вектори $t'_i \in (\mathbb{Z}_{2^\omega})^d$ со димензија d, каде што d е број на ω-бит зборови во еден блок. Со други зборови,

$$T' = \boxplus_{i=1}^{a}{}_d \; t'_i = t'_1 \boxplus_d t'_2 \boxplus_d \ldots \boxplus_d t'_a,$$

каде што $t'_i = (t'_{i_1}, t'_{i_2}, \ldots, t'_{i_d})$ е d-димензионален вектор од ω-бит зборови ($\omega = 16, 32, 64$). Во случај кога $N = 4$, димензијата на ваквите вектори е 8 ($d = 8$).

Во последниот чекор од оваа фаза, се ажурира вредноста на заедничката внатрешна состојба во CIS'. Ажурирањето на CIS се прави на тој начин што вредноста на CIS_{rate} се XOR-ира со тагот T' и после тоа се применува функцијата π.



Слика 2-4: Процесирање на поврзаните податоци AD во a паралелни блокови

Формалниот запис за ажурирање на вредноста на CIS е следен:

$$CIS' \leftarrow \pi(CIS_{rate} \bigoplus T' \ \ |||| \ \ CIS_{capacity}),$$

каде што симболот |||| означува оператор за наизменично спојување на деловите од заедничката внатрешна состојба. Имено, за $N = 4$, $CIS = \{CIS_1, CIS_2, CIS_3, CIS_4\}$, каде што CIS_1 и CIS_3 ја формираат ратата CIS_{rate}, додека пак CIS_2 и CIS_4 го формираат капацитетот $CIS_{capacity}$.

Целата фаза графички е прикажана на Слика 2-4.

Фаза 3. Процесирање на тајниот број. Оваа фаза може да се избегне ако должината на тајниот број е поставена да биде 0 (односно SMN е празен стринг). Ако SMN не е празен стринг, тогаш првиот чекор кој што треба да се преземе е повик на e-triplex компонентата со следните параметри $(CIS, ctr + a + 1, SMN)$. Излезот е парот (C_0, t_0). Наредниот чекор во оваа фаза е ажурирање на вредноста на CIS, која што се добива од моменталната внатрешна состојба по завршување со работа на e-triplex компонентата. Формално, ажурирањето на вредноста на CIS може да се претстави со следните два изрази:

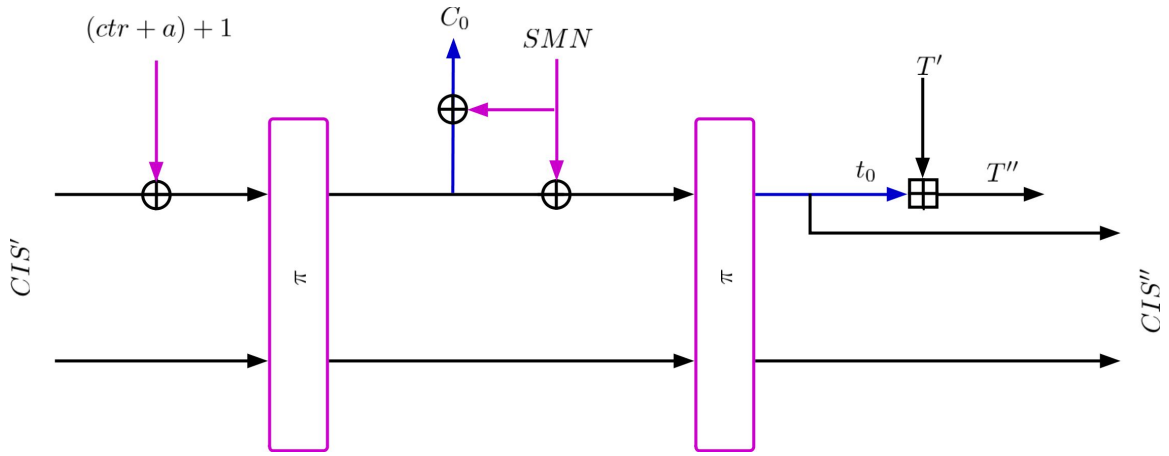
$$IS \leftarrow \pi(CIS'_{rate} \bigoplus (ctr + a + 1) \ \ |||| \ \ CIS'_{capacity}),$$
$$CIS'' \leftarrow \pi(IS_{rate} \bigoplus SMN \ \ |||| \ \ IS_{capacity}).$$

Тагот кој што се произведува од оваа фаза е

$$T'' = T' \boxplus_d t_0.$$

Графичката презентација на фазата за процесирање таен број е дадена на Слика 2-5.

Фаза 4. Процесирање на пораката. Пораката M се процесира блок по блок во паралелни нишки со помош на e-triplex компонентите. Правилото за пополнување на пораката M е следно:



Слика 2-5: Процесирање на тајниот дел од нонсот SMN

$$Pad(M) = M_1||M_2||\ldots||M_m||10^*,$$

каде што 1 и 0 претставуваат бајти. Притоа треба да се напомене дека ако M има должина која што е делива со големината на блокот, тогаш бројот на процесирани блокови од пораката се зголемува за 1, значи $m \leftarrow m + 1$. Кон секој блок од пораката M_j, додаваме единствен бројач за блокот, кој што се пресметува како:

$$ctr \leftarrow \begin{cases} ctr + a + j & \text{, ако } |SMN| = 0, \\ ctr + a + 1 + j & \text{, ако } |SMN| = r, \end{cases}$$

каде што j е редниот број на блокот од пораката кој што се процесира и притоа $0 < j \leqslant m$. Влезот во секоја компонента од e-triplex е CIS'', бројачот за соодветниот блок ctr и M_j, а излезот е парот (C_j, t_j). По дефиниција ставивме дека должината на последниот блок од шифрираната порака C_m е ист со должината на непополнетиот последен блок од оригиналната порака M_m т.е., $|C_m| = |M_m|$.

Финалниот таг T, се добива како сума $\boxplus_d$ од сите меѓу тагови на паралелните нишки t_j и претходно пресметаната вредност за тагот T''.

$$T = T'' \boxplus_d t_1 \boxplus_d \ldots \boxplus_d t_j \boxplus_d \ldots \boxplus_d t_m,$$

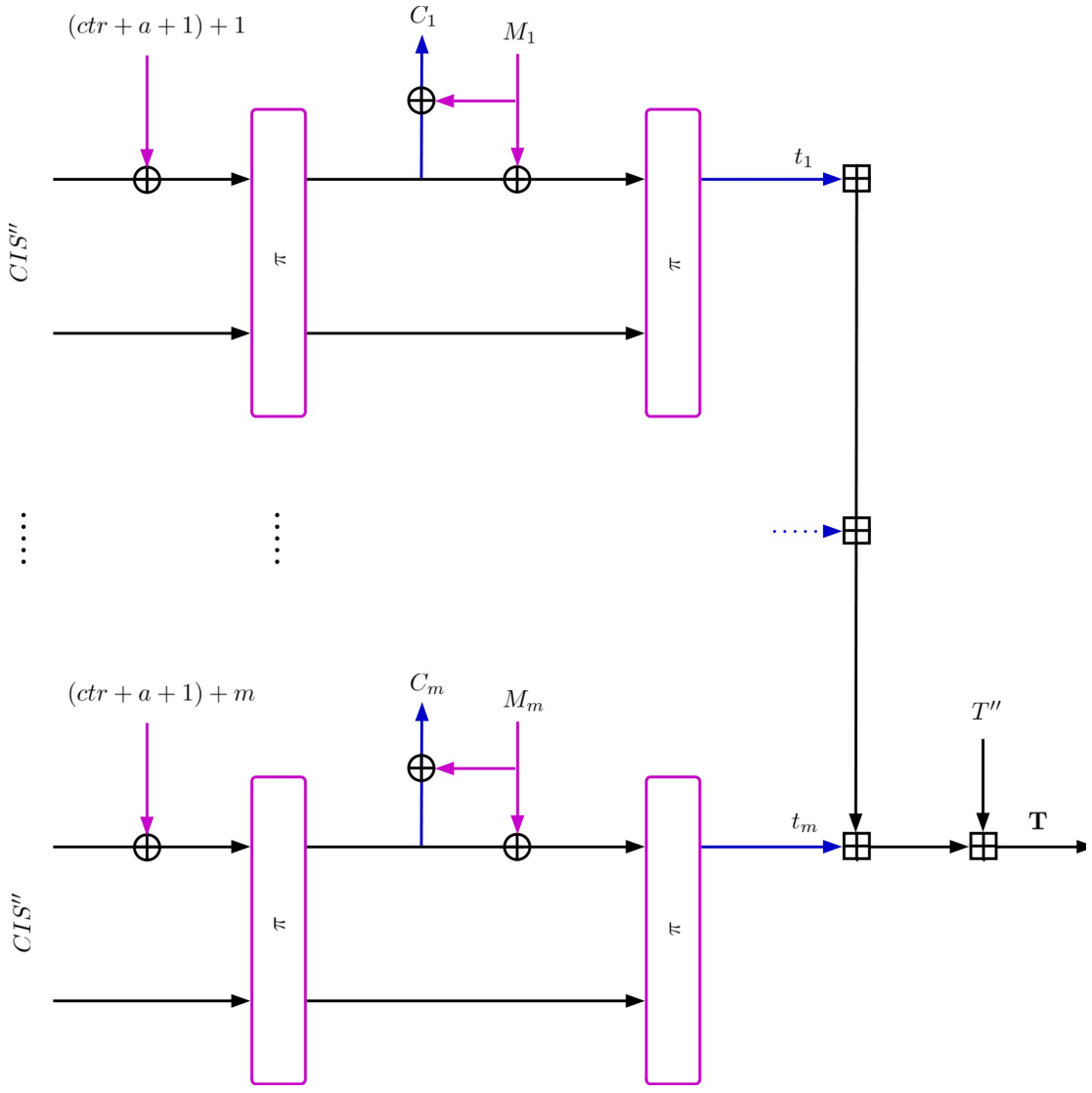
каде што $t_j = (t_{j1}, t_{j2}, \ldots, t_{jd})$ е d-димензионален вектор од ω-бит зборови ($\omega = 16, 32, 64$). Во овој случај кога $N = 4$, димензијата на векторот е $d = 8$.

Оваа фаза графички е опишана на Слика 2-6.

Излезот од процедурата енкрипција/автентикација е шифрирана порака

$$C = C_0||C_1||\ldots||C_m||T.$$

Делот за енкрипција и автентикација е прикажан со Алгоритам 1.



Слика 2-6: Процесирање на оригиналната порака M во m паралелни блокови

2.2.2 Декрипција и верификација

Процедурата за декрипција/верификација прима на влез клуч K, поврзани податоци AD, шифрирана порака C како и јавен број PMN и таг T. Соодветно оваа процедура враќа декриптиран пар (SMN, M) ако претходно тагот бил верифициран или во спротивно неважечки симобол $\perp$. Процедурата за декрипција/верификација е дефинирана аналогно на претходната за енкрипција/автентикација. И кај оваа процедура присутни се претходно опишаните четири фази со тоа што првите две (Иницијализација и Процесирање на поврзаните податоци) се комплетно исти, а разлика има само во енкрипциските фази. Декрипцијата на тајниот број е овозможена во фазата *Процесирање на тајниот број*. Оваа фаза во делот за декрипција и верификација е опишана на следниот начин:

фаза 3. Процесирање на тајниот број. Доколку должината на тајниот број не е 0 (односно SMN не е празен стринг), тогаш првиот чекор кој што треба да се преземе е повик на d-triplex компонентата со параметрите $(CIS, ctr + a + 1, C_0)$, каде што C_0 е првиот блок од шифрираната порака. Излезот од компонентата е парот (SMN, t_0). Наредниот чекот во оваа фаза е ажурирање на вредноста на CIS, која што се добива од моменталната внатрешна состојба по завршување со работа на d-triplex компонентата. Формално, ажурирањето на вредноста на CIS може да се претстави со следните два изрази:

$$IS \leftarrow \pi(CIS_{rate} \bigoplus (ctr + a + 1) \;\; |||| \;\; CIS_{capacity}),$$
$$CIS \leftarrow \pi(C_0 \;\; |||| \;\; IS_{capacity})$$

Пресметувањето на тагот е исто $T'' = T' \boxplus_d t_0$.

фаза 4. Процесирање на шифрираната порака. Шифрираната порака $C = C_1 || \ldots || C_j || \ldots || C_m$ се процесира блок по блок во паралелни нишки со помош на d-triplex компонентите. Кон секој блок од шифрираната порака C_j

додаваме единствен бројач за блокот, кој што се пресметува како:

$$ctr \leftarrow \begin{cases} ctr + a + j & \text{, ako } |C_0| = 0, \\ ctr + a + 1 + j & \text{, ako } |C_0| = r, \end{cases}$$

каде што j е редниот број на блокот од пораката кој се декриптира и притоа $0 < j \leqslant m$. Влезот во секоја компонента од d-triplex е CIS, бројачот за соодветниот блок ctr и C_j, а излезот е парот (M_j, t_j). Според дефиниција, должината на последниот блок од шифрираната порака C_m е ист со должината на непополнетиот последен блок од оригиналната порака M_m т.е., $|C_m| = |M_m|$.

Финалниот таг T, се добива како сума $\boxplus_d$ од сите меѓу тагови на паралелните нишки t_j и претходно пресметаната вредност за тагот T''.

$$T = T'' \boxplus_d t_1 \boxplus_d \ldots \boxplus_d t_j \boxplus_d \ldots \boxplus_d t_m,$$

каде што $t_j = (t_{j1}, t_{j2}, \ldots, t_{jd})$ е d-димензионален вектор од ω-бит зборови. Во овој случај кога $N = 4$, димензијата на векторот е $d = 8$.

На крај пресметаниот таг се споредува со тагот кој што се добива од процедурата на енкрипција/автентикација. Притоа, ако тие се поклопуваат декриптираната порака се враќа како резултат, во спротивно се враќа симбол за невалидна верификација $\perp$.

Процедурата декрипција/верификација е опишана со Алгоритам 2.

Алгоритам 1: Енкрипција и автентикација $\mathcal{E}_K(PMN, AD, SMN, M)$

1 Иницијализација: $K||PMN||10^*$ каде што $|K||PMN||10^*| = b$
2 $CIS \longleftarrow \pi(K||PMN||10^*)$
3 $ctr \longleftarrow [CIS_{capacity}]^{64}$
4 $AD = AD_1||AD_2||\ldots||AD_{a-1}||AD_a$
5 $AD \longleftarrow Pad(AD)$
6 $M = M_1||M_2||\ldots||M_{m-1}||M_m$ и $m_{old} \longleftarrow m$, $lastBlockLength \longleftarrow |M_m|$
7 $M \longleftarrow Pad(M)$
8 **for** $i = 1$ **to** a **do**
8.1 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + i)||||CIS_{capacity})$
8.2 $IS \longleftarrow \pi(IS_{rate} \oplus AD_i||||IS_{capacity})$
8.3 $t'_i \longleftarrow IS_{rate}$
8.4 $T' \longleftarrow T' \boxplus_d t'_i$
9 $ctr \longleftarrow ctr + a$
10 $CIS \longleftarrow \pi(CIS_{rate} \oplus T'||||CIS_{capacity})$
11 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + 1)||||CIS_{capacity})$
12 $C_0 \longleftarrow IS_{rate} \oplus SMN$
13 $IS \longleftarrow \pi(IS_{rate} \oplus SMN||||IS_{capacity})$
14 $t_0 \longleftarrow IS_{rate}$, $T'' \longleftarrow T' \boxplus_d t_0$, $T \longleftarrow T''$
15 $CIS \longleftarrow IS$
16 $ctr \longleftarrow ctr + 1$
17 **for** $i = 1$ **to** m **do**
17.1 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + i)||||CIS_{capacity})$
17.2 $C_i \longleftarrow IS_{rate} \oplus M_i$
17.3 $IS \longleftarrow \pi(IS_{rate} \oplus M_i||||IS_{capacity})$
17.4 $t_i \longleftarrow IS_{rate}$
17.5 $T \longleftarrow T \boxplus_d t_i$
18 $C \longleftarrow C_0||C_1||\ldots||C_{m_{old}}||T$, каде што $|C_{m_{old}}| = lastBlockLength$
19 **return** C

Алгоритам за пополнување: $Pad(S)$

1 $S = S_1||S_2||\ldots||S_l$
2 **if** $|S_l| < r$ **then**
3 $S_l \longleftarrow S_l||10^*$, каде што $|S_l| = r$
4 **else**
5 $S_{l+1} \longleftarrow 10^*$, каде што $|S_{l+1}| = r$ и $l \longleftarrow l + 1$
6 **return** S

Алгоритам 2: Декрипција и верификација $\mathcal{D}_K(PMN, AD, C, T)$

1 Иницијализација: $K||PMN||10^*$ каде што $|K||PMN||10^*| = b$
2 $CIS \longleftarrow \pi(K||PMN||10^*)$
3 $ctr = [CIS_{capacity}]^{64}$
4 $AD = AD_1||AD_2||\ldots||AD_{a-1}||AD_a$
5 $AD = Pad(AD)$
7 $C = C_0||C_1||\ldots||C_m$, каде што $|C_i| = r$ за $0 \leqslant i < m$ и $lastBlockLength \longleftarrow |C_m|$
8 **for** $i = 1$ **to** a **do**
8.1 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + i)||||CIS_{capacity})$
8.2 $IS \longleftarrow \pi(IS_{rate} \oplus AD_i||||IS_{capacity})$
8.3 $t'_i \longleftarrow IS_{rate}$
8.4 $T' \longleftarrow T' \boxplus_d t'_i$
9 $ctr \longleftarrow ctr + a$
10 $CIS \longleftarrow \pi(CIS_{rate} \oplus T'||||CIS_{capacity})$
11 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + 1)||||CIS_{capacity})$
12 $SMN \longleftarrow IS_{rate} \oplus C_0$
13 $IS \longleftarrow \pi(C_0||||IS_{capacity})$
14 $t_0 \longleftarrow IS_{rate}$, $T'' \longleftarrow T' \boxplus_d t_0$, $\mathtt{T} \longleftarrow T''$
15 $CIS \longleftarrow IS$
16 $ctr \longleftarrow ctr + 1$
17 **for** $i = 1$ **to** $m - 1$ **do**
17.1 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + i)||||CIS_{capacity})$
17.2 $M_i \longleftarrow IS_{rate} \oplus C_i$
17.3 $IS \longleftarrow \pi(C_i||||IS_{capacity})$
17.4 $t_i \longleftarrow IS_{rate}$
17.5 $\mathtt{T} \longleftarrow \mathtt{T} \boxplus_d t_i$
18 **if** $(|C_m| < r)$ **then**
19 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + m)||||CIS_{capacity})$
20 $M_m \longleftarrow IS_{rate} \oplus C_m$
21 $IS_{rate} \longleftarrow IS_{rate} \oplus 0^{lastBlockLength}10^*$
22 $IS \longleftarrow \pi(C_m||[IS_{rate}]_{(r-m)}||||IS_{capacity})$
23 $t_m \longleftarrow IS_{rate}$
24 $\mathtt{T} \longleftarrow \mathtt{T} \boxplus_d t_m$
25 **else**
26 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + m)||||CIS_{capacity})$
27 $M_m \longleftarrow IS_{rate} \oplus C_m$
28 $IS \longleftarrow \pi(C_m||||IS_{capacity})$
29 $t_m \longleftarrow IS_{rate}$
30 $\mathtt{T} \longleftarrow \mathtt{T} \boxplus_d t_m$
31 $IS \longleftarrow \pi(CIS_{rate} \oplus (ctr + m + 1)||||CIS_{capacity})$
32 $IS \longleftarrow \pi(IS_{rate} \oplus 10^*||||IS_{capacity})$
33 $t_{m+1} \longleftarrow IS_{rate}$
34 $\mathtt{T} \longleftarrow \mathtt{T} \boxplus_d t_{m+1}$
35 $M \longleftarrow M_1||M_2||\ldots||M_m$
31 **if** $T \neq \mathtt{T}$ **then**
32 **return** $\perp$
33 **else return** (SMN, M)

2.3 π-функцијата

Главната улога во секоја сунѓер-конструкција ја игра пермутациската функција и скоро целата сигурност се темели на неа. Целта при нашиот дизајн беше првично да се обезбеди доволно јака пермутациска функција, која што за различни вредности на параметарот ω (должина на збор во битови) овозможува различни особини т.е. да биде ефикасна кога $\omega = 64$ и лесна кога $\omega = 16$.

Во дизајнот на π-Cipher оваа пермутациска функција ја нарековме π функција. Таа користи слични операции како и операциите на нејзиниот претходник, хеш функцијата Edon-R [36]. Разликата е тоа што овде влез во пермутациската функција се 4-торки наместо претходно што се користеа 8-торки. Пермутацијата оперира над состојба од b бита и ја ажурира внатрешната состојба преку секвенца од R поврзани трансформации - рунди. Внатрешната состојба IS може да се претстави како листа од N четворки, секоја со должина од ω бита, каде што $b = N \times 4 \times \omega$, односно

$$IS = (\underbrace{(IS_{11}, IS_{12}, IS_{13}, IS_{14})}_{I_1}, \underbrace{(IS_{21}, IS_{22}, IS_{23}, IS_{24})}_{I_2}, \ldots, \underbrace{(IS_{N1}, IS_{N2}, IS_{N3}, IS_{N4})}_{I_N}). \tag{2.1}$$

Генерално, пермутациската функција π се состои од три главни трансфомации $\mu, \nu, \sigma : \mathbb{Z}_{2^\omega}^4 \rightarrow \mathbb{Z}_{2^\omega}^4$, каде што $\mathbb{Z}_{2^\omega}$ е множеството од сите цели броеви помеѓу 0 и $2^\omega - 1$. Овие трансформации всушност ја вршат улогата на дифузија и нелинеарно мешање на влезот.

За реализација на π-функцијата, применети се следните операции:

- Собирање по модул 2^ω (операцијата ADD);
- Операција ротација во лево (циклично поместување во лево), $ROTL^\rho(X)$, каде што X е ω-бит збор и ρ е цел број за кој важи $0 \leqslant \rho < \omega$;
- XOR операција (логички оператор исклучиво ИЛИ) $\oplus$ над ω битни зборови.

Нека $\mathbf{X} = (X_0, X_1, X_2, X_3)$, $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$ и $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ се дадени четворки од ω битни зборови.

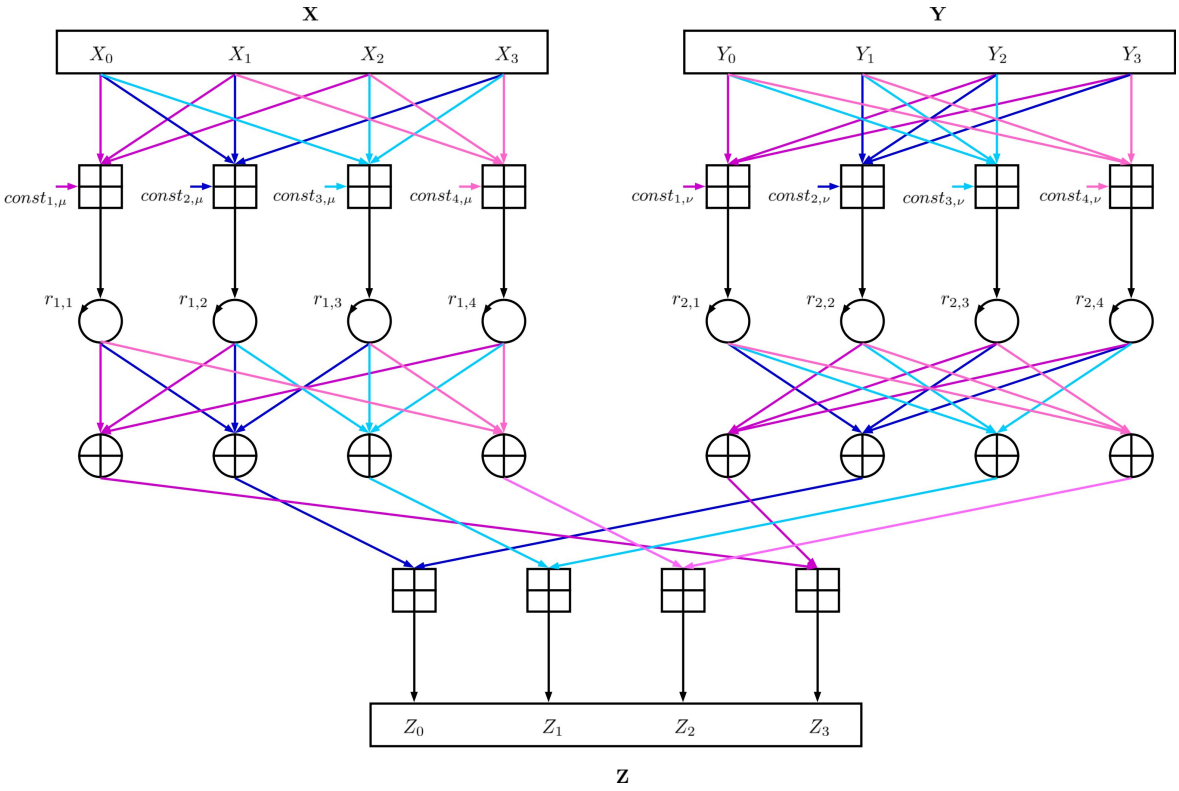
Понатака, нека со $*$ ја означиме следната операција:

$$\mathbf{Z} = \mathbf{X} * \mathbf{Y} \equiv \sigma(\mu(\mathbf{X}) \boxplus_4 \nu(\mathbf{Y})) \tag{2.2}$$

каде што $\boxplus_4$ е собирање по компоненти на два 4-димензионални вектори во полето $\left(\mathbb{Z}_{2^\omega}\right)^4$.

Алгоритамската дефиниција на операцијата $*$ над два 4-димензионални вектори $\mathbf{X}$ и $\mathbf{Y}$ за различни големини на зборовите е дадена во Табела 2.2, Табела 2.3 и Табела 2.4.

Исто така дадена е и графичка репрезентација на операцијата $*$ на Слика 2-7. Алгебарската дефиниција на оваа операција е посебно опишана во Глава 3.



Слика 2-7: Графичка репрезентација на ARX операцијата $*$

Да го земеме во предвид равенството (2.1) каде што внатрешната состојба е претставена како $IS = (I_1, I_2, \ldots, I_N)$. Една рунда од функцијата π се состои од две последователни итерации (од лево на десно и од десно на лево) над влезот

Операцијата $*$ за 16 битни зборови

Влез: $\mathbf{X} = (X_0, X_1, X_2, X_3)$ и $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$
каде што X_i и Y_i се 16 битни променливи.
Излез: $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ каде што Z_i се 16 битни променливи.
Привремени 16 битни променливи: $T_0, \ldots, T_{11}$.

μ–трансформација за $\mathbf{X}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^1(\texttt{0xF0E8} + X_0 + X_1 + X_2); \\ T_1 & \leftarrow & ROTL^4(\texttt{0xE4E2} + X_0 + X_1 + X_3); \\ T_2 & \leftarrow & ROTL^9(\texttt{0xE1D8} + X_0 + X_2 + X_3); \\ T_3 & \leftarrow & ROTL^{11}(\texttt{0xD4D2} + X_1 + X_2 + X_3); \end{array}$$

2.
$$\begin{array}{lcl} T_4 & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_5 & \leftarrow & T_0 \oplus T_1 \oplus T_2; \\ T_6 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_7 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \end{array}$$

ν–трансформација за $\mathbf{Y}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^2(\texttt{0xD1CC} + Y_0 + Y_2 + Y_3); \\ T_1 & \leftarrow & ROTL^5(\texttt{0xCAC9} + Y_1 + Y_2 + Y_3); \\ T_2 & \leftarrow & ROTL^7(\texttt{0xC6C5} + Y_0 + Y_1 + Y_2); \\ T_3 & \leftarrow & ROTL^{13}(\texttt{0xC3B8} + Y_0 + Y_1 + Y_3); \end{array}$$

2.
$$\begin{array}{lcl} T_8 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_9 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \\ T_{10} & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_{11} & \leftarrow & T_0 \oplus T_1 \oplus T_2; \end{array}$$

σ–трансформација за двете $\mu(\mathbf{X})$ и $\nu(\mathbf{Y})$:

1.
$$\begin{array}{lcl} Z_3 & \leftarrow & T_4 + T_8; \\ Z_0 & \leftarrow & T_5 + T_9; \\ Z_1 & \leftarrow & T_6 + T_{10}; \\ Z_2 & \leftarrow & T_7 + T_{11}; \end{array}$$

Табела 2.2: Алгоритамски опис на ARX операцијата $*$ за 16 битни зборови.

Операцијата $*$ за 32 битни зборови

Влез: $\mathbf{X} = (X_0, X_1, X_2, X_3)$ и $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$
каде што X_i и Y_i се 32 битни променливи.
Излез: $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ каде што Z_i се 32 битни променливи.
Привремени 32 битни променливи: $T_0, \ldots, T_{11}$.

μ–трансформација за $\mathbf{X}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^5(\texttt{0xF0E8E4E2} + X_0 + X_1 + X_2); \\ T_1 & \leftarrow & ROTL^{11}(\texttt{0xE1D8D4D2} + X_0 + X_1 + X_3); \\ T_2 & \leftarrow & ROTL^{17}(\texttt{0xD1CCCAC9} + X_0 + X_2 + X_3); \\ T_3 & \leftarrow & ROTL^{23}(\texttt{0xC6C5C3B8} + X_1 + X_2 + X_3); \end{array}$$

2.
$$\begin{array}{lcl} T_4 & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_5 & \leftarrow & T_0 \oplus T_1 \oplus T_2; \\ T_6 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_7 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \end{array}$$

ν–трансформација за $\mathbf{Y}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^3(\texttt{0xB4B2B1AC} + Y_0 + Y_2 + Y_3); \\ T_1 & \leftarrow & ROTL^{10}(\texttt{0xAAA9A6A5} + Y_1 + Y_2 + Y_3); \\ T_2 & \leftarrow & ROTL^{19}(\texttt{0xA39C9A99} + Y_0 + Y_1 + Y_2); \\ T_3 & \leftarrow & ROTL^{29}(\texttt{0x9695938E} + Y_0 + Y_1 + Y_3); \end{array}$$

2.
$$\begin{array}{lcl} T_8 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_9 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \\ T_{10} & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_{11} & \leftarrow & T_0 \oplus T_1 \oplus T_2; \end{array}$$

σ–трансформација за двете $\mu(\mathbf{X})$ и $\nu(\mathbf{Y})$:

1.
$$\begin{array}{lcl} Z_3 & \leftarrow & T_4 + T_8; \\ Z_0 & \leftarrow & T_5 + T_9; \\ Z_1 & \leftarrow & T_6 + T_{10}; \\ Z_2 & \leftarrow & T_7 + T_{11}; \end{array}$$

Табела 2.3: Алгоритамски опис на ARX операцијата $*$ за 32 битни зборови.

Операцијата $*$ за 64 битни зборови

Влез: $\mathbf{X} = (X_0, X_1, X_2, X_3)$ и $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$
каде што X_i и Y_i се 64 битни променливи.
Излез: $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ каде што Z_i се 64 битни променливи.
Привремени 64 битни променливи: $T_0, \ldots, T_{11}$.

μ–трансформација за $\mathbf{X}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^{7}(\texttt{0xF0E8E4E2E1D8D4D2} + X_0 + X_1 + X_2); \\ T_1 & \leftarrow & ROTL^{19}(\texttt{0xD1CCCAC9C6C5C3B8} + X_0 + X_1 + X_3); \\ T_2 & \leftarrow & ROTL^{31}(\texttt{0xB4B2B1ACAAA9A6A5} + X_0 + X_2 + X_3); \\ T_3 & \leftarrow & ROTL^{53}(\texttt{0xA39C9A999695938E} + X_1 + X_2 + X_3); \end{array}$$

2.
$$\begin{array}{lcl} T_4 & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_5 & \leftarrow & T_0 \oplus T_1 \oplus T_2; \\ T_6 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_7 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \end{array}$$

ν–трансформација за $\mathbf{Y}$:

1.
$$\begin{array}{lcl} T_0 & \leftarrow & ROTL^{11}(\texttt{0x8D8B87787472716C} + Y_0 + Y_2 + Y_3); \\ T_1 & \leftarrow & ROTL^{23}(\texttt{0x6A696665635C5A59} + Y_1 + Y_2 + Y_3); \\ T_2 & \leftarrow & ROTL^{37}(\texttt{0x5655534E4D4B473C} + Y_0 + Y_1 + Y_2); \\ T_3 & \leftarrow & ROTL^{59}(\texttt{0x3A393635332E2D2B} + Y_0 + Y_1 + Y_3); \end{array}$$

2.
$$\begin{array}{lcl} T_8 & \leftarrow & T_1 \oplus T_2 \oplus T_3; \\ T_9 & \leftarrow & T_0 \oplus T_2 \oplus T_3; \\ T_{10} & \leftarrow & T_0 \oplus T_1 \oplus T_3; \\ T_{11} & \leftarrow & T_0 \oplus T_1 \oplus T_2; \end{array}$$

σ–трансформација за двете $\mu(\mathbf{X})$ и $\nu(\mathbf{Y})$:

1.
$$\begin{array}{lcl} Z_3 & \leftarrow & T_4 + T_8; \\ Z_0 & \leftarrow & T_5 + T_9; \\ Z_1 & \leftarrow & T_6 + T_{10}; \\ Z_2 & \leftarrow & T_7 + T_{11}; \end{array}$$

Табела 2.4: Алгоритамски опис на ARX операцијата $*$ за 64–битни зборови.

со помош на две константни вредности.

Бројот на рунди R е променлив параметар. Кај π-Cipher v1.0 ние предложивме тој да биде 4, $R = 4$, додека пак во новата верзија π-Cipher v2.0 бројот на рунди го редуцираме на 3, $R = 3$.

Алгоритамската дефиниција на функцијата π е дадена во Табела 2.5.

Алгоритам: Една рунда на функцијата π
Влез: $I_1, \ldots, I_N, C_1$ и C_2, каде I_i се делчиња од влезниот стринг (4-торки од ω-бит зборови); C_1 и C_2 се рундовски константи (4-торки од ω-бит зборови). **Излез:** $J_1, \ldots, J_N$
1: $J_1 \leftarrow C_1 * I_1$ 2: **for** $i \leftarrow 1$ **to** N **do** 3: $\quad J_i \leftarrow J_{i-1} * I_i$ 4: **end for** 5: $J_N \leftarrow J_N * C_2$ 6: **for** $i \leftarrow (N-1)$ **downto** 1 **do** 7: $\quad J_i \leftarrow J_i * J_{i+1}$ 8: **end for**

Табела 2.5: Генерален алгоритам за една рунда на π-функцијата

Константите $C_1, C_2, \ldots, C_6$ кои се употребуваат во рундите на функцијата π се генерираат на ист начин како и константите употребени во операцијата $*$ и нивните вредности (во хексадецимална нотација) за различни големини на зборовите се дадени во Табела 2.6, Табела 2.7 и Табела 2.8.

2.3.1 За константите на фунцијата π

Со цел да се избегнат фиксни тривијални точки во пермутациската функција π, се одлучивме да искористиме константи во помошните пермутации μ и ν. Овие константи се избрани на тој начин што присуството на 1-ци и 0-ли во бинарната репрезентација на константите да биде балансирано. Користењето на ваквите

$$
\begin{aligned}
\mathbf{C}_1 &= \{\texttt{0xB4B2}, \texttt{0xB1AC}, \texttt{0xAAA9}, \texttt{0xA6A5}\} \\
\mathbf{C}_2 &= \{\texttt{0xA39C}, \texttt{0x9A99}, \texttt{0x9695}, \texttt{0x938E}\} \\
\mathbf{C}_3 &= \{\texttt{0x8D8B}, \texttt{0x8778}, \texttt{0x7472}, \texttt{0x716C}\} \\
\mathbf{C}_4 &= \{\texttt{0x6A69}, \texttt{0x6665}, \texttt{0x635C}, \texttt{0x5A59}\} \\
\mathbf{C}_5 &= \{\texttt{0x5655}, \texttt{0x534E}, \texttt{0x4D4B}, \texttt{0x473C}\} \\
\mathbf{C}_6 &= \{\texttt{0x3A39}, \texttt{0x3635}, \texttt{0x332E}, \texttt{0x2D2B}\}
\end{aligned}
$$

Табела 2.6: Константи за рундите за π16-Cipher

$$
\begin{aligned}
\mathbf{C}_1 &= \{\texttt{0x8D8B8778}, \texttt{0x7472716C}, \texttt{0x6A696665}, \texttt{0x635C5A59}\} \\
\mathbf{C}_2 &= \{\texttt{0x5655534E}, \texttt{0x4D4B473C}, \texttt{0x3A393635}, \texttt{0x332E2D2B}\} \\
\mathbf{C}_3 &= \{\texttt{0x271E1D1B}, \texttt{0x170FF0E8}, \texttt{0xE4E2E1D8}, \texttt{0xD4D2D1CC}\} \\
\mathbf{C}_4 &= \{\texttt{0xCAC9C6C5}, \texttt{0xC3B8B4B2}, \texttt{0xB1ACAAA9}, \texttt{0xA6A5A39C}\} \\
\mathbf{C}_5 &= \{\texttt{0x9A999695}, \texttt{0x938E8D8B}, \texttt{0x87787472}, \texttt{0x716C6A69}\} \\
\mathbf{C}_6 &= \{\texttt{0x6665635C}, \texttt{0x5A595655}, \texttt{0x534E4D4B}, \texttt{0x473C3A39}\}
\end{aligned}
$$

Табела 2.7: Константи за рундите за π32-Cipher

$$
\begin{aligned}
\mathbf{C}_1 &= \{\texttt{0x271E1D1B170FF0E8}, \texttt{0xE4E2E1D8D4D2D1CC}, \\
&\quad\ \texttt{0xCAC9C6C5C3B8B4B2}, \texttt{0xB1ACAAA9A6A5A39C}\} \\
\mathbf{C}_2 &= \{\texttt{0x9A999695938E8D8B}, \texttt{0x87787472716C6A69}, \\
&\quad\ \texttt{0x6665635C5A595655}, \texttt{0x534E4D4B473C3A39}\} \\
\mathbf{C}_3 &= \{\texttt{0x3635332E2D2B271E}, \texttt{0x1D1B170FF0E8E4E2}, \\
&\quad\ \texttt{0xE1D8D4D2D1CCCAC9}, \texttt{0xC6C5C3B8B4B2B1AC}\} \\
\mathbf{C}_4 &= \{\texttt{0xAAA9A6A5A39C9A99}, \texttt{0x9695938E8D8B8778}, \\
&\quad\ \texttt{0x7472716C6A696665}, \texttt{0x635C5A595655534E}\} \\
\mathbf{C}_5 &= \{\texttt{0x4D4B473C3A393635}, \texttt{0x332E2D2B271E1D1B}, \\
&\quad\ \texttt{0x170FF0E8E4E2E1D8}, \texttt{0xD4D2D1CCCAC9C6C5}\} \\
\mathbf{C}_6 &= \{\texttt{0xC3B8B4B2B1ACAAA9}, \texttt{0xA6A5A39C9A999695}, \\
&\quad\ \texttt{0x938E8D8B87787472}, \texttt{0x716C6A696665635C}\}
\end{aligned}
$$

Табела 2.8: Константи за рундите за π64-Cipher

константи допринесува да не може да се најде вредност X, за која ќе важи $\pi(X) = X$.

Големината и вредноста на константите зависи од тоа за која варијанта на π-Cipher се истите наменети. Најнапред го генериравме множеството *Constants* од сите возможни 8-битни вредности (1 бајт) кои имаат еднаква застапеност на 1-ци и 0-ли во нивната бинарна репрезентација. Вкупниот број на елементи во ова множество е 70 и истото е претставено со (2.3).

Константите за различните должини на зборовите во различните варијанти на π-Cipher се добиваат со спојување на последователни 8 битни елементи од множеството *Constants*.

$$\begin{aligned} Constants = \{&\texttt{0xF0}, \texttt{0xE8}, \texttt{0xE4}, \texttt{0xE2}, \texttt{0xE1}, \texttt{0xD8}, \texttt{0xD4}, \texttt{0xD2}, \texttt{0xD1}, \texttt{0xCC}, \\ &\texttt{0xCA}, \texttt{0xC9}, \texttt{0xC6}, \texttt{0xC5}, \texttt{0xC3}, \texttt{0xB8}, \texttt{0xB4}, \texttt{0xB2}, \texttt{0xB1}, \texttt{0xAC}, \\ &\texttt{0xAA}, \texttt{0xA9}, \texttt{0xA6}, \texttt{0xA5}, \texttt{0xA3}, \texttt{0x9C}, \texttt{0x9A}, \texttt{0x99}, \texttt{0x96}, \texttt{0x95}, \\ &\texttt{0x93}, \texttt{0x8E}, \texttt{0x8D}, \texttt{0x8B}, \texttt{0x87}, \texttt{0x78}, \texttt{0x74}, \texttt{0x72}, \texttt{0x71}, \texttt{0x6C}, \\ &\texttt{0x6A}, \texttt{0x69}, \texttt{0x66}, \texttt{0x65}, \texttt{0x63}, \texttt{0x5C}, \texttt{0x5A}, \texttt{0x59}, \texttt{0x56}, \texttt{0x55}, \\ &\texttt{0x53}, \texttt{0x4E}, \texttt{0x4D}, \texttt{0x4B}, \texttt{0x47}, \texttt{0x3C}, \texttt{0x3A}, \texttt{0x39}, \texttt{0x36}, \texttt{0x35}, \\ &\texttt{0x33}, \texttt{0x2E}, \texttt{0x2D}, \texttt{0x2B}, \texttt{0x27}, \texttt{0x1E}, \texttt{0x1D}, \texttt{0x1B}, \texttt{0x17}, \texttt{0x0F}\} \end{aligned} \tag{2.3}$$

π16-Cipher користи константи со должина од 16 бита. За да може да се реализира операцијата $*$ потребни се четири константи по 16 бита за трансформацијата μ и четири константи по 16-бита за трансформацијата ν, а додека пак за да се реализира функцијата π потреби се 6 константи за рундите, односно 6 четворки од по 16 бита или вкупно 64 последователни елементи од множеството *Constants*. Почнуваме од првиот елемент `0xF0` и земаме последователни 8 елементи, заклучно со елементот `0xD2` за да ги формираме константите на трансформацијата μ. Тоа се: $\{(\texttt{0xF0}, \texttt{0xE8}), (\texttt{0xE4}, \texttt{0xE2}), (\texttt{0xE1}, \texttt{0xD8}), (\texttt{0xD4}, \texttt{0xD2})\}$. На ист начин се креираат и константите за трансформацијата ν. Понатака се пристапува кон генерирање на константите за рундите и тоа по две константи за секоја рунда. Почнувајќи од елементот `0xB4` земаме последователни 8 елементи за да ја формираме четворката од 16 битни елементи на првата константа

$C_1 = \{(\texttt{0xB4}, \texttt{0xB2}), (\texttt{0xB1}, \texttt{0xAC}), (\texttt{0xAA}, \texttt{0xA9}), (\texttt{0xA6}, \texttt{0xA5})\}$. Потоа, земаме наредни 8 елементи (четворки од по 2 бајти) и ја формираме константата C_2 итн. се до генерирањето на последната константа C_6.

Со оглед на тоа што π32-Cipher користи 8 константи по 32-бита (4 бајти) за операцијата $*$ и 6 четворки од по 32-бита за константите за рундите, земаме 32 последователни елементи од множеството $Constants$ почнувајќи од `0xF0`. На овој начин ги генерираме константите за трансформациите μ и ν од операцијата $*$. Следните 96 последователни елементи се резервирани за константите на рундите на функцијата π. Поради тоа што ни требаат вкупно 128 последвателни елементи од множеството $Constants$, а тоа брои 70 елементи, како 71-ви елемент циклично го земаме првиот елемент од множеството.

π64-Cipher користи осум 64-битни константи за операцијата $*$. Тие се земаат како 64 последователни елементи од множеството на константи за иницијализација. За дефиниција на константите на рундите потребни се 6 четворки од по 64-бита (8 бајти), односно 192 последователни елементи кои циклично се земаат од множеството $Constants$ почнувајќи од елементот `0x27`.

Генерираните вредности за константите, како за операцијата $*$ така и за рундите на функцијата π за различните варијанти на π-Cipher се дадени соодветно во Секцијата 2.3.

Глава 3

Теорија на квазигрупи

Во оваа глава, ќе дадеме краток преглед кон математичките основи на теоријата на квазигрупи и како истите се искористени за дизајн на функцијата π.

3.1 Основни поими и дефиниции

Дефиниција 1. *Квазигрупа $(\mathcal{Q}, *)$ е алгебарска структура која што се состои од непразно множество $\mathcal{Q}$ и бинарна операција $* : \mathcal{Q}^2 \to \mathcal{Q}$ со својство дека секоја од равенките:*

$$\begin{aligned} a * x &= b \\ y * a &= b \end{aligned} \tag{3.1}$$

има единствени решенија x и y во множеството $\mathcal{Q}$.

Ред на квазигрупа $(\mathcal{Q}, *)$ е кардиналниот број $|\mathcal{Q}|$ на непразното множество $\mathcal{Q}$. Една квазигрупа $(\mathcal{Q}, *)$ велиме дека е конечна, ако нејзиниот ред е конечен број. Постои тесна поврзаност помеѓу конечните квазигрупи и Латинските квадрати односно, главниот дел од таблицата за множење (Келиева табела) со која може да се претстави една квазигрупа е Латински квадрат.

Дефиниција 2. *Латински квадрат е $n \times n$ табела исполнета со n различни*

симболи на таков начин што секој симбол се среќава точно еднаш во секој ред и секоја колона.

Едно важно својство на Латинските квадрати, а со тоа и на квазигрупите е ортогоналноста. За два Латински квадрати велиме дека се ортогонални ако со преклопување на двата квадрати секој добиен пар од симболи ќе се појавува точно еднаш во резултантниот квадрат. Концептот на ортогоналност алгебарски може да се опише како:

Дефиниција 3. *Две квазигрупи $(\mathcal{Q}, *)$ и $(\mathcal{Q}, \cdot)$, т.е. квазигрупи со операции $*$ и $\cdot$ дефинирани над истото множество $\mathcal{Q}$ се ортогонални ако системот равенки:*

$$\begin{aligned} x * y &= a \\ x \cdot y &= b \end{aligned} \tag{3.2}$$

има единствено решение за секој пар елементи $(a, b) \in \mathcal{Q}^2$.

Понатака во текстот ќе ја користиме нотацијата $(\mathcal{Q}, *) \perp (\mathcal{Q}, \cdot)$ за да претставиме ортогоналност на две квазигрупи. Јасно е дека $(\mathcal{Q}, *) \perp (\mathcal{Q}, \cdot) \Rightarrow (\mathcal{Q}, \cdot) \perp (\mathcal{Q}, *)$.

Друго својство кое го поседуваат квазигрупите е изотопизмот.

Дефиниција 4. *Нека $(\mathcal{Q}_1, *)$ и $(\mathcal{Q}_2, \cdot)$ се две квазигрупи. $(\mathcal{Q}_1, *)$ е изотопна со $(\mathcal{Q}_2, \cdot)$ ако постојат бијекции $(\alpha, \beta, \gamma) : \mathcal{Q}_1 \longrightarrow \mathcal{Q}_2$ такаа што $\gamma(x * y) = \alpha(x) \cdot \beta(y)$ за секои елементи $x, y \in \mathcal{Q}_1$.*

Подредената тројка (α, β, γ) се нарекува изотопизам. Во терминологијата на Латинските квадрати, изотопизмот (α, β, γ) е зададен со α - пермутација по редици, β - пермутација по колони и γ - пермутација на елементите од основното множество.

Познато е дека секоја квазигрупа $(\mathcal{Q}, *)$, има множество од пет квазигрупни операции наречени парастрофи. Множеството од парастрофи на квазигрупната операција $*$ е означено со $Par(*) = \{*, /, \backslash, \cdot, //, \backslash\backslash\}$. За секоја операција f, каде $f \in Par(*), (\mathcal{Q}, f)$, е исто така квазигрупа, позната како парастрофна на $(\mathcal{Q}, *)$ и

$Par(*) = Par(f)$, т.е.

$$Par(*) = Par(/) = Par(\backslash) = Par(\cdot) = Par(//) = Par(\backslash\backslash).$$

Петте парастрофи на квазигрупната операција $*$ се дефинирани во Табела 3.1.

Парастрофни операции					
(1)	$x/y = z$	$\iff$	$z * y = x$		
(2)	$x\backslash y = z$	$\iff$	$x * z = y$		
(3)	$x \cdot y = z$	$\iff$	$y * x = z$		
(4)	$x//y = z$	$\iff$	$y/x = z$	$\iff$	$z * x = y$
(5)	$x\backslash\backslash y = z$	$\iff$	$y\backslash x = z$	$\iff$	$y * z = x$

Табела 3.1: Парастрофи на квазигрупната операција $*$

Ова се основните дефиниции и својства за квазигрупите кои ќе ги користиме понатака во дефиницијата на функцијата π. Повеќе информации за квазигрупите може да се најдат во [24], [61], [93].

3.1.1 Алгебарска дефиниција на квазигрупи од ред 2^{64}, 2^{128} и 2^{256}

Алгебарската дефиниција на квазигрупите од соодветниот ред следува директно од алгебарската дефиниција на квазигрупите искористени во дизајнот на хеш функцијата Edon-R [36].

Нека $\mathcal{Q}_q = \{0,1\}^q$ е множество со кардинален број 2^q, каде што $q = 64, 128, 256$ и нека елементите од множеството $\mathbf{X} \in Q_q$ се претставени во бинарна репрезентација како q-бит зборови,

$$\mathbf{X} \equiv (\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_{q-2}, \widetilde{x}_{q-1}) \equiv \widetilde{x}_0 2^{q-1} + \widetilde{x}_1 2^{q-2} + \ldots + \widetilde{x}_{q-2} 2 + \widetilde{x}_{q-1},$$

каде што $\widetilde{x}_i \in \{0,1\}$.

Имајќи го во предвид својството за изотопизам на две квазигрупи, може да формираме квазигрупа $(\mathcal{Q}_q, *)$ изотопна на $(\left(\mathbb{Z}_{2^\omega}\right)^4, \boxplus_4)$, за $\omega = 16, 32, 64$ каде

што $\boxplus_4$ е операција на собирање по компоненти на два 4-димензионални вектори во множеството $\left(\mathbb{Z}_{2^\omega}\right)^4$. Според тоа може да дефинираме три пермутации $\mu, \nu, \sigma : \mathbb{Z}_2^q \longrightarrow \mathbb{Z}_2^q$ така што ќе важи:

$$\mathbf{X} * \mathbf{Y} = \sigma(\mu(\mathbf{X}) \boxplus_4 \nu(\mathbf{Y})), \tag{3.3}$$

за сите $\mathbf{X}, \mathbf{Y} \in \left(\mathbb{Z}_{2^\omega}\right)^4$.

Најнапред, треба да се дефинираат соодветните пермутации и да се докаже дека тие се бијекции, за даденото равенство (3.3) да биде валидно. Нашата интенција е да создадеме пермутација која што во својата основа ќе ги има како операции: собирање, ротација и логичкиот оператор исклучиво или, за да припаѓа кон класата на ARX-базирани функции. Како што наведовме во Секција 2.3, за реализација на функцијата π ги користиме следните операции:

- Собирање по модул 2^ω;
- Операција ротација во лево (циклично поместување во лево), $ROTL^\rho(X)$, каде што X е ω битен збор и ρ е цел број за кој важи $0 \leqslant \rho < \omega$;
- XOR е логичкиот оператор исклучиво ИЛИ, $\oplus$ применет над зборови со ω бита.

Понатака ја дефинираме следнава конвенција за елементите на множествата $\mathcal{Q}_q, q = 64, 128, 256$:

- Елементите $\mathbf{X} \in \mathcal{Q}_{64}$ се претставуваат како $\mathbf{X} = \{X_0, X_1, X_2, X_3\}$ каде што $X_{0,1,2,3}$ се 16 битни зборови,
- Елементите $\mathbf{X} \in \mathcal{Q}_{128}$ се претставуваат како $\mathbf{X} = \{X_0, X_1, X_2, X_3\}$ каде што $X_{0,1,2,3}$ се 32 битни зборови,
- Елементите $\mathbf{X} \in \mathcal{Q}_{256}$ се претставуваат како $\mathbf{X} = \{X_0, X_1, X_2, X_3\}$ каде што $X_{0,1,2,3}$ се 64 битни зборови.

Операцијата ротација на ω битен збор Y за ρ позиции во лево ќе ја обележуваме со $ROTL^\rho(Y)$. Да забележиме дека оваа операција е унарна и може да се

претстави како линеарно множење на матрица со вектор над прстенот $(\mathbb{Z}_2, +, \times)$, т.е. $ROTL^{\rho}(Y) = \mathbf{E}^{\rho} \cdot Y$, каде што $\mathbf{E}^{\rho} \in \mathbb{Z}_2^{\omega} \times \mathbb{Z}_2^{\omega}$ е матрица добиена од матрицата на идентитет со ротација на нејзините колони за ρ позиции во лево. Според ова, операцијата ротација во лево на елемент $\mathbf{X} \in \mathcal{Q}_q$, повторно може да се претстави со линеарно множење на матрица со вектор колона над прстенот $(\mathbb{Z}_2, +, \times)$, т.е. $ROTL^{\rho}(\mathbf{X}) = \mathbf{D}^{\rho} \cdot \mathbf{X}$, каде што $\mathbf{D}^{\rho} \in \mathbb{Z}_2^q \times \mathbb{Z}_2^q$, $\rho = (\rho_0, \rho_1, \rho_2, \rho_3) \in \{0, 1, \ldots \omega - 1\}^4$ е ротацискиот вектор и $\mathbf{X} = (X_0, X_1, X_2, X_3) \in \mathcal{Q}_q$. Матрицата $\mathbf{D}^{\rho}$ изгледа вака:

$$\mathbf{D}^{\rho} = \begin{bmatrix} \mathbf{E}^{\rho_0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}^{\rho_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E}^{\rho_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}^{\rho_3} \end{bmatrix},$$

каде што $\mathbf{E}^{\rho_i}$ се ротираните идентични матрици опишани претходно, а $\mathbf{0} \in \mathbb{Z}_2^{\omega} \times \mathbb{Z}_2^{\omega}$ се нулти матрици.

$\omega = 16$	$\omega = 32$
$\mathbf{C}_1 = \begin{bmatrix} \texttt{0xF0E8} \\ \texttt{0xE4E2} \\ \texttt{0xE1D8} \\ \texttt{0xD4D2} \end{bmatrix}, \mathbf{C}_2 = \begin{bmatrix} \texttt{0xD1CC} \\ \texttt{0xCAC9} \\ \texttt{0xC6C5} \\ \texttt{0xC3B8} \end{bmatrix}$	$\mathbf{C}_1 = \begin{bmatrix} \texttt{0xF0E8E4E2} \\ \texttt{0xE1D8D4D2} \\ \texttt{0xD1CCCAC9} \\ \texttt{0xC6C5C3B8} \end{bmatrix}, \mathbf{C}_2 = \begin{bmatrix} \texttt{0xB4B2B1AC} \\ \texttt{0xAAA9A6A5} \\ \texttt{0xA39C9A99} \\ \texttt{0x9695938E} \end{bmatrix}$

$\omega = 64$

$$\mathbf{C}_1 = \begin{bmatrix} \texttt{0xF0E8E4E2E1D8D4D2} \\ \texttt{0xD1CCCAC9C6C5C3B8} \\ \texttt{0xB4B2B1ACAAA9A6A5} \\ \texttt{0xA39C9A999695938E} \end{bmatrix}, \mathbf{C}_2 = \begin{bmatrix} \texttt{0x8D8B87787472716C} \\ \texttt{0x6A696665635C5A59} \\ \texttt{0x5655534E4D4B473C} \\ \texttt{0x3A393635332E2D2B} \end{bmatrix}$$

Табела 3.2: Вектор колони со константи $\mathbf{C}_1$ и $\mathbf{C}_2$ дадени во хексадецимална нотација за различни вредности на $\omega = 16, 32, 64$.

Операцијата собирање по модул, исто така, може да се претстави со помош на множење на матрици. За таа цел, користиме две бијективни пресликувања $\widetilde{\mathbb{A}}_1, \widetilde{\mathbb{A}}_3 : \left(\mathbb{Z}_{2^{\omega}}\right)^4 \longrightarrow \left(\mathbb{Z}_{2^{\omega}}\right)^4$ над прстенот $(\mathbb{Z}_{2^{\omega}}, +, \times)$, каде што $\omega = 16, 32, 64$ бита.

Пресликувањата $\widetilde{\mathbb{A}}_i, i = 1, 3$ се претставени како:

$$\widetilde{\mathbb{A}}_i(\mathbf{X}) = \mathbf{C}_j + \mathbb{A}_i \cdot \mathbf{X},$$

каде што $\mathbf{C}_j \in \left(\mathbb{Z}_{2^\omega}\right)^4, j = 1, 2$ се два вектори со константи кои за различните вредности на ω се дадени во Табела 3.2, а додека пак $\mathbb{A}_i, i = 1, 3$ се две 4×4 несингуларни матрици над прстенот $(\mathbb{Z}_{2^\omega}, +, \times)$. Елементите во матриците се состојат од 1–ци и 0–и. Поради фактот што овие матрици треба да бидат несингуларни, нивните елементи, се избрани од два Латински квадрати L_1 и L_2 од ред 4 и притоа за подобра дифузија на битовите истите се ортогонални. Имено, ги земаме горните Латински правоаголници $L_{1,1}$ и $L_{2,1}$ од двата Латински квадрати (како што е дадено во Табела 3.3). Земајќи ги колоните на овие Латински правоаголници како множества, всушност, конструираме два симетрични небалансирани блок дизајни (подетално за блок дизјани може да се најде во [90]). Овие блок дизајни се со параметри $(v, k, \lambda) = (4, 3, 2)$. Од нив конструираме матрици на соседство $A = (a_{ij})$ од ред v, според правилото:

$$a_{ij} = \begin{cases} 1 & \text{ако } x_j \in B_i, \\ 0 & \text{во спротивно,} \end{cases}$$

Во нашиот случај блоковите се дефинирани како $B_i = \{x_1, x_2, x_3\}$, каде $x_j \in \{0, 1, 2, 3\}$ и $i = \overline{1, 4}$.

Матрицата на соседство $\mathbb{A}_1$ која се добива од $L_{1,1}$ ја користиме за бијективно да ги трансформираме вредностите од операцијата собирање по модул 2^ω, работејќи во прстенот $(\mathbb{Z}_{2^\omega}, +, \times)$. На ист начин ја користиме и матрицата на соседство $\mathbb{A}_3$ добиена од $L_{2,1}$. Соодветните матрици на соседство се дадени во Табела 3.4.

За дефинирање на операцијата XOR (логички оператор - исклучиво или) за $\omega = 16, 32, 64$ битни зборови, повторно ќе искористиме две несингуларни матрици од ред $q \times q$ од прстенот $(\mathbb{Z}_2, +, \times)$, за $q = 64, 128, 256$. Дефинираме две линеарни бијективни пресликувања $\mathbb{A}_2, \mathbb{A}_4 : \left(\mathbb{Z}_{2^\omega}\right)^4 \longrightarrow \left(\mathbb{Z}_{2^\omega}\right)^4$ кои се претставени со две несингуларни матрици од ред $q \times q$ над прстенот $(\mathbb{Z}_2, +, \times)$. Повторно за

$$L_1 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ \hline 3 & 2 & 1 & 0 \end{array}\right] = \begin{bmatrix} L_{1,1} \\ L_{1,2} \end{bmatrix} \qquad L_2 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ \hline 1 & 0 & 3 & 2 \end{array}\right] = \begin{bmatrix} L_{2,1} \\ L_{2,2} \end{bmatrix}$$

Табела 3.3: Два ортогонални Латински квадрати кои се користат за дефинирање на операцијата собирање по модул $2^\omega, \omega = 16, 32, 64$.

$$\mathbb{A}_1 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbb{A}_3 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Табела 3.4: Матрици на соседство дефинирани од Латинските правоаголници $L_{1,1}$ и $L_{2,1}$ каде што елементите се земаат по колони

генерирање на овие матрици користиме Латински квадрати, овој пат Латински квадрат L_3 кој што е ортогонален со L_1 и L_2. Имено креираме два различни блок дизајни над истиот Латински квадрат. Земаме горен Латински правоаголник $L_{3,1}$ и долен Латински правоаголник $L'_{3,2}$ како што се прикажани во Табела 3.5. Од вака добиените Латински правоаголници ги формираме матриците на соседство $\mathbb{A}_2$ и $\mathbb{A}_4$ повторно со земање на елементите по колони. Соодветните матрици на соседство се дадени во Табела 3.6, каде што симболите $\mathbf{1}, \mathbf{0} \in \mathbb{Z}_2^\omega \times \mathbb{Z}_2^\omega$ означуваат $\mathbf{1}$ - идентични матрици, а $\mathbf{0}$ - нулти матрици.

Сега ќе дадеме формална дефиниција за пермутациите μ, ν и σ:

Дефиниција 5. *Трансформацијата* $\sigma : \mathcal{Q}_q \longrightarrow \mathcal{Q}_q \quad (q = 64, 128, 256)$ *се дефини-*

$$L_3 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \\ \hline 2 & 3 & 0 & 1 \end{array}\right] = \begin{bmatrix} L_{3,1} \\ L_{3,2} \end{bmatrix} \qquad L_3 = \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ \hline 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \end{array}\right] = \begin{bmatrix} L'_{3,1} \\ L'_{3,2} \end{bmatrix}$$

Табела 3.5: Латински квадрат поделен на различни блок дизајни за дефинирање на операцијата XOR на $\omega = 16, 32, 64$ битни зборови.

$$\mathbb{A}_2 = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix} \qquad \mathbb{A}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

Табела 3.6: Матрици на соседство дефинирани од Латинските правоаголници $L_{3,1}$ и $L'_{3,2}$ каде што елементите се земаат по колони

ра како:

$$\sigma(X_0, X_1, X_2, X_3) = (X_3, X_0, X_1, X_2).$$

Лема 1. *Трансформацијата σ е пермутација.*

доказ. Доказот следува директно од дефиницијата на трансформацијата. □

ω	q	ρ_μ	ρ_ν
16	64	(1, 4, 9, 11)	(2, 5, 7, 13)
32	128	(5, 11, 17, 23)	(3, 10, 19, 29)
64	256	(7, 19, 31, 53)	(11, 23, 37, 59)

Табела 3.7: Ротациските вектори кои се користат во пермутациите μ и ν за различни вредности на ω и q.

Дефиниција 6. *Трансформациите $\mu, \nu : \mathcal{Q}_q \longrightarrow \mathcal{Q}_q \quad (q = 64, 128, 256)$ се дефинирани како:*

$$\mu \equiv \widetilde{\mathbb{A}}_1 \circ ROTL^{\rho_\mu} \circ \mathbb{A}_2,$$
$$\nu \equiv \widetilde{\mathbb{A}}_3 \circ ROTL^{\rho_\nu} \circ \mathbb{A}_4$$

каде што ротациските вектори ρ_μ и ρ_ν се дадени во Табела 3.7 за сите вредности на $q = 64, 128, 256$, информациите за матрициите $\widetilde{\mathbb{A}}_1, \mathbb{A}_2, \widetilde{\mathbb{A}}_3$ и $\mathbb{A}_4$ се дадени соодветно во Табела 3.4 и Табела 3.6, каде симболите $\mathbf{1}, \mathbf{0} \in \mathbb{Z}_2^\omega \times \mathbb{Z}_2^\omega$ означуваат **1** *- идентични матрици, а* **0** *- нулти матрици.*

Лема 2. *Трансформациите μ и ν се пермутации над множеството $\mathcal{Q}_q$ каде што $q = 64, 128, 256$.*

доказ. За да докажеме дека μ и ν се пермутации над множеството $\mathcal{Q}_q$, треба да докажеме дека истите се бијекции. Според дефиницијата на трансформациите, истите се претставени преку несингуларни матрици $\mathbb{A}_1$ и $\mathbb{A}_3$ во прстенот $(\mathbb{Z}_{2^\omega}, +, \times)$, за $\omega = 16, 32, 64$ и несингуларни матрици $\mathbf{D}^\rho, \mathbb{A}_2$ и $\mathbb{A}_4$ во прстенот $(\mathbb{Z}_2, +, \times)$. □

Теорема 1. *Операцијата* $* : \mathcal{Q}_q^2 \longrightarrow \mathcal{Q}_q$ *дефинирана како:*

$$\mathbf{X} * \mathbf{Y} = \sigma(\mu(\mathbf{X}) \boxplus_4 \nu(\mathbf{Y})),$$

е некомутативна квазигрупна операција за $q = 64, 128, 256$.

доказ. Нека $q = 64$. Најнапред ќе дадеме доказ за случајот кога $q = 64$, односно должината на зборовите е $\omega = 16$. За да докажеме дека $*$ е квазигрупна операција, треба да докажеме дека $(\mathcal{Q}_{64}, *)$ не е лупа, односно не постои неутрален елемент e, таков што $x * e = x$ и $e * x = x$, за сите $x \in \mathcal{Q}_{64}$. Нека $\mathbf{E} \in \mathcal{Q}_{64}$ е неутрален елемент и притоа нека

$$\mu(\mathbf{E}) \boxminus_4 \nu(\mathbf{E}) = \mathbf{Const},$$

каде што $\mathbf{Const} \in \mathcal{Q}_{64}$ и $\boxminus_4$ е инверзна операција на операцијата $\boxplus_4$ и означува одземање по модул 2^{16} на елементите по компоненти. Доколку неутралниот елемент го замениме во дефиницијата на квазигрупната операција ќе добиеме:

$$\mathbf{E} * \mathbf{X} = \sigma(\mu(\mathbf{E}) \boxplus_4 \nu(\mathbf{X})) = \sigma(\nu(\mathbf{X}) \boxplus_4 \mu(\mathbf{E})).$$

Поради тоа што веќе е докажано дека σ е пермутација, може слободно да ја елемининараме од последното равенство и да добиеме:

$$\mu(\mathbf{E}) \boxplus_4 \nu(\mathbf{X}) = \nu(\mathbf{X}) \boxplus_4 \mu(\mathbf{E}),$$

односно

$$\mu(\mathbf{E}) \boxminus_4 \nu(\mathbf{E}) = \mu(\mathbf{X}) \boxminus_4 \nu(\mathbf{X}) = \mathbf{Const}.$$

Од последното равенство доаѓаме до заклучок дека за секој елемент $\mathbf{X} \in \mathcal{Q}_{64}$ из-

разот $\mu(\mathbf{X}) \boxminus_4 \nu(\mathbf{X})$ секогаш резултира со константна вредност, што е спротивно од дефиницијата на трансформациите. Со тоа докажуваме дека не постои неутрален елемент за квазигрупата $(\mathcal{Q}_{64}, *)$, односно таа не е лупа. Значи операцијата $*$ е некомутативна квазигрупна операција. Соодветно истиот доказ следува и во случаите кога $q = 128$ и $q = 256$. □

3.2 Дефиниција на функцијата π преку квазигрупна стринг трансформација

Квазигрупните стринг трансформации се истражувани во повеќе трудови [65], [66] и притоа нивната практична примена е прикажана во неколку докажани симетрични криптографски примитиви како што се проточниот шифрувач Edon80 [33] и хеш функцијата Edon-R [36]. Квазигрупната стринг трансформација со помош на квазигрупна операција $*$ врши бијективно пресликување на дадена влезна низа од елементи со должина q во друга резултантна низа од елементи со иста должина q.

За да ја дефинираме функцијата π ние ќе ја искористиме елементарната квазигрупна стринг трансформација, E-трансформација која што за прв пат е дефинирана во трудот [65].

Во продолжение ќе ги дадеме дефинициите на E-трансформациите.

Дефиниција 7. *Функцијата* $E_1 : \mathcal{Q}_q^{N+1} \to \mathcal{Q}_q^N$ *е дефинирана на следниот начин:*

$$\begin{aligned} E_1(\boldsymbol{C}, \boldsymbol{I}_1, \ldots, \boldsymbol{I}_N) = (\boldsymbol{J}_1, \ldots, \boldsymbol{J}_N), \quad & \textit{и притоа важи} \\ \boldsymbol{J}_1 &= \boldsymbol{C} * \boldsymbol{I}_1, \\ \boldsymbol{J}_i &= \boldsymbol{J}_{i-1} * \boldsymbol{I}_i, \ i = 2, \ldots, N \end{aligned} \tag{3.4}$$

каде што $\boldsymbol{C} \in \mathcal{Q}_q$ *е вектор од* $16, 32$ *или* 64 *битни константни вредности, а соодветно* $\mathbf{I} = (\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N), \boldsymbol{J} = (\boldsymbol{J}_1, \ldots, \boldsymbol{J}_N) \in \mathcal{Q}_q^N$ *се влез и излез на функцијата.*

Дефиниција 8. *Функцијата* $E_2 : \mathcal{Q}_q^{N+1} \to \mathcal{Q}_q^N$ *е дефинирана на следниот начин:*

$$\begin{aligned} E_2(\boldsymbol{C}, \boldsymbol{I}_1, \ldots, \boldsymbol{I}_N) &= (\boldsymbol{J}_1, \ldots, \boldsymbol{J}_N), \quad \textit{и притоа важи} \\ \boldsymbol{J}_N &= \boldsymbol{I}_N * \boldsymbol{C}, \\ \boldsymbol{J}_{N-i} &= \boldsymbol{I}_{N-i} * \boldsymbol{J}_{N-i+1}, \ i = 1, \ldots, N-1 \end{aligned} \tag{3.5}$$

каде што $\boldsymbol{C} \in \mathcal{Q}_q$ *е вектор од* $16, 32$ *или* 64 *битни константни вредности, а соодветно* $\mathbf{I} = (\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N), \mathbf{J} = (\boldsymbol{J}_1, \ldots, \boldsymbol{J}_N) \in \mathcal{Q}_q^N$ *се влез и излез на функцијата.*

Со композиција на функциите E_1 и E_2 се добива нова бијективна функција π_R (наречена рундовска функција) и ја претставуваме како:

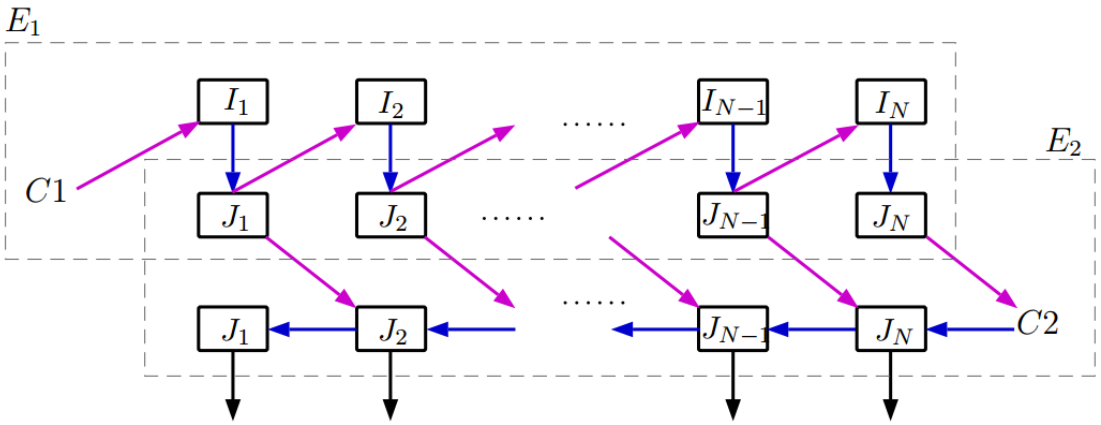
$$\pi_R(\mathbf{I}_1, \ldots, \mathbf{I}_N) = (\mathbf{J}_1, \ldots \mathbf{J}_N) = E_2(\mathbf{C}_2, E_1(\mathbf{C}_1, \mathbf{I}_1, \ldots, \mathbf{I}_N)). \tag{3.6}$$

каде со нејзино повторување се добива пермутациската функција π.

Дефиниција 9. *За дадена квазигрупа* $(\mathcal{Q}_q, *), q = 64, 128, 256$ *функцијата* π *е пермутација на елементите од множеството* $\mathcal{Q}_q$ *и дефинирана на следниов начин:*

$$\begin{aligned} &\pi(\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N) = (\boldsymbol{J}_1, \ldots, \boldsymbol{J}_N), \textit{каде што бројот на рунди е 3 и притоа,} \\ &\pi(\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N) = \pi_R(\pi_R(\pi_R(\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N))) = \\ &E_2(\boldsymbol{C}_6, E_1(\boldsymbol{C}_5, E_2(\boldsymbol{C}_4, E_1(\boldsymbol{C}_3, E_2(\boldsymbol{C}_2, E_1(\boldsymbol{C}_1, \boldsymbol{I}_1, \ldots, \boldsymbol{I}_N)))))) \end{aligned}$$

така што функциите E_1 *и* E_2 *се дефинирани соодветно со Дефиниција 7 и 8, а додека пак* $\boldsymbol{C}_1, \ldots \boldsymbol{C}_6$ *се вектори со* $16, 32$ *или* 64 *битни константни вредности дадени во Табела 2.6, Табела 2.7 и Табела 2.8 во Секција 2.3.*



Слика 3-1: Графичка репрезентација на една рунда од функцијата π.

Најдобара претстава за функцијата π, може да се добие од графичкиот приказ кој што е даден на Слика 3-1, каде што е дадена само една рунда на функцијата. На Сликата 3-1 со дијагоналните стрелки се интерпретира операцијата $*$ помеѓу почетокот и крајот на стрелката, а вертикалните стрелки се поистоветуваат со симболот $=$.

Глава 4

Сигурносна анализа

Во оваа глава ќе дадеме осврт кон сигурносната анализа на π-Cipher. Во таа смисла најнапред ќе дадеме комплетен теоретски доказ за сигурноста на дизајнот на π-Cipher. Поради тоа што во сите докази за теоретска сигурност се тргнува од фактот дека основната крипто примитива се подразбира дека е идеално сигурна и се оди кон докажување на дизајнот на алгоритамот, ние, сигурноста на нашата крипто примитива, пермутацискатa функција π, ќе ја поткрепиме со нумерички податоци.

4.1 Докажлива сигурност

Докажливата сигурност се оденсува на криптографска парадигма која што дозволува ригорозна анализа и проценка на сигурноста на криптографските шеми и протоколи. Се состои од два главни концепти. Првиот е формулација на дефинициите за сигурност. Овој концепт ја специфицира целта на криптографската шема што се стреми да ја достигне, како што се доверливост и автентичност на податоците и тоа преку точно дефинирани математички изрази. Дефиницијата за сигурност нормално се изразува како игра кој се игра помеѓу напаѓачот и криптографската шема. Обично напаѓачот е параметризиран алгоритам со голема пресметковна моќ и притоа треба да се најде веројатноста со која тој би победил во играта. Сигурноста на шемата се изразува како најголемата веројат-

ност за победа од некоја класа од напаѓачи. Според тоа ќе речеме дека шемата е "пробиена"ако постои изводлив напаѓач чија што веројатност за победа го надминува (е под) некој однапред дефиниран праг.

Вториот концепт е техниката за докажување со помош на редукција. Всушност со оваа техника се трансформира алгоритам кој што решава проблем B во алгоритам кој што решава проблем A и нормално таа трансформација треба да биде ефиксно пресметана. Притоа ако ваква трансформација постои, велиме дека проблемот B е редуциран до проблем A кој што најчесто е познат. Нека имаме шема која што е базирана на некој блок шифрувач. За да ја докажеме сигурноста на оваа шема, конструираме редукција со која што го трансформираме секој напаѓач кој што се обидува да ја разбие шемата на напаѓач кој што се обидува да го разбие блок шифрувачот. Со претходна претпоставка дека блок шифрувачот е идеално сигурен се докажува дека и шемата е идеална во секој случај.

4.2 Формална дефиниција на π-Cipher

Натпреварот CAESAR за автентикациска енкрипција во своите побарувања постави нови услови за користење на нонс. Наспроти досегашната пракса, нонсот да биде единствен број кој што е јавно познат, сега нонсот има можност да биде пар од броеви $N = (PMN, SMN)$, каде што PMN е јавниот дел, додека пак SMN е таен дел. Притоа јавниот дел од нонсот работи на ист принцип како и до сега, новината е во тајниот дел SMN кој сега не е јавно познат и треба да се енкриптира и автентицира, а после тоа истиот да се декриптира. Во натпреварот CAESAR користењето на тајниот дел од нонсот беше опционално и притоа само два кандидати успеаа да го вклопат во нивниот дизајн: π-Cipher [34] и ICEPOLE [83].

Прецизна формализација на новата варијанта на шифрувачи за автентикациска енкрипција кои користат нонс како $N = (PMN, SMN)$ покрај другите стандардни аргументи како клуч, поврзани податоци и порака е дадена во трудот [74].

Дефиниција 10 (AEAD со SMN [74]). *Автентикациска енкрипција со поврзани*

податоци (AEAD) и нонс како таен број (SMN) е тројката $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *каде што* $\mathcal{E}$ *е детерминистички енкриптирачки алгоритам кој што прима клуч* $K \in \mathcal{K} = \{0,1\}^k$*, јавен дел од нонсот* $P \in \mathcal{P} = \{0,1\}^p$*, поврзани податоци* $A \in \mathcal{A} = \{0,1\}^*$*, нонс како таен број* $S \in \mathcal{S} = \{0,1\}^s$ *и оригинална порака* $M \in \mathcal{M} = \{0,1\}^*$*за на крај да врати стринг* $\mathcal{C} = \mathcal{E}_K^{P,A}(S, M)$*. Декрипциското алгоритам* $\mathcal{D}$ *ги прима како влезни аргументи стринговите* K, P, A *и* $\mathscr{C} \subseteq \{0,1\}^*$ *и враќа пар од стрингови во* $(\mathcal{S}, \mathcal{M})$ *или неважечки симбол* $\perp$*, т.е.* $\mathcal{D}_K^{P,A}(\mathscr{C}) = (S, M)$ *или* $\mathcal{D}_K^{P,A}(\mathscr{C}) = \perp$*. Побаруваме* $\mathcal{D}_K^{P,A}(\mathcal{E}_K^{P,A}(S, M)) = (S, M)$ *за сите* $K \in \mathcal{K}$*,* $P \in \mathcal{P}$*,* $A \in \mathcal{A}$*,* $S \in \mathcal{S}$ *и* $M \in \mathcal{M}$*. Значи,* $|\mathcal{E}_K^{P,A}(S, M)| = l(|M|)$*, каде* l *е некоја линеарна функција од должината на пораката.*

Функциите за енкрипција и декрипција може да се претстават на следниот начин:

$$\mathcal{E} : \mathcal{K} \times \mathcal{P} \times \mathcal{A} \times \mathcal{S} \times \mathcal{M} \to \{0,1\}^*,$$
$$\mathcal{D} : \mathcal{K} \times \mathcal{P} \times \mathcal{A} \times \{0,1\}^* \to (\mathcal{S} \times \mathcal{M}) \cup \perp .$$

π-Cipher $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е AEAD шема со таен нонс SMN и соодветствува со Дефиницијата 10. Овој шифрувач се потпира на пермутациската функција $\pi : \{0,1\}^b \to \{0,1\}^b$ и соодветно: клуч K, јавен дел од нонсот PMN, поврзани податоци AD, таен дел од нонсот SMN, порака M и шифриран текст C. π-Cipher како што е дизајниран може да работи во различни модови во зависност од намената. Својства кои го карактеризираат π-Cipher е тоа што тој е онлајн и паралелен шифрувач, односно енкрипцијата на блокот M_i не зависи од блоковите кои следуваат после него ниту пред него и плус сите овие блокови може да се извршуваат независно сами за себе. Исто така тој може да работи и во мод со меѓу тагови. Ваквата конструкција и сигурносен модел на π-Cipher се дадени понатака во текстот.

4.3 Сигурносен модел

Нека $Perm(n)$ за $n \in \mathbb{N}$ го означува множеството на сите пермутации од n бита. Кога пишуваме $x \xleftarrow{\$} \mathcal{X}$ за некое конечно множество $\mathcal{X}$, мислиме на тоа

дека x ќе претставува примерок кој што е униформно случаен од $\mathcal{X}$. За $x \in \{0,1\}^n$ и $a, b \leqslant n$ означуваме со $[x]^a$ и $[x]_b$, a најлевите и b најдесните битови од x соодветно. За паровите $(j,k),(j',k')$ користиме лексикографско подредување, односно $(j,k) > (j',k')$ значи дека $j > j'$ или $j = j'$ и $k > k'$.

Нека Π е автентикациско енкрипциска шема дефинирана во Дефиниција 10. Како дополнување ние ќе дфинираме \$ да биде идеална верзија од алгоритамот $\mathcal{E}$ со клуч K, каде што \$ враќа $C \xleftarrow{\$} \{0,1\}^{|M|+|S|+\tau}$ за влез од нови вредности за (P,A,S,M).

Поради тоа што одиме кон докажување на сигурноста на шемата, земаме дека основната примитива, пермутацијата π е земена како рамномерно случајна од $Perm(b)$, каде што b е параметар што означува должина на состојба на пермутацискатa функција и е одреден од соодветната шема, овој модел се нарекува идеален модел. Исто така направивме модел со две игри, каде и двете започнуваат со избирање на случаен клуч K од просторот на клучеви $\mathcal{K}$. Во реалната игра, пророк $\mathcal{O}$ поставува прашање на $\mathcal{E}$ и истото се одговара со реални вредности. Од друга страна пак, во идеалната игра, пророкот поставува прашање на \$ и истото се одговара со стринг во кој битовите се случајни и рамномерно распределени. Главната улога во овие игри ја игра напаѓачот $\mathcal{A}$, што всушност претставува веројатносен алгоритам и кој има пристап до некои од пророците $\mathcal{O}$. Со $\mathcal{A}^{\mathcal{O}} \Rightarrow 1$ го означуваме настанот каде напаѓачот $\mathcal{A}$ после интеракцијата со пророкот $\mathcal{O}$, дава на излез 1. Уште повеќе, напаѓачот има неограничена пресметковна моќ и неговата комплексност се мери во број на прашања поставени до пророкот.

Понатака ќе ги дадеме формалните дефиниции за сигурноста на шемата кои се соодветно изведени од [12], [52].

Приватност

Нека p означува листа од случајни пермутации на шемата Π. Ја дефинираме предноста на напаѓачот $\mathcal{A}$ во разбивање на приватноста на шемата Π според следното:

$$\mathbf{Adv}^{priv}_{\Pi}(\mathcal{A}) = |\mathbf{Pr}(\mathcal{A}^{p^{\pm},\mathcal{E}_K} \Rightarrow 1) - \mathbf{Pr}(\mathcal{A}^{p^{\pm},\$} \Rightarrow 1)|,$$

каде што веројатностите се земени со случаен избор на $p, \$, K$ и случаен избор на $\mathcal{A}$. Фактот што напаѓачот има пристап и до нормалните и до инверзните пермутации во p го означуваме со $p^{\pm}$. Претпоставуваме дека напаѓачот не ги повторува вредностите за нонсот, односно никогаш не поставува две прашања до $\mathcal{E}$ или \$ користејќи ист нонс. Во овој случај станува збор за нонсот $N = (P, S)$. Со $\mathbf{Adv}^{priv}_{\Pi}(q_p, q_{\mathcal{E}}, \lambda_{\mathcal{E}})$, ја означуваме максималната предност земена од сите напаѓачи кои прашуваат на $p^{\pm}$ најмногу q_p пати и кои поставуваат најмногу $q_{\mathcal{E}}$ прашања со вкупна должина од најмногу $\lambda_{\mathcal{E}}$ блокови на $\mathcal{E}$ и \$. Овој тип на дефиниција на приватност е познат како IND-CPA сигурност на автентикациско енкрипциска шема, каде шифрираниот текст не смее да се разликува од енкрипцијата на случајни битови.

Интегритет

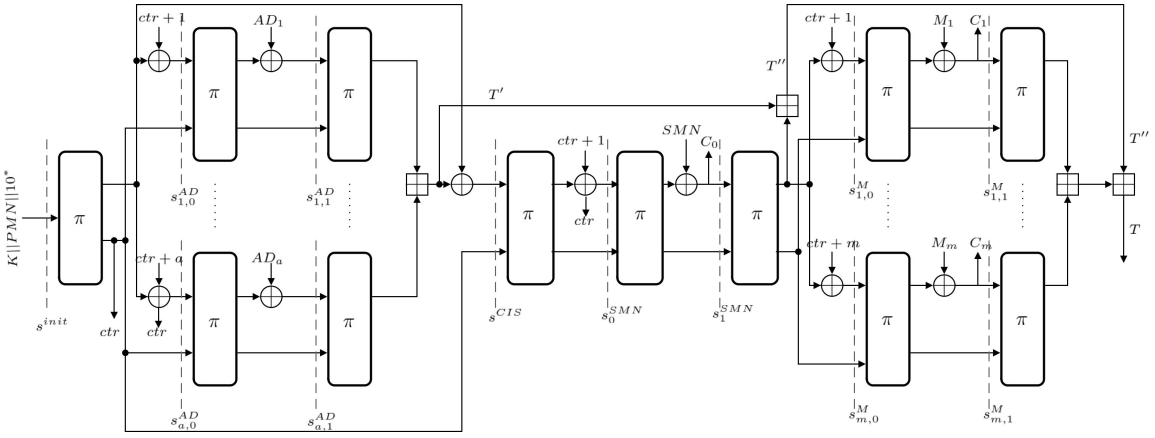
Исто како и претходно, нека p означува листа од случајни пермутации на шемата Π. Ја дефинираме предноста на напаѓачот $\mathcal{A}$ во разбивање на интегритетот на шемата Π според следното:

$$\mathbf{Adv}^{auth}_{\Pi}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{p^{\pm}, \mathcal{E}_K, \mathcal{D}_K} \text{ forges}),$$

каде што веројатностите се земени со случаен избор на p, K и случаен избор на $\mathcal{A}$. Овде велиме дека "$\mathcal{A}$ forges" ако $\mathcal{D}_K$ врати валидна порака (не симболот $\perp$), кога на влез прима (P, A, C) и кога C никогаш не е добиен од страна на $\mathcal{E}_K$ при влезно прашање (P, A, S, M) за некоја порака M. Наспроти фактот што напаѓачот не смее да го повторува нонсот при $\mathcal{E}_K$, дозволено му е да го повторува нонсот при декрипциските прашања и тоа само јавниот дел P. Со $\mathbf{Adv}^{auth}_{\Pi}(q_p, q_{\mathcal{E}}, \lambda_{\mathcal{E}}, q_{\mathcal{D}}, \lambda_{\mathcal{D}})$, ја означуваме максималната предност земена од сите напаѓачи кои прашуваат на $p^{\pm}$ најмногу q_p пати и кои поставуваат најмногу $q_{\mathcal{E}}$ прашања со вкупна должина од најмногу $\lambda_{\mathcal{E}}$ блокови на $\mathcal{E}$ и најмногу $q_{\mathcal{D}}$ прашања со вкупна должина од најмногу $\lambda_{\mathcal{D}}$ блокови на $\mathcal{D}$. Овој тип на дефиниција на интегритет е познат како INT-CTXT сигурност на автентикациско енкрипциска шема, каде што е обезбеден интегритетот на шифрираниот текст.

Модел на π-Cipher

За реализација на доказите во Секција 4.4 и 4.5, земаме напаѓач кој што поставува q_p пермутациски прашања и $q_{\mathcal{E}}$ енкрипциски прашања со вкупна должина од $\lambda_{\mathcal{E}}$ блокови. Во доказот за интегритет (Секција 4.5) напаѓачот има можност дополнително да поставува $q_{\mathcal{D}}$ декрипциски прашања со вкупна должина од $\lambda_{\mathcal{D}}$ блокови. Земаме дека прашањата до $\mathcal{E}$ се состојат од a блокови на поврзани податоци и m блокови на порака.



Слика 4-1: Шема на π-Cipher со означени состојби s

Според дизајнот на π-Cipher дефинираме вредност на состојба s, која што всушност ќе ни претставува влез во точно еден повик на пермутациската функција π. Во зависност на која пермутациска функција припаѓа соодветно оваа променлива ќе добива нови ознаки за полесен преглед понатака во текстот. Шемата на π-Cipher со соодветно означени состојби s е дадена на Слика 4-1. Да забележиме дека s^{init} е вредноста на состојбата при првиот повик на пермутациската функција π во фазата на иницијализација. После тоа следуваат a паралелни гранки за обработка на блоковите на поврзаните податоци во фазата на AD. Ги обележуваме нив со $s^{AD}_{l,0}$ - состојбата пред првиот повик на функцијата π и $s^{AD}_{l,1}$-состјбата пред вториот повик на функцијата π во истата гранка l каде $l = \overline{1,a}$. Со s^{CIS} ја обележуваме вредноста на состојбата за ажурирање на заедничката внатрешна состојба CIS, која се реализира на крај на втората фаза по обработка на поврзаните податоци. Следни вредности за состојбата се во фазата на

SMN, за привот повик s_0^{SMN} и вториот повик на функцијата π, s_1^{SMN}. На крај во последната фаза, вредностите на состојбите се распределени на ист начин како и во фазата на поврзани податоци, само сега што имаме m паралелни гранки. Значи, вредностите на состојбите се $s_{l,0}^M$ и $s_{l,1}^M$, за $l = \overline{1,m}$. Според ова соодветните вредности за состојбата s можеме да ги претставиме како:

$$\left(s^{init};\quad \begin{bmatrix} s_{1,0}^{AD} & s_{1,1}^{AD} \\ \vdots & \vdots \\ s_{a,0}^{AD} & s_{a,1}^{AD} \end{bmatrix};\quad s^{CIS};\quad s_0^{SMN};\quad s_1^{SMN};\quad \begin{bmatrix} s_{1,0}^{M} & s_{1,1}^{M} \\ \vdots & \vdots \\ s_{m,0}^{M} & s_{m,1}^{M} \end{bmatrix} \right).$$

Овде, ако j-то енкрипциско прашање е со должина $a+m$ блокови, тогаш бројот на состојби, означен со $\sigma_{\mathcal{E},j}$ е $1 + 2a + 1 + 2 + 2m$. Според тоа, вкупниот број на сите состојби $\sigma_{\mathcal{E}}$, евалуации на функцијата π преку енкрипциско прашање, е:

$$\sigma_{\mathcal{E}} := \sum_{j=1}^{q_{\mathcal{E}}} \sigma_{\mathcal{E},j} \leqslant q_{\mathcal{E}}(2a + 2m + 4) = 2\lambda_{\mathcal{E}} + 4q_{\mathcal{E}}. \tag{4.1}$$

За декрипциските прашања на $\mathcal{D}$ се е аналогно и притоа вкупниот број на сите состојби, евалуации на функцијата π преку декрипциско прашање, се означува со $\sigma_{\mathcal{D}} := \sum_{j=1}^{q_{\mathcal{D}}} \sigma_{\mathcal{D},j}$.

4.4 Приватност на π-Cipher

Теорема 2. *Нека* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *е предложената шема* π*-Cipher, каде што основната криптографска примитива е заменета како идеална примитива* p *која што оперира над* b *битови. Значи,*

$$\boldsymbol{Adv}_{\Pi}^{priv}(q_p, q_{\mathcal{E}}, \lambda_{\mathcal{E}}) \leqslant \frac{2(q_p + \sigma_{\mathcal{E}})^2}{2^b} + \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{q_p r}{2^c} + \frac{q_{\mathcal{E}} a}{2^{b/2}} + \sqrt{\frac{8e\sigma_{\mathcal{E}} q_p}{2^b}},$$

каде што $\sigma_{\mathcal{E}}$ *е дефинирана во (4.1),* q_p *е максималниот број на прашања до идеалната пермутација* $p^{\pm}$*,* $q_{\mathcal{E}}$ *е максималниот број на енкрипциски прашања со вкупна должина од* $\lambda_{\mathcal{E}}$ *блокови и* a *е максималниот број на блокови за поврзани податоци.*

доказ. Земаме било кој напаѓач $\mathcal{A}$, кој што има пристап до $(p^{\pm}, \mathcal{E}_K)$ или $(p^{\pm}, \$)$ и чија цел е да направи разлика помеѓу нив. За појасно, пишуваме:

$$\mathbf{Adv}_{\Pi}^{priv}(\mathcal{A}) = |\mathbf{Pr}(\mathcal{A}^{p^{\pm}, \mathcal{E}_K} \Rightarrow 1) - \mathbf{Pr}(\mathcal{A}^{p^{\pm}, \$} \Rightarrow 1)| = \Delta_{\mathcal{A}}(p^{\pm}, \mathcal{E}_K; p^{\pm}, \$). \qquad (4.2)$$

Најнапред ќе ја замениме пермутациската функција p со двонасочна функција $f : \{0,1\}^b \longrightarrow \{0,1\}^b$ и со користење на PRP/PRF Switching Lema [89] добиваме:

$$\Delta_{\mathcal{A}}(p^{\pm}, \mathcal{E}_K; p^{\pm}, \$) - \Delta_{\mathcal{A}}(f^{\pm}, \mathcal{E}_K; f^{\pm}, \$) \leqslant \frac{(q_p + \sigma_{\varepsilon})(q_p + \sigma_{\varepsilon} - 1)}{2^{b+1}} \leqslant \frac{(q_p + \sigma_{\varepsilon})^2}{2^b}. \quad (4.3)$$

Од сега па натака, ние ќе го рестриктираме нашето внимание кон напаѓач кој има пристап до пророкот $(f^{\pm}, \{\mathcal{E}_K, \$\})$. За фунцкијата $f^{\pm}$ постои листа $\mathscr{F}$ од сите парови (x, y) од прашања/одговори. На почеток оваа листа е празна. За прашања кон функцијата f, $f(x)$, ако парот $(x, y) \in \mathscr{F}$, соодветната вредност $y = f(x)$ се враќа. За ново прашање $f(x)$, а и за y случајно избран од множеството $\{0,1\}^b$, ако (x, y) како пар постои во листата $\mathscr{F}$ тогаш функцијата се прекинува, инаку парот (x, y) се додава на листата. Објаснувањето за прашања кон инверзната функција f^{-1}, $f^{-1}(y)$ се слични. Сега, $p^{\pm}$ и $f^{\pm}$ се однесуваат идентично се додека последната не се прекине и притоа напаѓачот може да направи максимум $q_p + \sigma_{\mathcal{E}}$ евалуации кон функцијата f со веројатност која што е дадена со неравенството

4.3.

За реализација на овој доказ користиме напаѓач кој што не ја повторува вредноста на нонсот за две прашања кон ист пророк, а исто така работиме и со цели блокови на пораките.

Дефинираме два настани на колизија, именувани како **погоди** (англ. guess)[1]и **точно погоден** (англ. hit)[2]. Настанот **погоди** го дефинираме како:

$$\mathbf{guess} = \bigcup_{i,j,k} \mathbf{guess}(i; j, k),$$
$$\mathbf{guess}(i; j, k) \equiv x_i = s_{j,k}$$

каде што x_i соодветствува на директно прашање до примитивата $f^{\pm}$ за $i \in \{1, \ldots, q_p\}$, а додека пак $s_{j,k}$ е вредност на состојбата на примитивата при енкрипциско прашање за $j \in \{1, \ldots q_{\mathcal{E}}\}$ и $k \in \{1, \ldots, \sigma_{\mathcal{E},j}\}$. Настанот **точно погоден** го деифнираме како:

$$\mathbf{hit} = \bigcup_{j,k,j',k'} \mathbf{hit}(j, k; j', k'),$$
$$\mathbf{hit}(j, k; j', k') \equiv pred(s_{j,k}) \neq pred(s_{j',k'}) \wedge s_{j,k} = s_{j',k'}$$

каде што $j, j' \in \{1, \ldots q_{\mathcal{E}}\}$ и $k, k' \in \{1, \ldots, \sigma_{\mathcal{E},j}\}$, $pred(x)$ - означува состојба која е претходник на состојбата x. Притоа настанот $\mathbf{hit}(j, k; j', k')$ означува точно погодена состојба на примитива од две енкрипциски прашања j' и j.

Според тоа настан на колизија (event) ќе се случи доколу еден од претходните настани се случи, односно event = **guess** $\vee$ **hit**. Со математичка формулација тоа ќе го напишеме како:

$$\Delta_{\mathcal{A}}(f^{\pm}, \mathcal{E}_K; f^{\pm}, \$) \leqslant \mathbf{Pr}(\mathcal{A}^{f^{\pm}, \mathcal{E}_K} \text{ sets event}). \tag{4.4}$$

Веројатноста на настанот да се случи колизија е соодветно дефинирана како во

[2]Понатака во текстот, посебно во делот на математичките формулации, ќе се користат англиските називи на овие два настани.

[55] и тоа:

$$\mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit}) = \mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit} | \neg(\mathbf{key} \vee \mathbf{multi})) + \mathbf{Pr}(\mathbf{key} \vee \mathbf{multi}). \quad (4.5)$$

Забележуваме дека има други два дополнителни настани, **key** и **multi**. Настанот **key** се однесува на сите повици на примитивата кои точно го погодуваат клучот, додека пак настанот **multi** се однесува на состојбите s кои имаат колизија на јавниот дел од состојбата - ратата. Нека $\rho \geqslant 1$ е некој праг за кој што следното е задоволено:

$$max_{\alpha \in \{0,1\}^r} |\{j' \leqslant j, 1 < k' \leqslant k : \alpha\{[s_{j',k'}]^r, [f(s_{j',k'})]^r\}\}| > \rho,$$

за $j \in \{1, \ldots, q_{\mathcal{E}}\}$ и $k \in \{1, \ldots, \sigma_{\mathcal{E},j}\}$.

Настанот guess. Овој настан може да се случи при $i-$то (за $i = 1, \ldots, q_p$) прашање на примитивата или при било која евалуација на состојба на $j-$то енкрипциско прашање (за $j = 1, \ldots, q_{\mathcal{E}}$). Да забележиме дека вредностите на состојбите за $j-$то енкрипциско прашање се следните:

$$\left(s_j^{init}; \quad \begin{bmatrix} s_{j,1,0}^{AD} & s_{j,1,1}^{AD} \\ \vdots & \vdots \\ s_{j,a,0}^{AD} & s_{j,a,1}^{AD} \end{bmatrix}; \quad s_j^{CIS}; \quad s_{j,0}^{SMN}; \quad s_{j,1}^{SMN}; \quad \begin{bmatrix} s_{j,1,0}^{M} & s_{j,1,1}^{M} \\ \vdots & \vdots \\ s_{j,m,0}^{M} & s_{j,m,1}^{M} \end{bmatrix} \right). \quad (4.6)$$

Претпоставуваме дека настанот (**guess** $\vee$ **hit** $\vee$ **key** $\vee$ **multi**) не се има случено претходно, а исто така и настанот (**key** $\vee$ **multi**) не е поставен од соодветното прашање претходно. Од понатамошната анализа ќе ја отстраниме состојбата s_j^{init} од настанот **guess** поради тоа што овој случај припаѓа на настанот **key**. За $i = 1, \ldots, q_p$ нека $j_i \in \{1, \ldots, q_\varepsilon\}$ претставува број на енкрипциски прашања извршени пред i-то прашање на примитивата. Исто така за $j = 1, \ldots, q_{\mathcal{E}}$ нека $i_j \in \{1, \ldots, q_p\}$ е бројот на прашања до примитивата извршени пред j-то енкрипциско прашање. Овде имаме две игри, во

првата играме со прашања до примитивата, а додека пак во втората играме со енкрипциски прашања.

- Да земеме дека имаме прашање до примитивата (x_i, y_i), за $i \in \{1, \ldots, q_p\}$, кое што може да биде нормално или инверзно прашање и претпоставуваме дека тоа прашање не било претходно поставено до $f^{\pm}$. Според тоа за прашање x_i постојат најмногу ρ вредности на состојби кои имаат иста рата и притоа она што е непознато за напаѓачот е тајниот дел од состојбата - капацитетот. Следствено лош настан може да се случи кога $x_i = s$ и тоа со најголема веројатност $\rho/2^c$. За инверзното прашање, ситуацијата е малку покомплексна. Ја земаме вредноста за y_i и ја споредуваме со множеството на сите енкрипциски прашања направени пред i-то прашање на примитивата. Ако $y_i = f(s)$ за било која од состојбите, тогаш и $x_i = s$, инаку $x_i = s$ се одредува со веројатноста $\sum_{j=1}^{j_i} \frac{\sigma_{\epsilon,j}}{2^b}$. Можеме да заклучиме дека веројатноста да се случи настанот **guess** при директни прашања до примитивата е најмногу $\frac{q_p \rho}{2^c} + \sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \frac{\sigma_{\epsilon,j}}{2^b}$.
- Во втората игра поставуваме енкрипциски прашања j за $j \in \{1, \ldots, q_{\mathcal{E}}\}$. Во овој случај треба да ја најдеме веројатноста за која некоја од состојбите дадени во (4.6) е погодена, со претпоставка дека истото прашање не било претходно поставено. Разгледуваме три подслучаи:
  1. Вредноста на состојбата $s_{j,l,0}^{AD}$ за $l = \{1, \ldots, a\}$, се пресметува според $f(s_j^{init}) \oplus (ctr + l)$ каде што ctr е вредноста на бројачот и истата е тајна, односно не е контролирана од страна на напаѓачот. По претпоставка дека вредноста на $f(s_j^{init})$ е случајно избрана од множеството $\{0,1\}^b$, веројатноста да се погоди некоја состојба е $\frac{i_j}{2^b}$ (каде што i_j е бројот на прашања до примитивата извршени пред j-то енкрипциско прашање, а a е максималниот број на блокови на поврзани податоци кои се извршуваат во паралелни гранки). Вредноста на состојбата $s_{j,l,1}^{AD}$ за $l = 1, \cdots, a$, се пресметува како $f(s_{j,l,0}^{AD}) \oplus AD_l$ каде што AD_l е вредност позната на напаѓачот. Веројатноста да се погодат вакви состојби е најмногу $\frac{a i_j}{2^b}$.

2. Следно се ажурира вредноста на заедничката внатрешна состојба за пермутацијата CIS, односно соодветната состојба се пресметува како $s_j^{CIS} = f(s_j^{init}) \oplus T'$. Во овој случај ниту една од вредностите не му е позната на напаѓачот, па според тоа се погодува со веројатност $1/2^b$, односно за j-то прашање $i_j/2^b$.
3. Иста е и ситуацијата кај состојбите во фазата на SMN, притоа за $s_{j,0}^{SMN}$ и $s_{j,1}^{SMN}$, веројатноста е ограничена на $i_j/2^b$.
4. За последното гранење (процесирање на оригиналната порака) ја имаме истата ситуација како и во делот на поврзани податоци.

Од сите овие подслучаи заклучуваме дека $j-$то енкрипциско прашање го достигнува настанот погоди со веројатност не поголема од $\frac{ai_j}{2^b} + \frac{ai_j}{2^b} + 3\frac{i_j}{2^b} + \frac{mi_j}{2^b} + \frac{mi_j}{2^b}$. односно со збир на сите енкрипциски прашања, добиваме:

$$\sum_{j=1}^{q_\mathcal{E}} \frac{ai_j}{2^b} + \frac{ai_j}{2^b} + 3\frac{i_j}{2^b} + \frac{mi_j}{2^b} + \frac{mi_j}{2^b} = \sum_{j=1}^{q_\mathcal{E}} \frac{i_j(2a+2m+3)}{2^b} \leqslant \sum_{j=1}^{q_\mathcal{E}} \frac{i_j \sigma_{\mathcal{E},j}}{2^b}.$$

Користиме дека $\sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \sigma_{\mathcal{E},j} \leqslant q_p \sigma_\mathcal{E}$ каде што j_i секогаш ја има својата максимална вредност $q_\mathcal{E}$ и $\sum_{j=1}^{q_\mathcal{E}} i_j \sigma_{\mathcal{E},j} = \sum_{j=1}^{q_\mathcal{E}} \sum_{k=1}^{\sigma_{\mathcal{E},j}} i_j \leqslant q_p \sigma_\mathcal{E}$ каде што i_j секогаш ја има максималната вредност q_p.

Заклучуваме дека,

$$\begin{aligned} \mathbf{Pr}(\mathbf{guess} | \neg(\mathbf{key} \vee \mathbf{multi})) &\leqslant \frac{q_p \rho}{2^c} + \sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \frac{\sigma_{\mathcal{E},j}}{2^b} + \sum_{j=1}^{q_\mathcal{E}} \frac{i_j \sigma_{\mathcal{E},j}}{2^b} \\ &\leqslant \frac{q_p \rho}{2^c} + \frac{2 q_p \sigma_\mathcal{E}}{2^b}. \end{aligned} \tag{4.7}$$

Настанот hit. Со овој настан ќе ги најдеме сите независни состојби кои прават колизија при енкрипциски прашања. За иницијализациската состојба s^{init} важи: $s_j^{init} \neq s_{j'}^{init}$ во секој случај за $j \neq j'$ заради уникатноста на нонсот. Според тоа било која друга вредност на состојба $s_{j,l}$, точно ја погодува вредноста на иницијалната состојба $s_{j'}^{init}$ само ако $[s_{j,l}]^k = K$, што може да се случи со веројатност $\sigma_\mathcal{E}/2^k$. Во сите други состојби $(\sigma_\mathcal{E} - q_\mathcal{E})$ да се случи

колизија меѓу било кои две состојби $s_{j,l}, s_{j',l'}$, имаме:

- $s^{AD}_{j,l,0} = f(s^{init}_j) \oplus (ctr + l)$, $l = \{1, \ldots, a\}$. Поради фактот што s^{init}_j е секогаш нова вредност, колизија помеѓу $s^{AD}_{j,l,0}$ и некоја друга состојба може да се случи со веројатност не поголема од $a/2^b$ за сите l-ови. Да забележиме дека $s^{AD}_{j,l,0}$ никогаш не може да направи колизија со таков тип состојба од истото енкрипциско прашање поради инкременталната структура на бројоачот за паралелните гранки. Ако $s^{AD}_{j,l,0}$ е нова состојба, тогаш тоа значи дека имаме нов влез во функцијата f и според тоа $s^{AD}_{j,l,1}$ е исто така нова состојба. Ваквата состојба може точно да погоди некоја постара состојба со веројатност $1/2^b$. Ако $s^{AD}_{j,l,1}$ е нова вредност за сите l-ови, тогаш излезот од функцијата f е случајна вредност од множеството $\{0,1\}^b$, и тагот T' генериран на крајот на фазата за процесирање на поврзани податоци е исто така случајна вредност.

- $s^{CIS}_j = f(s^{init}_j) \oplus T'$ точно ја погодува вредноста на состојбата од некое постаро прашање со веројатност најмногу $1/2^b$. Овде имаме еден специјален случај. Вредноста на состојбата s^{CIS}_j може да направи колизија со некоја од состојбите $s^{AD}_{j,l,0}$ од истото прашање ако и само ако $[(ctr + l)||0^*]^r$ е во колизија со T' (се друго е исто, односно состојбата s^{init}_j им е заедничка). Веројатноста за овој потслучај не е поголема од $a/2^r$.

- $s^{SMN}_{j,0} = f(s^{CIS}_j) \oplus (ctr + a + 1)$. Оваа вредност на состојбата е нова, па може да биде во колизија со некоја постара вредност со веројатност $1/2^b$. Истото важи и за $s^{SMN}_{j,1}$.

- $s^{M}_{j,l,0} = f(s^{SMN}_{j,1}) \oplus (ctr + a + 1 + l)$, е нова вредност на состојба за сите паралелни гранки l и велиме дека може да направи колизија со некоја состојба од постаро прашање со веројатност $m/2^b$. Како и до сега, ако $s^{M}_{j,l,0}$ е нова состојба, следствено нови состојби се и $s^{M}_{j,l,1}$ за сите вредности на l и притоа излезот на функцијата f е рамномерен и случаен. Ова докажува дека меѓу вредностите t_j се случајни вредности и дека генерираниот финален таг е исто така случајна вредност.

Земајќи ги сите случаи во предвид, j-то енкрипциско прашање го поставува **hit** со најголема веројатност од: $\frac{(2a+3+2m)}{2^b}(\sigma_{\mathcal{E},1} + \sigma_{\mathcal{E},2} + \ldots + \sigma_{\mathcal{E},j-1}) + \frac{a}{2^r}$.

Со збир на веројатностите од сите енкрипциски прашања, добиваме:

$$\begin{aligned}\mathbf{Pr}(\mathbf{hit}|\neg(\mathbf{key} \vee \mathbf{multi})) &\leqslant \frac{\sigma_{\mathcal{E}}}{2^k} + \sum_{j=1}^{q_{\mathcal{E}}} \frac{(2a+3+2m)}{2^b}(\sigma_{\mathcal{E},1} + \ldots + \sigma_{\mathcal{E},j-1}) + \frac{a}{2^r} \\ &\leqslant \frac{\sigma_{\mathcal{E}}}{2^k} + \frac{q_{\mathcal{E}}a}{2^r} + \frac{\binom{\sigma_{\mathcal{E}}-q_{\mathcal{E}}}{2}}{2^b} \\ &\leqslant \frac{\sigma_{\mathcal{E}}}{2^k} + \frac{q_{\mathcal{E}}a}{2^r} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}}. \end{aligned} \tag{4.8}$$

Настанот key. Овој настан се користи за да ја ограничиме колизијата која се случува помеѓу битовите на клучот K и k-те најлеви битови од x_i од i-то прашање на примитивата каде што $i \in \{1, \ldots q_p\}$. Напаѓачот може да направи најмногу q_p обиди и според тоа имаме $\mathbf{Pr}(\mathbf{key}) \leqslant q_p/2^k$.

Настанот multi. Овој настан се користи за да се ограничи бројот на состојби кои прават колизија во јавниот дел - ратата. Затоа, земаме нова вредност за состојбата $s_{j,l-1}$; понатака за фиксна вредност $x \in \{0,1\}^b$ е задоволено $f(s_{j,l-1}) = x$ или $s_{j,l} = f(s_{j,l-1}) \oplus w = x$ за некоја однапред определена вредност w со веројатност $2/2^b$. Нека $\alpha \in \{0,1\}^r$, повеќе од ρ вредности на состојби точно ја погодуваат α со веројатност не поголема од $\binom{\sigma_{\mathcal{E}}}{\rho}(2/2^r)^\rho$. Ако се повикаме на Стирлинговата апроксимација за факториел ($n! \sim \sqrt{2\pi n}(\frac{n}{e})^n \geqslant (\frac{n}{e})^n$), ќе добиеме $\binom{\sigma_{\mathcal{E}}}{\rho}(2/2^r)^\rho \leqslant (\frac{2e\sigma_{\mathcal{E}}}{\rho 2^r})^\rho$. Ако ги земеме во предвид сите возможни избори за α пресметуваме дека

$$\mathbf{Pr}(\mathbf{multi}) = 2^r \left(\frac{2e\sigma_{\mathcal{E}}}{\rho 2^r}\right)^\rho. \tag{4.9}$$

Поради тоа што ги имаме сите ограничувања за сите настани поединечно, сега

можеме да ја пресметаме и финалната веројатност на напаѓачот.

$$\begin{aligned}\mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit}) &= \mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit} | \neg(\mathbf{key} \vee \mathbf{multi})) + \mathbf{Pr}(\mathbf{key} \vee \mathbf{multi}) \\ &= \mathbf{Pr}(\mathbf{guess} | \neg(\mathbf{key} \vee \mathbf{multi})) + \mathbf{Pr}(\mathbf{hit} | \neg(\mathbf{key} \vee \mathbf{multi})) \\ &+ \mathbf{Pr}(\mathbf{key}) + \mathbf{Pr}(\mathbf{multi}) \\ &\leqslant \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} + \frac{q_p \rho}{2^c} + \frac{q_\varepsilon a}{2^r} + \frac{2q_p \sigma_{\mathcal{E}}}{2^b} + 2^r \left(\frac{2e\sigma_{\mathcal{E}}}{\rho 2^r}\right)^\rho .\end{aligned} \tag{4.10}$$

За да ја поедноставиме претходната евалуација и исто така да го замениме ρ со некои познати вредности, земаме дека $\rho \geqslant max\{r, \sqrt{\frac{2e\sigma_{\mathcal{E}} 2^c}{q_p 2^r}}\}$, па едноставно заменуваме $\rho = r + \sqrt{\frac{2e\sigma_{\mathcal{E}} 2^c}{q_p 2^r}}$ и го добиваме следново:

$$\mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit}) \leqslant \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} + \frac{q_p r}{2^c} + \frac{q_{\mathcal{E}} a}{2^r} + \frac{2q_p \sigma_{\mathcal{E}}}{2^b} + \sqrt{\frac{8e\sigma_{\mathcal{E}} q_p}{2^b}}. \tag{4.11}$$

На крај го имаме финалното ограничување за приватност на π-Cipher:

$$\mathbf{Adv}^{priv}_{\Pi}(\mathcal{A}) \leqslant \frac{(q_p + \sigma_{\mathcal{E}})^2}{2^b} + \frac{2q_p \sigma_{\mathcal{E}}}{2^b} + \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} + \frac{q_p r}{2^c} + \frac{q_{\mathcal{E}} a}{2^r} + \sqrt{\frac{8e\sigma_{\mathcal{E}} q_p}{2^b}}. \tag{4.12}$$

Претпоставуваме дека $\frac{2q_p\sigma_{\mathcal{E}} + (\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2/2}{2^b} \leqslant \frac{(q_p + \sigma_{\mathcal{E}})^2}{2^b}$. Исто така, $\frac{aq_{\mathcal{E}}}{2^r} = \frac{aq_{\mathcal{E}}}{2^{b/2}}$ според дизајнот на пермутациската функција и ги добиваме финалните ограничувања за приватоста на π-Cipher, со што го комплетираме доказот:

$$\mathbf{Adv}^{priv}_{\Pi}(\mathcal{A}) \leqslant \frac{2(q_p + \sigma_{\mathcal{E}})^2}{2^b} + \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{q_p r}{2^c} + \frac{q_{\mathcal{E}} a}{2^{b/2}} + \sqrt{\frac{8e\sigma_{\mathcal{E}} q_p}{2^b}}.$$

□

4.5 Автентичност на π-Cipher

Теорема 3. *Нека $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е предложената шема π-Cipher каде што основната криптографска примитива е заменета со идеална примитива p која*

што оперира над b битови. Значи,

$$\boldsymbol{Adv}_{\Pi}^{Auth}(\mathcal{A}) \leqslant \frac{(q_p+\sigma_{\mathcal{E}}+\sigma_{\mathcal{D}})^2}{2^b} + \frac{q_p+q_{\mathcal{D}}+\sigma_{\mathcal{E}}+\sigma_{\mathcal{D}}}{2^k} + \frac{q_p r}{2^c} + \frac{a(q_{\mathcal{E}}+q_{\mathcal{D}})}{2^{b/2}} + \frac{\sigma_{\mathcal{D}}(q_p+\sigma_{\mathcal{E}}+\sigma_{\mathcal{D}}/2)}{2^c} + \sqrt{\frac{8e\sigma_{\mathcal{E}}q_p}{2^b}},$$

каде што σ_ε и $\sigma_{\mathcal{D}}$ се дефинирани во (4.1), q_p е максималниот број на прашања до идеалната пермутација $p^{\pm}$, q_ε е максималниот број на енкриптирачки прашања со вкупна должина од λ_ε блокови, $q_{\mathcal{D}}$ е максималниот број на декриптирачки прашања со вкупна должина од $\lambda_{\mathcal{D}}$ блокови и a е максималниот број на блокови за поврзани податоци.

доказ. Фалсификат на AEAD шема е дефиниран како можност на напаѓачот $\mathcal{A}$ да генерира валидни (N, AD, C, T) торки, без директно поставување прашања на енкрипциските пророк. Поинаку кажано, напаѓачот пробува да постави декрипциско прашање кое нема да резултира со неважечки симбол $\perp$.

Нека $\Pi = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ е π-Cipher дефиниран според Дефиниција 10 со иделана пермутација (π-функција) која што оперира над $b = (r + c)$ битови. Напаѓачот $\mathcal{A}$ има пристап до енкрипциските пророк $\mathcal{E}$, декрипциските пророк $\mathcal{D}$, идеалната пермутациска функција p и нејзината инверзна p^{-1} функција. Наша цел е да ја ограничиме предноста на напаѓачот да ја фалсификува шемата $\Pi = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$:

$$Adv_{\Pi}^{Auth}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{p^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}). \tag{4.13}$$

Со замена на идеалната пермутација p со двонасочна функција $f : \{0,1\}^b \to \{0,1\}^b$ и со користење на PRP/PRF Switching Lemma [89] исто така користејќи го фактот дека напаѓачот не може да направи повеќе од $q_p + \sigma_\varepsilon + \sigma_{\mathcal{D}}$ евалуации до функцијата f, ги добиваме следниве ограничувања:

$$\begin{aligned}\mathbf{Pr}(\mathcal{A}^{p^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}) - \mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}) &\leqslant \frac{(q_p+\sigma_{\mathcal{E}}+\sigma_{\mathcal{D}})(q_p+\sigma_\varepsilon+\sigma_{\mathcal{D}}-1)}{2^{b+1}} \\ &\leqslant \frac{(q_p+\sigma_{\mathcal{E}}+\sigma_{\mathcal{D}})^2}{2^{b+1}}.\end{aligned}$$

За да ја исполниме нашата цел, ќе се фокусираме на напаѓач $\mathcal{A}$ кој што има

пристап до пророкот $(f^{\pm}, \mathcal{E}, \mathcal{D})$ и кој што прашува само прашања со цели блокови. Исто така претпоставуваме дека напаѓачот $\mathcal{A}$ игра со респект спрема нонсот, што означува дека тој никогаш не поставува две прашања до $\mathcal{E}$ со користење на ист нонс, но му е дозволено да го повторува нонсот кога прави декрипциски прашања.

За овој доказ ние ќе го користиме истото поставување како и при доказот за приватност, за настаните **guess** и **hit**, со таа разлика што тука истите ќе ги проширime со нови $\mathcal{D}$-поврзани настани на колизија, $\mathcal{D}_{\mathbf{guess}}$ и $\mathcal{D}_{\mathbf{hit}}$. Вредностите за состојбите се исти како и во (4.6) само со додаден симболот δ како индекс, каде што $\delta \in \{\mathcal{E}, \mathcal{D}\}$. Ние заклучивме дека:

$$\begin{aligned}\mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}) \leqslant \mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}|\neg\text{event}) + \\ \mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ sets event}). \qquad (4.14)\end{aligned}$$

Границата на веројатноста дека $\mathcal{A}$ фалсификува кога настанот не е случен е иста со случајот кога $\mathcal{A}$ може да го погоди финалниот таг од некое декрипциско прашање j. Финалниот таг се пресметува како $f(s^{AD}_{\mathcal{D},j,1,1}) \boxplus \ldots \boxplus f(s^{AD}_{\mathcal{D},j,a,1}) \boxplus f(s^{SMN}_{\mathcal{D},j,1}) \boxplus f(s^{M}_{\mathcal{D},j,1,1}) \boxplus \ldots \boxplus f(s^{M}_{\mathcal{D},j,m,1})$. Поради тоа што прашањето е ново и притоа настаните **guess** и **hit** не се случиле претходно значи дека најмалку една од овие состојби е нова, па затоа излезот од функцијата f за таа состојба ќе биде рамномерно распределен со случајни битови, па следствено и финалниот таг е случаен. Во овој случај, j-от обид за фалсификат е успешен со веројатност најмногу $1/2^{\tau}$. Со збир за сите декрипциски прашања $q_{\mathcal{D}}$ добиваме:

$$\mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ forges}|\neg\text{event}) \leqslant \frac{q_{\mathcal{D}}}{2^{\tau}}.$$

Должината на финалниот таг е иста со должината на клучот како што претходно објаснавме во Секција 4.2, па според тоа слободно може да пишуваме дека веројатноста на напаѓачот да ја фалсификува шемата кога настанот не се случил е $\frac{q_{\mathcal{D}}}{2^k}$.

Следно, треба да се даде објаснување за тоа кои се границите на веројатноста кога напаѓачот го погодува настанот. Овде, настанот е дефиниран како: event =

$\mathbf{guess} \vee \mathbf{hit} \vee \mathcal{D}_{\mathbf{guess}} \vee \mathcal{D}_{\mathbf{hit}}$ и притоа

$$\mathbf{Pr}(\mathcal{A}^{f^{\pm},\mathcal{E}_K,\mathcal{D}_K} \text{ sets event}) \leqslant \mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit} \vee \mathcal{D}_{\mathbf{guess}} \vee \mathcal{D}_{\mathbf{hit}}) \leqslant$$
$$\mathbf{Pr}(\mathbf{guess} \vee \mathbf{hit} \vee \mathcal{D}_{\mathbf{guess}} \vee \mathcal{D}_{\mathbf{hit}} | \neg(\mathbf{key} \vee \mathbf{multi})) + \mathbf{Pr}(\mathbf{key} \vee \mathbf{multi}). \quad (4.15)$$

Настанот $\mathcal{D}_{guess}$ Да забележиме дека во рамките на декрипциските прашања, напаѓачот може сам да си ја одреди ратата на состојбата. Согласно на тоа, вредноста на шифрираниот текст која напаѓачот сам ја избира за декрипциските прашања, всушност ја дефинираат ратата на состојбата. Според тоа $\mathcal{D}_{guess}$ е лош настан се додека постои декрипциско прашање и прашање кон примитивата чии делови на капацитет се еднакви. Ова се случува со веројатност не поголема од $q_p\sigma_{\mathcal{D}}/2^c$,

$$\mathbf{Pr}(\mathcal{D}_{\mathbf{guess}} | \neg(\mathbf{key} \vee \mathbf{multi})) \leqslant q_p\sigma_{\mathcal{D}}/2^c. \quad (4.16)$$

Настанот $\mathcal{D}_{\mathbf{hit}}$. Кај овој настан, напаѓачот има можност да го реискористи нонсот во декрипциските прашања. Мораме да напоменеме тука дека само јавниот дел од нонсот може да се реискористи, PMN. Во овој случај SMN е сеуште таен број и непознат за напаѓачот. Секоја состојба во декрипциските прашања, може да биде точно погодена од иницијалната (каде што само делот за клучот е непознат) со најголема веројатност од $\sigma_{\mathcal{D}}/2^k$. За остатокот ги имаме следните подслучаи:

1. $(PMN; AD, C) = (PMN_{\delta,j}; AD_{\delta,j}, C_{\delta,j})$ но $T \neq T_{\delta,j}$. Овој случај е невозможен и веројатноста е 0, поради тоа што финалниот таг е ист само доколку PMN, AD, SMN и оригиналната порака се исти.

2. $(PMN; AD) = (PMN_{\delta,j}; AD_{\delta,j})$ но $C \neq C_{\delta,j}$. Нека шифрираните текстови $C \neq C_{\delta,j}$ се различни во сите нивни блокови, тогаш $s_{j,0}^{SMN} = s_{\delta,j,0}^{SMN}$ и $s_{j,1}^{SMN} = C_0||[s_{\delta,j,1}^{SMN}]_c \neq s_{\delta,j,1}^{SMN}$. Ова значи дека состојбата е нова и може точно да се погоди со некоја постара состојба со веројатност $1/2^c$. Вкупно, j-то декрипциско прашање го поставува настанот $\mathcal{D}_{\mathbf{hit}}$ со веројатаност најмногу: $\frac{\sigma_{\mathcal{E}}\sigma_{\mathcal{D},j}+\sigma_{\mathcal{D},j}(\sigma_{\mathcal{D},1}+\ldots+\sigma_{\mathcal{D},j-1})}{2^c}$.

Овде имаме еден специјален случај, што ако SMN е ист број? Тоа значи дека $C_0 = C_{\delta,j,0}$ и заедничката состојба CIS во фазата на обработка на пораката е иста $f(s_{j,1}^{SMN}) = f(s_{\delta,j,1}^{SMN})$. Па според тоа, резонирањето овде се поистоветува со случајот кога остатокот од шифрираниот текст е различен односно тој има најдолг заеднички префикс.

Нека земеме дека $C_{\delta,j}$ дели најдолг заеднички префикс l со C и притоа $l < m$. Во овој случај $s_{j,l,0}^{C} = s_{\delta,j,l,0}^{C}$ и $s_{j,l,1}^{C} = C_l||[s_{\delta,j,l,1}^{C}]_c \neq s_{\delta,j,l,1}^{C}$, значи $s_{j,l,1}^{C}$ е нова состојба и нов влез за функцијата f. Таа може со точност да погоди некоја конкретна постара состојба со веројатност $1/2^c$, па истата граница може да се искористи од претходно.

3. $PMN = PMN_{\delta,j}$ но $AD \neq AD_{\delta,j}$. Анализата е иста како и во случајот за шифрираниот текст, освен сучајот на внатрешна колизија која може да настане помеѓу состојбата на ажурираната вредност за CIS и состојбите во фазата на поврзани податоци (веројатноста за овој случај е $a/2^r$). Во остатокот резонирањето се сведува на тоа дека сите состојби се нови.

4. $PMN \neq PMN_{\delta,j}$. Ако јавниот дел од нонсот е различен, тогаш секогаш со сигурност состојбата s^{init} е нова, па според тоа нови се и сите понатамошни состојби од дизајнот.

Со збир на веројатностите од сите прашања добиваме:

$$\begin{aligned}\Pr(\mathcal{D}_{\mathbf{hit}}|\neg(\mathbf{key} \vee \mathbf{multi})) &\leqslant \sum_{j=1}^{q_{\mathcal{D}}} \frac{\sigma_{\mathcal{E}}\sigma_{\mathcal{D},j} + \sigma_{\mathcal{D},j}(\sigma_{\mathcal{D},1} + \ldots + \sigma_{\mathcal{D},j-1})}{2^c} \\ &\quad + \frac{q_{\mathcal{D}}a}{2^r} + \frac{\sigma_{\mathcal{D}}}{2^k} \\ &\leqslant \frac{\sigma_{\mathcal{D}}\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}^2/2}{2^c} + \frac{q_{\mathcal{D}}a}{2^r} + \frac{\sigma_{\mathcal{D}}}{2^k}.\end{aligned}$$

Заедно со ограничувањата од доказот за приватност преку (4.15) добиваме:

$$\mathbf{Pr}(\text{event}) \leqslant \frac{q_p + q_{\mathcal{D}} + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}}{2^k} + \frac{q_p\sigma_{\mathcal{E}}}{2^{b+1}} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} + \frac{aq_{\mathcal{E}} + aq_{\mathcal{D}}}{2^{b/2}} + \sqrt{\frac{8e\sigma_{\mathcal{E}}q_p}{2^b}} + \frac{q_p\sigma_{\mathcal{D}}}{2^c} + \frac{\sigma_{\mathcal{D}}\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}^2/2}{2^c} + \frac{q_p r}{2^c}. \tag{4.17}$$

За крај, ги добиваме финалните гранични вредности за интегритетот на шемата на π-Cipher:

$$\mathbf{Adv}_{\Pi}^{Auth}(\mathcal{A}) \leqslant \frac{(q_p + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})^2}{2^{b+1}} + \frac{q_{\mathcal{D}}}{2^k} + \frac{q_p + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}}{2^k} + \frac{q_p\sigma_{\mathcal{E}}}{2^{b+1}} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} + \frac{aq_{\mathcal{E}} + aq_{\mathcal{D}}}{2^{b/2}} + \sqrt{\frac{8e\sigma_{\mathcal{E}}q_p}{2^b}} + \frac{q_p\sigma_{\mathcal{D}}}{2^c} + \frac{\sigma_{\mathcal{D}}\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}^2/2}{2^c} + \frac{q_p r}{2^c}. \tag{4.18}$$

Земаме во предвид дека $\frac{q_p\sigma_{\mathcal{E}}}{2^{b+1}} + \frac{(\sigma_{\mathcal{E}} - q_{\mathcal{E}})^2}{2^{b+1}} \leqslant \frac{(q_p + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})^2}{2^{b+1}}$ со што го комплетираме доказот:

$$\mathbf{Adv}_{\Pi}^{Auth}(\mathcal{A}) \leqslant \frac{(q_p + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})^2}{2^b} + \frac{q_p + q_{\mathcal{D}} + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}}{2^k} + \frac{q_p r}{2^c} + \frac{a(q_{\mathcal{E}} + q_{\mathcal{D}})}{2^{b/2}} + \frac{\sigma_{\mathcal{D}}(q_p + \sigma_{\mathcal{E}} + \sigma_{\mathcal{D}}/2)}{2^c} + \sqrt{\frac{8e\sigma_{\mathcal{E}}q_p}{2^b}}.$$

□

4.6 Сигурност на пермутацискатa функција π

Пермутацискатa функција π како главен двигател на сигурноста на шифрувачот во претходниот доказ ја зедовме со претпоставка дека е идеална пермутација. Претходно напоменавме дека π функцијата главно се базира на употреба на квазигрупи со чија помош се дефинирани основните ARX операции.

Ако ја разгледуваме функцијата π како ARX базирана симетрична крипто примитива, треба да се направи диференцијална криптоанализа на истата. Диференцијалната криптоанализа има за цел да проучува како разликата во влезовите се пропагира низ целата примитива и притоа влијае на излезот. Доколку се појават сериозни отстапувања од рамномерно случајните секвенци, тоа доведува

до слабости на примитивата кои понатака можат да бидат искористени за напад на клучот или за произведување на фалсификат на податоците.

Во литературата постојат многу истажувања во областа на диференцијална криптоанализа, меѓутоа тие се однесуваат за супституциско-пермутациски мрежи. Помалку е направено во областа на диференцијалните карактеристики на ARX базираните примитиви. Да напоменеме дека класичната диференцијална криптоанализа која што е наменета за супституциско-пермутациските мрежи не може да се примени кај ARX дизајните. Две групи на истражувачи од Франција [63] и Белгија [73] се пионери во автоматското изучување на диференцијалните карактеристики на ARX дизајните. Проблемот тука е што нивните автоматски алатки се ограничени на мало множество од операции и променливи. Во нашиот случај овие алатки не можат да се применат на функцијата π, туку со користење на нивната методологија, може да се изнајде начин и истиот да се генерализира за ARX дизјни со поголем број на променливи во влезот. Овој тип на истражување е надвор од темата на овој докторски труд, па затоа ќе го оставиме ова прашање како отворен проблем за понатамошна работа [91].

Она што ќе го направиме за нашата π функција е мерење на некои основни параметри за да може да се каже дека пермутацијата игра улога на рамномерно случајна пермутација.

4.6.1 Тестирање на ефектот на лавина на функцијата π

Во симетричната криптографија ефектот на лавина (англ. *avalanche effect*) е еден од својствата што треба да се испита за да се одреди сигурноста на криптографската примитива. Ефектот на лавина се постигнува тогаш кога влезот на примитивата ќе се смени скоро незабележително (да речеме со промена на еден бит), а излезот на истата ќе се смени драстично (скоро половина од излезните битови се промениле). Ефектот а лавина во криптографијата е мерка за рандомизација на битовите.

Анализа на дифузија на битовите:

Ќе покажеме практично, нумерички колкав е ефектот на лавина на операцијата * и една рунда од функцијата π за сите вредности на должината на зборовите

$\omega = 16, 32, 64$.

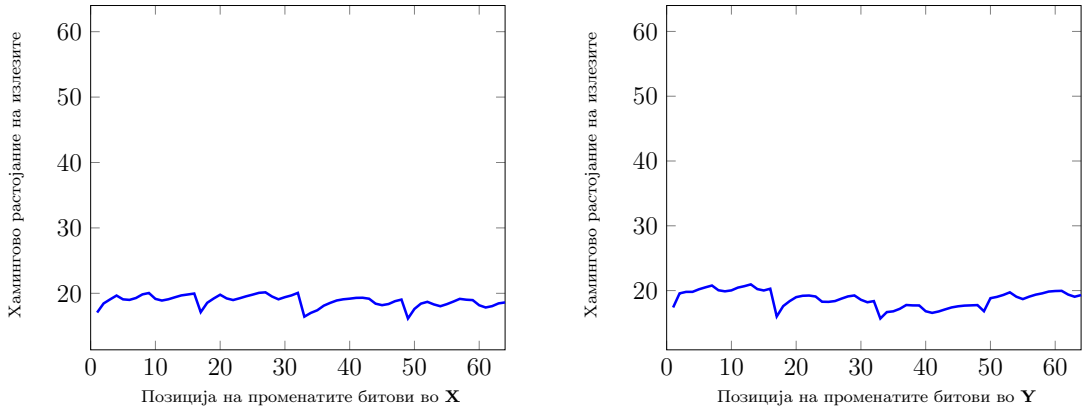
Според тоа имаме две експериментални поставувања:

1. Мерење на пропагацијата од еден бит разлика во влезот (X, Y) на операцијата *, над 10000 случајно избрани примероци;

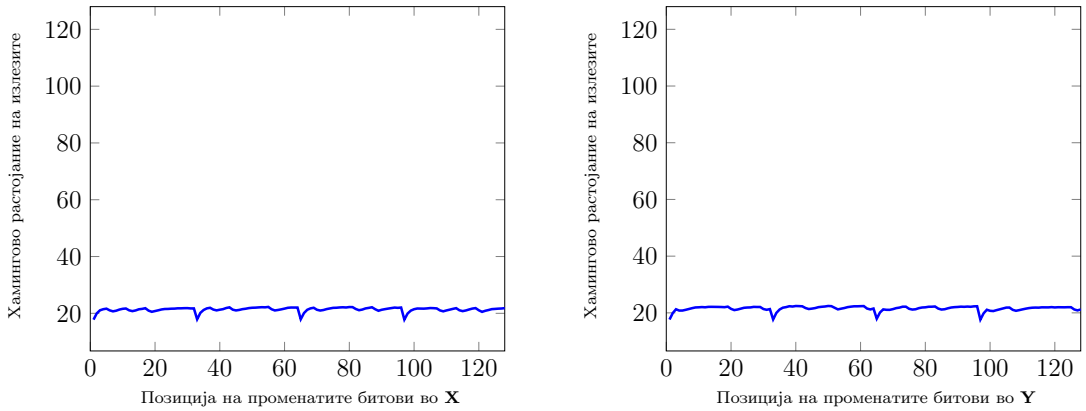
2. Мерење на пропагацијата од еден бит разлика на интерната состојба на функцијата π, над 1000 случајно избрани примероци;

3. Мерење на пропагацијата од еден бит разлика на интерната состојба на функцијата π, кога влезот во функцијата се состои само од нули.

За експериментот под број 1, генериравме 10000 случајни вредности за променливите $\mathbf{X}$ и $\mathbf{Y}$ од операцијата $*$. Нека $\mathbf{X}, \mathbf{X}', \mathbf{Y}, \mathbf{Y}' \in \mathcal{Q}_q, q = 64, 128, 256$ се претставени како $\mathbf{X} = (X_0, X_1, X_2, X_3)$, $\mathbf{X}' = (X'_0, X'_1, X'_2, X'_3)$, $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$, $\mathbf{Y}' = (Y'_0, Y'_1, Y'_2, Y'_3)$ и нека $\mathbf{Z} = \mathbf{X} * \mathbf{Y}$, $\mathbf{Z}' = \mathbf{X}' * \mathbf{Y}$ и $\mathbf{Z}'' = \mathbf{X} * \mathbf{Y}'$. За првиот подслучај на експериментот земаме $\mathbf{X}$ и $\mathbf{X}'$ да се разликуваат во само еден бит, односно Хаминговото растојание $H(\mathbf{X}, \mathbf{X}') = 1$ и мериме колкаво ќе биде Хаминговото растојание во резултатот од операцијата *, односно $H(\mathbf{Z}, \mathbf{Z}')$. Направени се по 10000 експерименти со промена на секој бит од променливата $\mathbf{X}$, односно со вкупна промена на 64, 128 и 256 бита. Вториот потслучај е кога $\mathbf{Y}$ и $\mathbf{Y}'$ се разликуваат во само еден бит, односно Хаминговото растојание $H(\mathbf{Y}, \mathbf{Y}') = 1$ и мериме колкаво ќе биде Хаминговото растојание во резултатот од операцијата *, односно $H(\mathbf{Z}, \mathbf{Z}'')$. Исто така направени се по 10000 експерименти со промена на секој бит од променливата $\mathbf{Y}$, односно со вкупна промена на 64, 128 и 256 бита. Резултатите од мерењата за сите вредности на параметарот $\omega = 16, 32, 64$ (должина на збор) се прикажани на Слика 4-2, Слика 4-3 и Слика 4-4. Од прикажаните податоци може да се забележи дека операцијата *, сама по себе не дава добри својства на дифузија на битовите, односно не го достигнува прагот на ефектот на лавина. Тоа зборува дека голема е веројатноста да се открие влезот, знаејќи го излезот од операцијата *. Токму затоа ние операцијата * не ја користиме како засебна, туку ја комбинираме со квазигрупна стринг трансформација за да добиеме поголема дифузија на битовите.



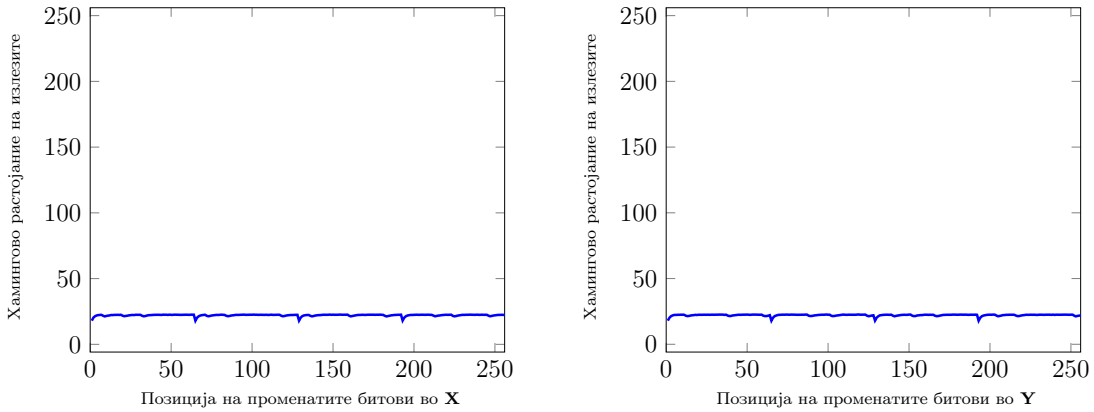
Слика 4-2: Ефектот на лавина на операцијата $*$ за $\omega = 16$



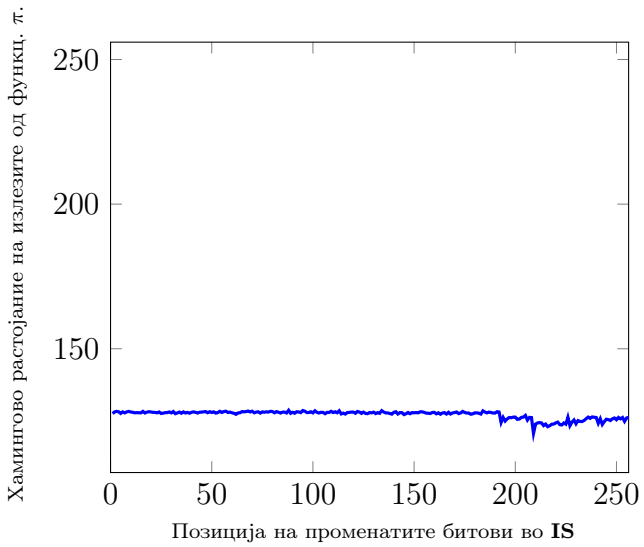
Слика 4-3: Ефектот на лавина на операцијата $*$ за $\omega = 32$

За експериментот под број 2, генерираме 1000 случајни вредности за интерната состојба IS на пермутациската функција π со една рунда. Нека $\mathbf{IS}, \mathbf{IS}' \in \mathcal{Q}_q^N, q = 64, 128, 256, N = 4$ се претставени како $\mathbf{IS} = (\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4), \mathbf{IS}' = (\mathbf{I}'_1, \mathbf{I}'_2, \mathbf{I}'_3, \mathbf{I}'_4)$ и нека $\mathbf{J} = \pi_R(\mathbf{IS})$ и $\mathbf{J}' = \pi_R(\mathbf{IS}')$. Притоа земаме $\mathbf{IS}, \mathbf{IS}'$ да се разликуваат во точно еден бит, односно Хаминговото растојание меѓу нив да биде 1, $H(\mathbf{IS}, \mathbf{IS}') = 1$. Мериме колкаво ќе биде Хаминговото растојание на излезите на функцијата π со промена на секој бит во влезот **IS**. Резултатите од мерењата за сите вредности на параметарот $\omega = 16, 32, 64$ (должина на збор) се прикажани на Слика 4-5, Слика 4-6 и Слика 4-7.

За експериментот под број 3, земаме интерната состојба да биде 0 и ја при-



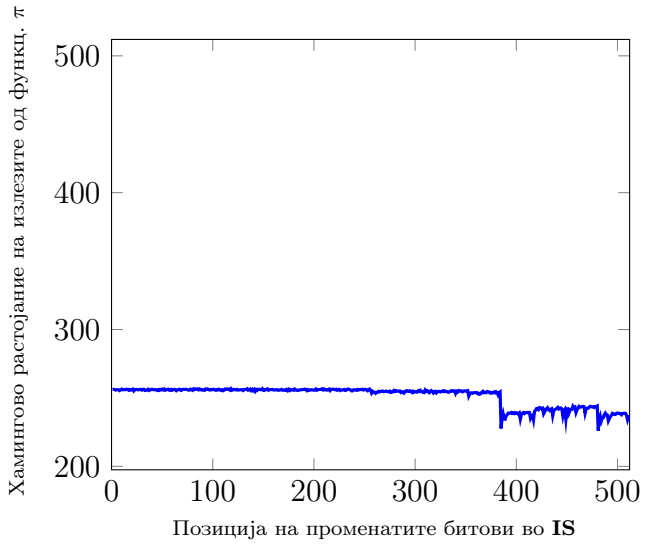
Слика 4-4: Ефектот на лавина за операцијата $*$ за $\omega = 64$



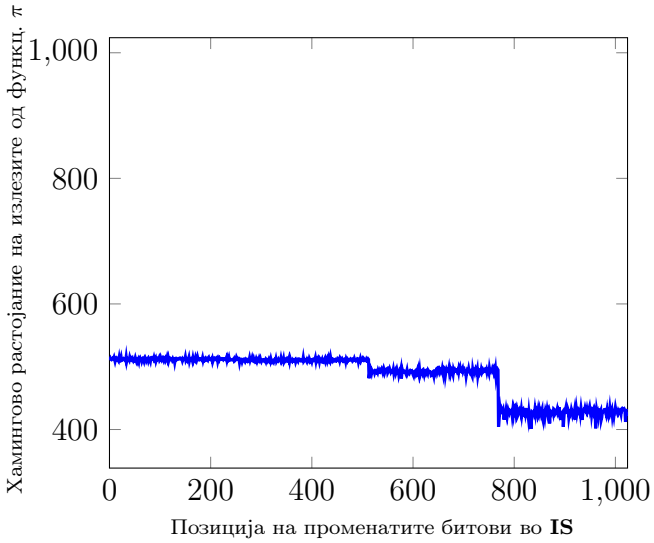
Слика 4-5: Ефектот на лавина за една рунда на функцијата π кога $\omega = 16$ ($Min = 120.732$, $\mathbf{Avg} = \mathbf{127.255}$, $Max = 128.731$)

менуваме премутациската функција π со една рунда. Мериме колкаво ќе биде Хаминговото растојание на излезите на функцијата π со промена на секој бит поединечно во влезот **IS**. т.е мериме колку е вредноста на $H(\pi_R(\mathbf{IS}), \pi_R(\mathbf{IS}'))$, кога $H(\mathbf{IS}, \mathbf{IS}') = 1$ и $\mathbf{IS} = 0$. Резултатите се прикажани соодветно на Слика 4-8, Слика 4-9 и Слика 4-10.

Од последните два експерименти (2 и 3) јасно се гледа дека е постигнат ефектот на лавина за пермутациската функција π дури и за една рунда. Резултатите

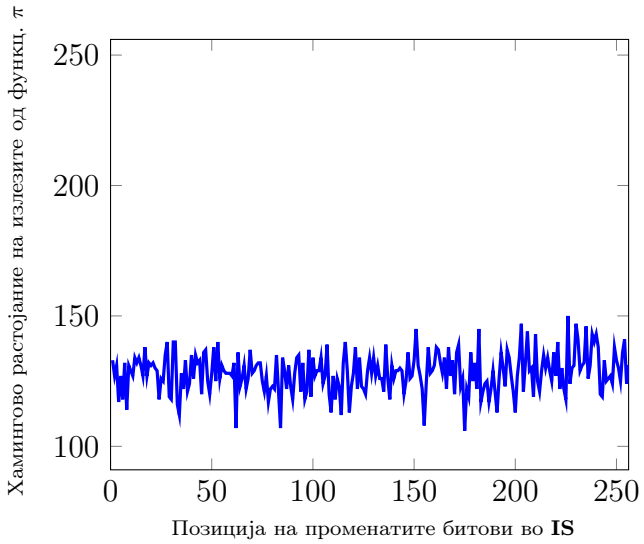


Слика 4-6: Ефектот на лавина за една рунда на функцијата π кога $\omega = 32$ ($Min = 226.063$, **Avg** $=$ **256.765**, $Max = 251.472$)

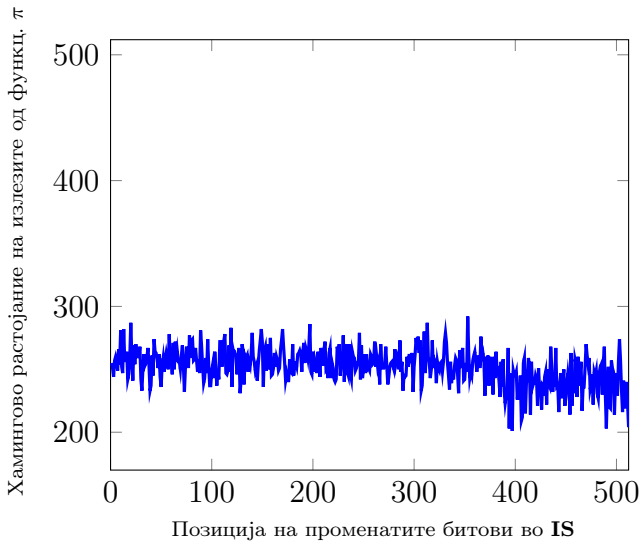


Слика 4-7: Ефектот на лавина за една рунда на функцијата π кога $\omega = 64$ ($Min = 400.88$, **Avg** $=$ **485.646**, $Max = 515.76$)

за $\omega = 16$ односно $|IS| = 256$ покажуваат дека Хаминговото растојание е половина од големината на влезот, ≈ 128 бита во сите случаи на тестирање на функцијата π. Соодветно истото важи и за другите вредности на ω, со што е потврден теоретскиот модел за идеална случајна пермутација.

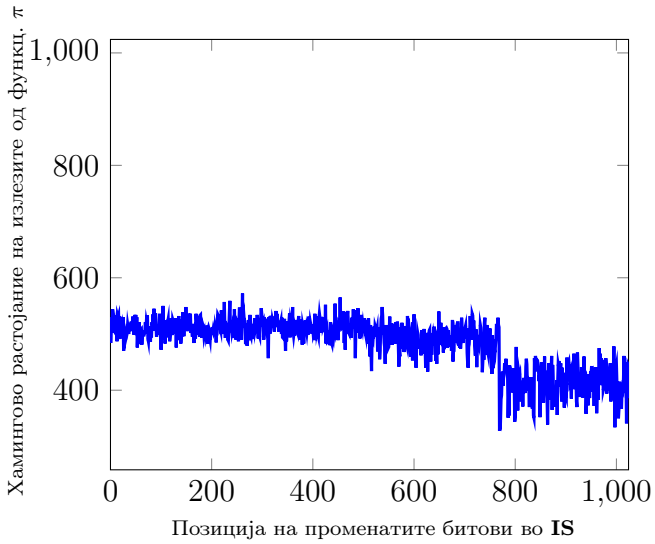


Слика 4-8: Ефектот на лавина за една рунда на функцијата π кога $\omega = 16$ и **IS** $= 0$ ($Min = 106.0$, **Avg** $=$ **128.238**, $Max = 150.0$)



Слика 4-9: Ефектот на лавина за една рунда на функцијата π кога $\omega = 32$ и **IS** $= 0$ ($Min = 201.0$, **Avg** $=$ **251.857**, $Max = 292.0$)

Дифузијата на битовите, односно колку пермутацијата π игра улога на идеална случајна пермутација, потврдено е и преку алатката TestU01 [62]. TestU01 е софтверска библиотека имплементирана во програмскиот јазик C, која што нуди различни батерии со статистички тестови за мерење на рамномерната случајнос-



Слика 4-10: Ефектот на лавина за една рунда на функцијата π кога $\omega = 64$ и $\mathbf{IS} = 0$ ($Min = 328.0, \mathbf{Avg} = \mathbf{481.805}, Max = 572.0$)

та на битовите во излезот.

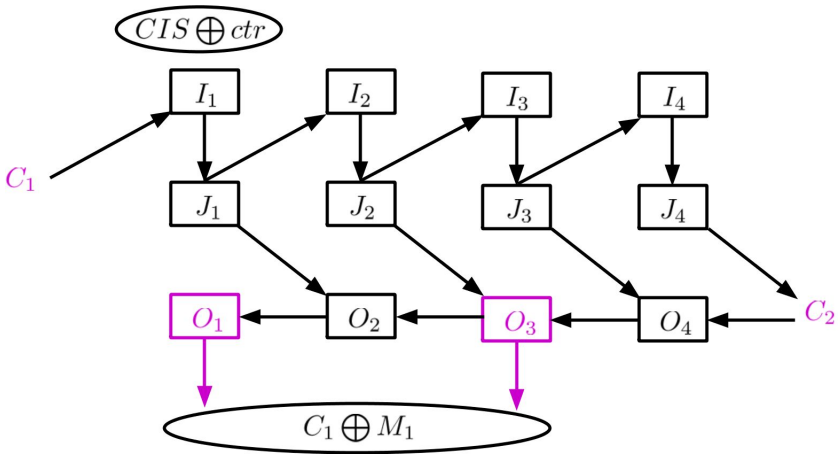
4.7 Криптоанализа на 16 битната варијанта на π-Cipher

Во овој дел во детали ќе опишеме напад на една рунда од π16-Cipher096.

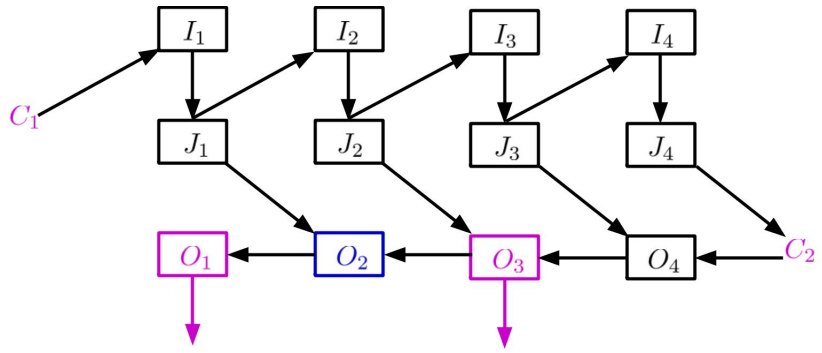
Доколку π16-Cipher096 работи во мод на една рунда тоа значи дека функцијата π се состои од само едно извршување на функцијата π_R и на напаѓачот ќе му бидат познати 128 бита од шифрираниот текст, односно половина од состојбата на пермутацискатa функција. Според дизајнот на π-Cipher ако излезот од функцијата π е $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2, \mathbf{O}_3, \mathbf{O}_4)$, на напаѓачот му се познати вредностите $\mathbf{O}_1$ и $\mathbf{O}_3$ кои сами по себе се 64 битни вредности, односно четворки од по 16 бита. Други познати вредности за напаѓачот се и рундовските константи, во случајов константите $\mathbf{C}_1$ и $\mathbf{C}_2$. Според ова сценарио напаѓачот применува напад со познат шифриран текст и негова задача е да дознае што повеќе бита од капацитетот (тајниот дел) од внатрешната состојба на функцијата π, односно она што треба да се потруди да го открие напаѓачот е $\mathbf{O}_2$ или $\mathbf{O}_4$, Слика 4-11. Знаејќи една од

овие вредности лесно може да стигне до откривање на целата внатрешна состојба на функцијата π, а со тоа и можност за разбивање на клучот и манипулација со шифрувачот.

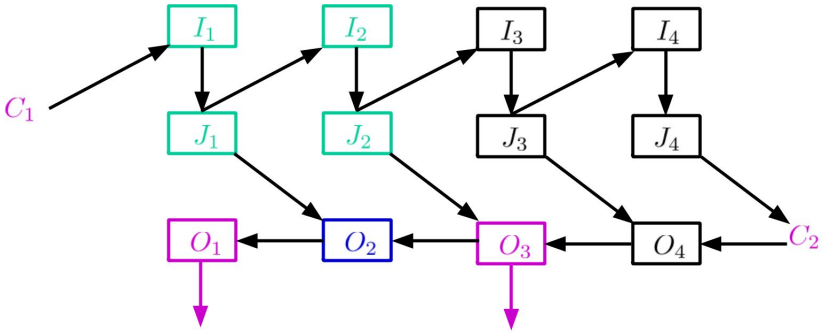
За да може да се имплементира овој напад, се користи техниката "*$\overline{и}о\overline{ш}ogu$ $и$ $ог\overline{ш}а\overline{ш}ни*". Значи, за $\omega = 16, q = 64$ и притоа треба да се погодат сите 64 бита за променливата $\mathbf{O}_2$, графичкиот приказ е на Слика 4-12. Со помош на методата груба сила, односно со пробување на сите можни 2^{64} вредности за $\mathbf{O}_2$, на единствен начин ја формираме листата $\mathcal{L}_1 = \{(\mathbf{I}_1^{(i)}, \mathbf{I}_2^{(i)}) \mid 0 \leqslant i < 2^{64}\}$ од сите 2^{64} различни вредности за паровите $(\mathbf{I}_1^{(i)}, \mathbf{I}_2^{(i)})$ влезови во функцијата π (види Слика 4-13). Ваква слична листа може да се направи и за следниот енкриптирачки блок од порака, $\mathcal{L}_2 = \{(\mathbf{II}_1^{(i)}, \mathbf{II}_2^{(i)}) \mid 0 \leqslant i < 2^{64}\}$. Знаејќи дека при енкрипција на два последователни блокови од пораката, секогаш на почеток се прави инектирање на бројачот во заедничката внатрешна состојба CIS и притоа бројачот за вториот блок е зголемен за 1 од вредноста што ја имал за претходниот блок, следува дека вредностите на променливите $\mathbf{I}_1^{(i)}$ и $\mathbf{II}_1^{(i)}$ ќе се разликуваат за 1, додека пак $\mathbf{I}_2^{(i)}$ и $\mathbf{II}_2^{(i)}$ имаат иста вредност. Значи кога ќе направиме пресек на двете добиени листи $\mathcal{L}_1$ и $\mathcal{L}_2$ со веројатност 1 ќе ги добиеме бараните парови. Комплексноста на овој напад е $2 * 2^{64}$ пресметки на операцијата *, а комплексноста на меморискиот простор кој се зафаќа е $2^{65} * 16 = 2^{69}$ бајти.



Слика 4-11: Функцијата π со една рунда за π16-Cipher096



Слика 4-12: O_2 ја погодуваме со пробување на сите можни 2^{64} вредности



Слика 4-13: Сите зелени правоаголници се вредности добиени на единствен начин според дефиницијата на операцијата $*$ според соодветно погодена вредност за O_2

Овој тип на напад е проширен на 2 рунди за π16-Cipher096 од страна на две различни групи криптографи и објавен во трудовите [3] и [18]. Во продолжение ќе објаснеме како претходно дефинираната слабост на шифрувачот π16-Cipher096 од наша страна беше проширена на 2 рунди (види Слика 4-14).

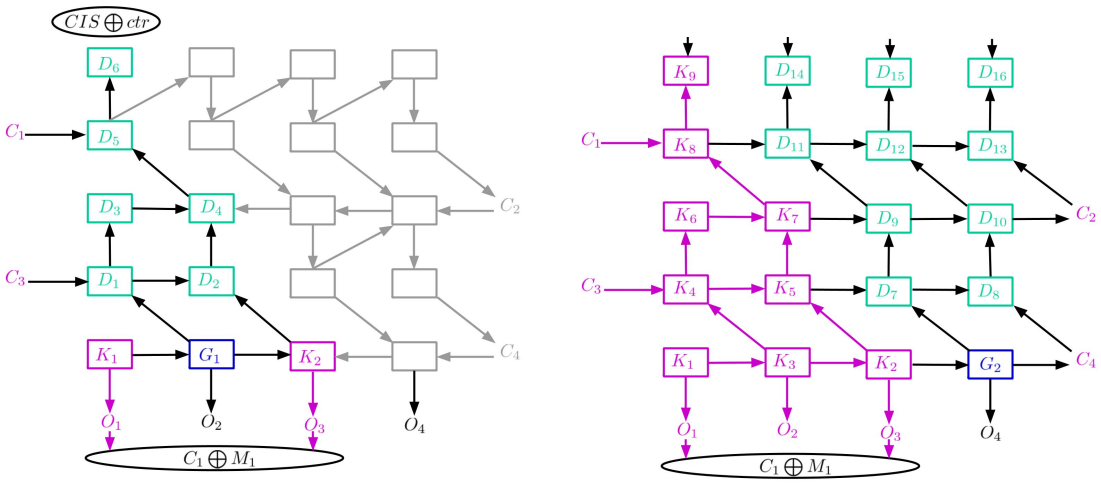
Нека функцијата π е дефинриана со две рунди. Го користиме истото сценарио како и претходно, односно знаејќи ги оригиналниот и шифрираниот текст ние добиваме информација за излезот O_1 и O_3 од функцијата π. Со користење на техниката *поголеми и одземени* може да добиеме информација за променливата $CIS \bigoplus ctr$ од влезот. Нека познатите вредности ги обележиме со K_1 и K_2 соодветно и нека ја погодуваме вредноста на O_2 што е означена со G_1. Ја имаме следната листа од чекори:

1. Искористи ги K_1 и G_1, за да го определиш D_1;
2. Искористи ги G_1 и K_2, за да го определиш D_2;
3. Искористи ги C_1 и D_1, за да го определиш D_3;
4. Искористи ги D_1 и D_2, за да го определиш D_4;
5. Искористи ги D_3 и D_4, за да го определиш D_5;
6. Искористи ги C_1 и D_5, за да го определиш D_6;

Графички приказ на оваа процедура е даден на Слика 4-14а. Откако истава постапка ќе се повтори и за вториот екриптирачки блок од пораката, вредноста за $CIS \bigoplus ctr$ ќе се одгатни и сите вредности наназад $D_5 \ldots D_1$ ќе постанат конкретни вредности $K_5 \ldots K_1$, а не множество од 2^{64} можни кандидати.

Понатака, повторно се користи техниката *\overline{и}о\overline{г}оди и о\overline{г}а\overline{ш}ни*, така што сега вредноста што ја погодуваме е O_4 и истата е означена со G_2. Ја имаме следната листа од чекори:

1. Искористи ги K_2 и G_2, за да го определиш D_7;
2. Искористи ги G_2 и C_4, за да го определиш D_8;
3. Искористи ги K_5 и D_7, за да го определиш D_9;
4. Искористи ги D_7 и D_8, за да го определиш D_{10};
5. Искористи ги K_7 и D_9, за да го определиш D_{11};
6. Искористи ги D_9 и D_{11}, за да го определиш D_{12};
7. Искористи ги D_{10} и C_2, за да го определиш D_{13};
8. Искористи ги K_8 и D_{11}, за да го определиш D_{14};
9. Искористи ги D_{11} и D_{12}, за да го определиш D_{15};
10. Искористи ги D_{12} и D_{13}, за да го определиш D_{16};



(а) Прва фаза од нападот

(б) Втора фаза од нападот

Слика 4-14: Напад на π16-Cipher096 со две рунди

Графичкиот приказ на оваа процедура е даден на Слика 4-14б. Со повторување на постапката и за вториот енкриптирачки блок, при пресекот на добиените резултантни листи треба да се најдат парови за I_2, I_3 и I_4 кои имаат исти вредности. Повторно веројатноста за ова е 1, што значи дека на единствен начин може да се одгатни вредноста на состојбата CIS за функцијата π. Знаејќи ја CIS напаѓачот има можност да ја фалсификува пораката. Исто така со овој напад може да се разбие и клучот имајќи го во предвид фактот да не се користи тајниот параметар за нонсот - SMN. Во случај кога SMN е присутен во дизајнот како параметар, нападот е валиден само до откривање на CIS. Временската комплексност за да се изврши овој напад е 2^{72}, додека пак мемориската комплексност е 2^{69} бајти. На овој начин разбиена е предложената модификација на лесната варијанта на шифрувачот π-Cipher со две рунди и без употреба на SMN, објавена во трудот [69].

4.8 Нивоа на сигурност

Во ова секција ќе ги специфицираме сигурносните нивоа за сите варијанти на π-Cipher според параметрите кои се соодветни за секоја од нив и дадени во

Табела 2.1. За овие параметри, π-Cipher ги задоволува следниве сигурносни нивоа (изразени со log_2 за цена на напад), дадени во Табела 4.1.

Ниво	Pi16Cipher096v2 Битови на сигурност	Pi32Cipher128v2 Битови на сигурност	Pi64Cipher128v2 Битови на сигурност	Pi64Cipher256v2 Битови на сигурност
Доверливост на оригиналниот текст	96	128	128	256
Доверливост на тајниот нонс SMN	96	128	128	256
Интегритет на оригиналниот текст	96	128	128	256
Интегритет на поврзаните податоци	96	128	128	256
Интегритет на јавниот нонс PMN	96	128	128	256
Интегритет на тајниот нонс SMN	96	128	128	256
Доверливост на ориг. текст M, кога $(K, AD, (PMN, SMN_1))$ и $(K, AD, (PMN, SMN_2))$	96	128	128	256
Интегритет на ориг. текст M, кога $(K, AD, (PMN, SMN_1))$ и $(K, AD, (PMN, SMN_2))$	96	128	128	256

Табела 4.1: Листа на нивоа на сигурност кај сите варијанти на π-Cipher

Нивоата на сигурност за различните варијанти на π-Cipher соодветствуваат на сигурност отпорна на повторно враќање на клучот. Разгледувани се случаите кога нонсот е уникатна вредност (целосна сигурност), а исто така кога дел од нонсот се реискористува (средно ниво на робустност).

Во деталниот опис на дијанот на π-Cipher, напоменавме дека нонсот во овој случај е изграден од парот $N = (PMN, SMN)$. Не случајно е доделен вториот дел, SMN, тој игра значајна улога во сигурноста на целата шема. Доколку легитимната страна која го има клучот K двапати го искористи исиот нонс N за да енкриптира два различни парови од оригинална порака и поврзани податоци, (M_1, AD) и (M_2, AD) со истиот таен клуч K и иста вредност за AD, тогаш велиме дека доверливоста и интегритетот на пораката не се задоволени со шемата

π-Cipher. Значи може да заклучиме дека првите 6 цели од Табела 4.1, се задоволени под претпоставка дека параметарот PMN од нонсот е секогаш различен за два различни парови од оригинална порака и поврзани податоци, (M_1, AD) и (M_2, AD) со истиот таен клуч K.

Дополнително, π-Cipher може да обезбеди некое средно ниво на робустност, со употреба на различни вредности за параметарот SMN. Имено, кога легитимната страна која го има клучот K, искористи исти вредности за PMN, но различни вредности SMN_1 и SMN_2, за енкриптирање на два различни парови од оригинална порака и поврзани податоци, (M_1, AD) и (M_2, AD) со истиот таен клуч K и иста вредност за AD, тогаш велиме дека доверливоста и интегритетот на пораката се комплетно задоволени со шемата π-Cipher. Меѓутоа, во овој случај доверливоста и интегритетот на SMN_1 и SMN_2 не се задоволени.

Нашата претпоставка е дека сигурноста наспроти роденденскиот напад е 128 бита за π16-Cipher096v2, 256 бита за π32-Cipher128v2 и 512 бита за π64-Cipher128v2 и π64-Cipher256v2. Истата сигурноста важи и за погодување на тагот и тоа 128 бита за π16-Cipher096v2, 256 бита за π32-Cipher128v2 и 512 бита за π64-Cipher128v2 и π64-Cipher256v2.

Глава 5

Мод на работа на π-Cipher за уреди со ограничена меморија

Во оваа глава ќе дадеме детален опис на новиот мод на операции на шемата π-Cipher кој поддржува меѓу тагови. Со овој мод, овозможено е π-Cipher да изврши верификација на тагот за големи пораки дури и на уреди кои имаат ограничена меморија и без притоа да се направи ослободување на неверифициран дел од декриптираниот шифриран текст. Како и самата шема и овој мод е паралелен, поддржува онлајн енкрипција и декрипција и дава некое средно ниво на отпорност на шемата со реискористување на нонсот, како дополнителна робустна карактеристика на шифрувачот.

5.1 Општо за онлајн AEAD шемите

Концептот за онлајн автентикациска енкрипција, предложен во трудот [31] укажува на тоа дека енкрипцијата може да се изведе попатно, без дополнителни ресурси. Односно може да се каже дека при онлајн автентикациска енкрипција блокот од шифриран текст C_i, може да се пресмета без учество на блокот од оригиналната порака M_j, каде што $j > i$. Традиционалните AEAD шеми не го задоволуваат ова корисничко сценарио, најголем дел од нив се "offline"што значи дека уредот треба да нуди можност за зачувување на комплетниот шифриран

текст од фазата на енкрипција за да може да се произведе тагот. Но, не сите уреди имаат доволно голема меморија за да може да се зачува целата порака, па според тоа не може директно да се користат за обработка на големи пораки. Најголем дел од AEAD шемите базирани на блок-шифрувачи и сунѓер конструкции кои користат дуплекс мод, може да се користат за онлајн автентикациска енкрипција. Исто така голем дел од шифрувачите поднесени на натпреварот CAESAR го задоволуваат ова својство.

Ако процесот на енкрипција-потоа-автентикација се одвива во дадениот редослед на страната на испраќачот, на страната на примачот овој процес е преведен како верификација-потоа-декрипција. Начинот да се трансформираат верификацијата и декрипцијата на шифрираниот текст попатно станува многу покомплексен. Може да се спроведе на два различни начини. Првиот начин е со употреба на две поминувања на шифрираниот текст, односно се спроведува во две фази. Во првата фаза само тагот се верифицира, а после тоа се прави ослободување на декриптираните блокови и тоа онлајн. Овој пристап се користи кај уреди кои имаат ограничена меморија и не можат наеднаш да го соберат целиот шифриран текст во меморија. Пример на ваков тип на протокол кој го овозможува ова својство е Декриптирај-потоа-Маскирај (Decrypt-then-Mask, DTM) [31]. Вториот начин на сигурна онлајн верификација и декрипција на податоците е со користење на меѓу тагови. Во овој случај верификацијата и декрипцијата на блоковите се прави попатно без користење на некои специјални побарувања на меморија од уредите или дополнителни повици кон примитивите. Овој тип на AEAD со меѓу тагови за прв пат е дефиниран кај дуплекс модот на сунѓер конструкциите во [14]. Многу од кандидатите на натпреварот CAESAR за AEAD шифрувачи ја следат дуплекс конструкцијата како што се: ASCON [19], ICEPOLE [83], NORX [53], Keyak [38], PRIMATEs [29], π-Cipher [34] и STRIBOB [64], но голем дел од нив се секвенцијални и без опции за онлајн декрипција на податоците.

Во текот на натпреварот CAESAR, многу прашања се покренаа околу тоа дали шифрувачот е онлајн или не и дали е отпорен на повторување на нонсот или не. Во контекст на ова беше публикуван труд [89], во кој авторите Рогавеј и Шримптон даваат построга формална дефиниција за AEAD шема која е отпорна

на повторување на нонс. Според нив цената што се плаќа за да се креира ваков тип на AEAD шема е тоа што истата не може да биде онлајн. Подоцна, во трудот [43], Хоан и тимот даваат нови дефиниции за онлајн автентикациска енкрипција и ги именуваат како (корегирана) OAE1 и OAE2. Ако ги земеме овие дефиниции во предвид ниту еден од кандидатите на натпреварот CAESAR во исто време не може да биде онлајн и отпорен на повторување на нонсот. Ова се случува поради фактот што кога клучот е ист и нонсот ќе се повтори кај секоја онлајн AEAD шема не може да се одбегне пролевањето на најдолг заеднички префикс на блоковите на пораките.

Во текот на натпреварот неколку кандидати учесници базирани на блок-шифрувачи се промовираа како онлајн, со едно поминување и отпорни на повторување на нонс (COPA [27], ELmD [81] и POET [30]). Секој од нив се карактеризира со паралелизабилност и два повици кон основната крипто примитива во текот на процесот на енкрипција (во случајот на POET се користи еден повик кон блок шифрувач и два повици кон хеш функција). Уште повеќе, овие шифрувачи користат инверзна функција на примитивата за да се оствари декрипцијата на податоците. А според нивните документации може да се каже дека POET и ElmD во исто време поддржуваат и меѓу тагови.

Ние продуцираме нов мод на нашиот шифрувач, π-Cipher со меѓу тагови. Без да се случи ослободување на оригиналната порака без претходно да биде верифицирана, овој мод дозволува верификација на тагот за долги пораки дури и кај уреди кои имаат ограничена меморија. Според тоа можеме да кажеме дека овој мод се карактеризира со: паралелизам, онлајн енкрипција и декрипција и дава некое средно ниво на отпорност на шемата со реискористување на нонсот. Наспроти другите кандидати кои поддржуваат паралелизам и меѓу тагови (ElmD и POET), нашата шема нема потреба од користење на инверзна функција во фазата за декрипција, туку само со единствена имплементација на пермутациската функција обете енкрипција/автентикација и декрипција/верификација се изведуваат на онлајн начин.

5.2 Формална дефиниција

Во трудот [43] авторите имаат дадено комплетна дефиниција за сегментирана шема AEAD која содржи константно проширување τ на сегментот. Овој тип на AEAD шема е даден се со цел да се следат новите трендови со софтверската и хардверската индустрија. Имено, софтверските библиотеки треба да поддржуваат апликациско програмски интерфејси за онлајн енкрипција и декрипција. Во случајот на AEAD шемите, онлајн енкрипцијата е поддржана скоро од сите понови шифрувачи, но за да се достигне и декрипцијата да биде онлајн, треба верификацијата да се прави после секој блок или група од блокови (сегменти). Затоа авторите го означиле овој тип на AEAD шема како сегментирана AEAD шема. Формалната дефиниција е следна:

Дефиниција 11 (Сегментирана AEAD шема [43])**.** *Сегментирана AEAD шема е тројката* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *каде што просторот на клучот* $\mathcal{K}$ *е непразно множество со рамномерна дистрибуција на битовите и енкрипциски алгоритам* $\mathcal{E} = (\mathcal{E}.init, \mathcal{E}.next, \mathcal{E}.last)$ *и декрипциски алгоритам* $\mathcal{D} = (\mathcal{D}.init, \mathcal{D}.next, \mathcal{D}.last)$ *специфицирани со тројките од детерминистички алгоритми. Поврзани со* Π *се просторот на поврзани податоци* $\mathcal{A} \subseteq \{0,1\}^*$*, просторот за нонс* $\mathcal{N} \subseteq \{0,1\}^*$*, просторот на состојбите* $\mathcal{S}$ *и соодветното проширување* τ*. Овие параметри се појавуваат во формалните презентации на компонентите на* $\mathcal{E}$ *и* $\mathcal{D}$*:*

$$\begin{aligned}
&\mathcal{E}.init : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \to \mathcal{S} && \mathcal{D}.init : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \to \mathcal{S} \\
&\mathcal{E}.next : \mathcal{K} \times \mathcal{S} \times \{0,1\}^{**} \to \{0,1\}^{**} \times \mathcal{S} && \mathcal{D}.next : \mathcal{K} \times \mathcal{S} \times \{0,1\}^{**} \to (\{0,1\}^{**} \times \mathcal{S}) \\
&\mathcal{E}.last : \mathcal{K} \times \mathcal{S} \times \{0,1\}^{*} \to \{0,1\}^{*} && \mathcal{D}.last : \mathcal{K} \times \mathcal{S} \times \{0,1\}^{*} \to \{0,1\}^{*} \cup \perp,
\end{aligned}$$

каде $\{0,1\}^{**} = (\{0,1\}^*)^*$ *означува множество од сегмент-стрингови каде секоја компонента од сегмент-стрингот всушност претставува стринг сама за себе.*

Бројот на компоненти во сегмент-стрингот $\boldsymbol{M}$ е означена со $|\boldsymbol{M}| = m$, додека пак i-та компонента од $\boldsymbol{M}$, $i \in [1..m]$, е означена како $\boldsymbol{M}_i$. Да забележиме дека индексирањето започнува од 1. Значи, $\boldsymbol{M} = (M_1, M_2, \ldots M_m) \in \{0,1\}^{**}$. Овде,

$\boldsymbol{M}$ се трансформира во сегментиран шифриран текст $\boldsymbol{C} = (C_1, C_2, \ldots, C_m) = \mathcal{E}(K, N, A, \boldsymbol{M})$ каде што $|C_i| = |M_i| + \tau$ за сите $i \in [1..m]$ и каде енкрипцискиот алгоритам е даден со тројката од алгоритми како во Дефиниција 11.

π-Cipher $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е AEAD шема со таен дел од нонс - SMN и ја дефинираме во Секција 4.2 со Дефиниција 10. Оваа шема се потпира на пермутацискатa функција $\pi : \{0,1\}^b \to \{0,1\}^b$ и соодветно клуч - K, јавен дел од нонсот - PMN, поврзани податоци - AD, таен дел од нонсот - SMN, порака - M и шифриран текст - C. π-Cipher покрај тоа што е онлајн и паралелен, може да се искористи и во мод на операција каде проблемот со ослободување на непроверен оригинален текст е адресиран [4]. За да може прецизно и математички да го дефинираме овој мод на операција на π-Cipher ние природно ги споивме двете дефиниции за *AEAD со SMN* - Дефиниција 10 и *Сегментирана AEAD шема* - Дефиниција 11 во *Сегментирана AEAD шема со SMN* каде што автентикацискатa сегментација е постигната со пресметување на меѓу тагови. Со моделот на меѓу тагови, π-Cipher нуди сигурност наспроти напади насочени поединечно кон податочни блокови. Во овој модел напаѓачот има можност да испраќа пораки блок по блок на шифрувачот и да ги добие континуирано соодветните шифрирани блокови автентицирани.

Основната идеја е за оригинална порака $M \in \{0,1\}^{mn}$, π-Cipher да генерира шифриран текст $C \in \{0,1\}^{n+mn+m\tau+\tau}$, каде $m \geqslant 0$ е бројот на блокови на пораки/шифрирани текстови, $n \geqslant 0$ е должина на еден блок во битови и $\tau = k$ е должината на тагот во битови. Во овој случај шифрираниот текст има проширување од $n + m\tau + \tau$-битови, каде што n битовите соодветствуваат на енкриптираниот број SMN, $m\tau$ ги претставуваат таговите на секој енкриптиран блок M_i поединечно и τ е резервиран за финалниот таг. Фазата на процесирање на пораката останува непроменета, а исто така непроменето останува и пресметувањето на тагот. Меѓу таговите t_j се генерираат и ослободуваат после секој блок од пораката. Финалниот таг T се пресметува на крај од целата порака и се состои од збирот на таговите генерирани после секој блок на поврзаните податоци T', тагот генериран после фазата на SMN t_0 и сите ослободени меѓу тагови на блоковите

од пораката t_j.

$$T = T' \boxplus t_0 \boxplus_{j=1}^{m} t_j.$$

Енкрипциските и декрипциските функции може да се опишат на следниот начин:

$$\mathcal{E}(K, PMN, AD, SMN, M) \rightarrow (C, IT, T) \quad \text{and}$$
$$\mathcal{D}(K, PMN, AD, C, IT, T) \rightarrow (SMN, M) \cup \perp,$$

каде што $IT = (\underbrace{\lambda, \lambda, \ldots, \lambda}_{a+1}, t_1, t_2, \ldots, t_m)$ ја претставува секвенцата од меѓу тагови. Овде, со λ е обележан празен стринг во нашиот модел. Како што може да се забележи не се ослободуваат меѓу тагови од поврзаните податоци, ниту пак од SMN. Значи, вкупниот број на ослободени меѓу тагови е m (исто како и бројот на блокови на пораката). Должината на секој од меѓу таговите во битови е $|t_j| = k$ (како и должината на саиот клуч).

Формалнаата дефиниција на модот на операции на π-Cipher со меѓу тагови може формално да се претстави со следната дефиниција:

Дефиниција 12 (Сегментирана AEAD шема со SMN)**.** *π-Cipher $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е сегментирана AEAD шема со константно проширување на сегментот и таен дел од нонсот SMN, каде што просторот на клучот $\mathcal{K}$ е непразно множество со рамномерна дистрибуција на битовите и енкрипциски алгоритам $\mathcal{E} = (\mathcal{E}.init, \mathcal{E}.smn, \mathcal{E}.msg)$ и декрипциски алгоритам $\mathcal{D} = (\mathcal{D}.init, \mathcal{D}.smn, \mathcal{D}.msg)$ специфицирани со тројките од детерминистички алгоритми.*

Поврзани со π-Cipher се неговиот простор $\mathcal{A} \subseteq \{0,1\}^*$ на поврзани податоци AD, просторот $\mathcal{P} \subseteq \{0,1\}^*$ на јавниот дел од нонсот PMN, просторот $\mathcal{S}$ на тајниот дел од нонсот SMN, заедничката внатрешна состојба CIS со простор $\mathcal{IS}$ и проширувањето τ на сегментот. Овие параметри се појавуваат во формалните презентации на компонентите $\mathcal{E}$ и $\mathcal{D}$ како што следува:

$$\mathcal{E}.init : \mathcal{K} \times \mathcal{P} \times \mathcal{A} \to \mathcal{IS} \qquad \mathcal{D}.init : \mathcal{K} \times \mathcal{P} \times \mathcal{A} \to \mathcal{IS}$$
$$\mathcal{E}.smn : \mathcal{IS} \times \mathcal{S} \to \{0,1\}^* \times \mathcal{IS} \qquad \mathcal{D}.smn : \mathcal{IS} \times \mathcal{S} \to (\{0,1\}^* \times \mathcal{IS})$$
$$\mathcal{E}.msg : \mathcal{IS} \times \{0,1\}^{**} \to \{0,1\}^{**} \qquad \mathcal{D}.msg : \mathcal{IS} \times \{0,1\}^{**} \to \{0,1\}^{**} \cup \perp .$$

Algorithm 1: $\mathcal{E}(K, PMN, AD, SMN, \boldsymbol{M})$

1: $m \leftarrow |\boldsymbol{M}|$
2: **if** $m = 0$ **then**
3: **return** λ
4: **end if**
5: $(M_1, M_2, \ldots, M_m) \leftarrow \boldsymbol{M}$
6: $CIS \leftarrow \mathcal{E}.init(K, PMN, AD)$
7: $(C_0, CIS') \leftarrow \mathcal{E}.smn(CIS, SMN)$
8: **for** $i \leftarrow 1$ **to** m **do**
9: $C_i \leftarrow \mathcal{E}.msg(CIS', M_i)$
10: **end for**
11: **return** $(C_0, C_1, \ldots, C_m)$

Algorithm 2: $\mathcal{D}(K, PMN, AD, \boldsymbol{C})$

1: $m \leftarrow |\boldsymbol{C}| - 1$
2: **if** $m = 0$ **then**
3: **return** λ
4: **end if**
5: $(C_0, C_1, \ldots, C_m) \leftarrow \boldsymbol{C}$
6: $CIS \leftarrow \mathcal{D}.init(K, PMN, AD)$
7: $(SMN, CIS') \leftarrow \mathcal{D}.smn(CIS, C_0)$
8: **for** $i \leftarrow 1$ **to** m **do**
9: **if** $\mathcal{D}.msg(CIS', C_i) = \perp$ **then**
10: **if** $m = 1$ **then**
11: **return** λ
12: **else**
13: **return** $(M_1, \ldots, M_{i-1})$
14: **end if**
15: **else**
16: $M_i \leftarrow \mathcal{D}.msg(CIS', C_i)$
17: **end if**
18: **end for**
19: **return** $(M_1, \ldots, M_m)$

Слика 5-1: Алгоритми за енкрипција и декрипција на сегментирана AEAD шема со SMN за π-Cipher

Енкрипциискиот алгоритам (со автентикација) оперира над SMN и сегментираната оригиналана порака $\boldsymbol{M} = (M_1, M_2, \ldots, M_m) \in \{0,1\}^{**}$ за да произведе сегментиран шифриран текст $\boldsymbol{C} = (C_0, C_1, \ldots, C_m) = \mathcal{E}(K, PMN, AD, SMN, \boldsymbol{M})$. Постои соодветен декрипциски алгоритам $\mathcal{D}$ таков што $\mathcal{D}(K, PMN, AD, \boldsymbol{C}) =$

$(SMN, \boldsymbol{M})$ или $\perp$ ако декрипцискиот алгоритам заедно со верификацијата дадат погрешна вредност за некој од сегментите. Алгоритмите се дефинирани на Слика 5-1.

5.3 Доказ за сигурност

Доказот за сигурноста на π-Cipher со меѓу тагови комплетно се базира на самиот доказ на π-Cipher даден во Глава 4. Во овој случај треба да се обрати внимание на меѓу таговите поради фактот што сега тие се познати и предмет на манипулација на напаѓачот за разлика од претходно каде што беа непознати.

Приватност Определуваме горна граница за предноста на напаѓачот кој може да направи разлика помеѓу излезот продуциран од предложената шема и случаен пророк во идеален пермутациски модел. Според тоа приватноста на шемата π-Cipher со меѓу тагови е дадена со следната теорема:

Теорема 4. *Нека $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е предложената шема π-Cipher со меѓу тагови, каде што основната криптографска примитива, пермутациската функција π е заменета со идеална пермутација p која што оперира над b битови. Значи,*

$$\boldsymbol{Adv}_{\Pi}^{priv}(q_p, q_{\mathcal{E}}, \lambda_{\mathcal{E}}) \leqslant \frac{2(q_p + \sigma_{\mathcal{E}})^2}{2^b} + \frac{q_p + \sigma_{\mathcal{E}}}{2^k} + \frac{q_p r}{2^c} + \frac{q_{\mathcal{E}} a}{2^{b/2}} + \sqrt{\frac{8e\sigma_{\mathcal{E}} q_p}{2^b}},$$

каде што $\sigma_{\mathcal{E}}$ е дефинирана во (4.1), q_p е максималниот број на прашања до идеалната пермутација $p^{\pm}$, $q_{\mathcal{E}}$ е максималниот број на енкрипциски прашања со вкупна должина од $\lambda_{\mathcal{E}}$ блокови и a е максималниот број на блокови за поврзани податоци.

Доказот е адекватен и без промени со тој што беше даден во Глава 4, Секција 4.4 за стандардната шема π-Cipher.

Автентичност Ја анализираме сигуроста на автентичност на таговите продуцирани со предложената шема. Да се направи фалсификат на AEAD шема

се дефинира како способност на напаѓачот $\mathcal{A}$ да генерира валидна торка (P, A, C_i, T_i) без притоа да постави директно прашање до енкрипциското пророк со параметри $\mathcal{E}(P, A, S, M_i)$ кои резултираат со (C_i, T_i). Според тоа автентичноста на шемата π-Cipher со меѓу тагови е дадена со следната теорема:

Теорема 5. *Нека $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ е предложената шема π-Cipher со меѓу тагови, каде што основната криптографска примитива, пермутациската функција π е заменета со идеална пермутација p која што оперира над b битови. Значи,*

$$\boldsymbol{Adv}_{\Pi}^{auth}(\mathcal{A}) \leqslant \frac{(q_p + \sigma_\varepsilon + \sigma_\mathcal{D})^2}{2^{b+1}} + \frac{\sigma_\mathcal{D}(q_p + \sigma_\varepsilon + \sigma_\mathcal{D})}{2^c} + \frac{q_p r}{2^c} + \frac{a(q_\varepsilon + q_\mathcal{D})}{2^{b/2}} + \frac{q_p + \sigma_\varepsilon + \sigma_\mathcal{D}}{2^k} + \frac{m^2 q_\mathcal{D}^2 + m q_\mathcal{D}}{2^k} + \sqrt{\frac{8e\sigma_\varepsilon q_p}{2^b}},$$

каде што σ_ε и $\sigma_\mathcal{D}$ се дефинирани во (4.1), q_p е максималниот број на прашања до идеалната пермутација $p^{\pm}$, q_ε е максималниот број на енкриптирачки прашања со вкупна должина од λ_ε блокови, $q_\mathcal{D}$ е максималниот број на декриптирачки прашања со вкупна должина од $\lambda_\mathcal{D}$ блокови и a е максималниот број на блокови за поврзани податоци.

Теоремата за автентичност на π-Cipher со меѓу тагови се разликува од теоремата за автентичност на стандардната шема π-Cipher по тоа што првата има повисоки граници за безбедност поради фактот што сега напаѓачот има поголема слобода и може да манипулира со меѓу таговите. Односно границата е намалена за вредноста $\frac{m^2 q_\mathcal{D}^2 + m q_\mathcal{D}}{2^k}$.

Имено, во случајот на π-Cipher со меѓу тагови, за сите декрипциски прашања $q_\mathcal{D}$ напаѓачот го ограничуваме на:

$$\mathbf{Pr}(\mathcal{A}^{f^{\pm}, \mathcal{E}_K, \mathcal{D}_K} \text{ forges} | \neg \text{event}) \leqslant \frac{m q_\mathcal{D}}{2^k}.$$

каде што веројатноста на напаѓачот да погоди еден од меѓу таговите во j-то декрипциско прашање е $m/2^k$.

Кога го имаме настанот $\mathcal{D}_{hit}$, во потслучајот (2), напаѓачот пробува да постави прашање така што ќе го повтори парот (PMN, AD), а C останува различно. Во овој случај после секој блок на пораката, меѓу таг се генерира и истиот може да се случи да се погоди со вредноста на некоја постара состојба. Веројатноста да се погоди таг t_i во j-то декрипциско прашање со некој постар генериран таг може да се случи во веројтност од $1/2^k$. А за сите такви меѓу тагови и декрипциски прашања можности има $\binom{mq_{\mathcal{D}}}{2}$ и веројатноста е ограничена на $\frac{m^2 q_{\mathcal{D}}^2}{2^k}$.

Глава 6

Дополнителни карактеристики на π-Cipher

Во оваа глава ќе дадеме детален опис на карактеристиките на шифрувачот π-Cipher со кои тој се издвојува од другите кандидати на натпреварот CAESAR. При дизајнот на шифрувачот параметарот N (кој означува број на делчиња на која е поделена внатрешната состојна на пермутацискатa функција π) беше дефиниран иницијално со вредност 4, но со таа забелешка што истиот може да се менува. Оваа предност е искористена за да може да се креираат енкрипциски блокови со произволна должина. Исто така кога се работи со големи податоци, скоро секогаш алгоритми со инкрементална функција се добредојдени. Предноста на операцијата собирање по компоненти (за генерирање на тагот) кај π-Cipher е искористена за да се дефинира нов инкрементален мод на шифрувачот.

6.1 Блокови со произволно голема должина

π-Cipher е дизјаниран да може да биде прилагодлив за различни должини на зборови, различни нивоа на сигурност и различни должини на блоковите. За разлика од шемите кои се базираат на блок шифрувачи, каде големината на блокот е фиксна и не може да се менува, многу важна карактеристика на π функцијата е тоа што нејзината состојба може произволно да се издолжува.

Автентикацискатa енкрипција на податоците кои се складирани на различни медиуми има свои специфичности и се разликува од автентикацискатa енкрипција на податоците кои се пренесуваат. Една забележителна разлика е тоа што алгоритмите за автентикациска сигурност кои се користат кај складиштата на податоци треба да овозможат независна енкрипција и декрипција на поголеми делови од податоци. Овие делови најчесто се претставени со сектори на дискот. Старите хард дискови имале сектори од по 512 бајти. Со развојот на технологијата овие бројки драстично се зголемиле и достигнале до 4096 бајти (4KB) големина на диск сектор со помош на новиот напреден формат AF [50]. Новите AF HDD уреди за складирање на податоци може да читаат од и запишуваат на физичкиот сектор на хард дискот податоци со големина од 4KB.

Спротивно на стандардните хард дискови, новите дискови SSD (Solid State Drive) се состојат од полупроводливи мемориски блокови и не содржат механички делови (како во случајот со секторите). Најмалата единица кај SSD дисковите е страна и таа може да биде со големина од 2KB, 4KB, 8KB, 16KB итн. Притоа. не е овозможено да се прочита помалку од една страница наеднаш [37]. Неколку страници од SSD, формираат блок. На пример, Samsung SSD 840 EVO има блокови од 2MB каде секој блок се состои од 256 страници, секоја со големина од 8KB.

Користење на енкрипциски алгоритам кој ќе може да енкриптира еден сектор (една страница) како еден блок од порака се смета за предност. Од оваа перспектива, ние може да ја искористиме многу ефикасно 64-битната варијанта на π-Cipher, со подесување на параметарот N да може да ги задоволи потребите на произволно големи блокови кои се совпаѓаат со големина на сектор или страница на соодветен диск [35]. На пример, ако $N = 256$ (наспроти предложената вредност $N = 4$), ние можеме да енкриптираме еден цел сектор од 4KB како еден блок. Идентификацискиот број на секторот може да се искористи како дел од бројачот кој се користи во π-Cipher кога се процесира секој блок, додека пак автентикацискиот таг кој ќе се произведе за тој сектор, може да се зачува во соодветна податочна структура поврзана со дадениот диск сектор. Поради паралелизмот на π-Cipher сите вакви диск сектори може да се процесираат паралелно

и независно еден од друг.

За енкриптирање на старите стандардни хард дискови, каде големината на диск секторот е 512B, ратата на π-Cipher би требало да биде 512B и параметарот $N = 32$ за да може еден сектор да се испроцесира наеднаш како еден блок од податоци.

Имајќи го сето ова во предвид, ние предлагаме 5 различни инстанци на π64-Cipher256 со параметри N кои соодветствуваат на големините на постарите и модерните диск сектори. Предложените варијанти се дадени во Табела 6.1. Да забележиме дека бројот на рунди тука е 2, со оглед на големата состојба на функцијата π овој параметар од 2 рунди е доволен за да се обезбеди приватност и интегритет на податоците.

	$klen$ (битови)	PMN (битови)	SMN (битови)	Ратата во бајти	N	Tag T (битови)	R
широк блок од 512B	256	512	0	512	32	256	2
широк блок од 2KB	256	512	0	2048	128	256	2
широк блок од 4KB	256	512	0	4096	256	256	2
широк блок од 8KB	256	512	0	8192	512	256	2
широк блок од 16KB	256	512	0	16384	1024	256	2

Табела 6.1: Карактеристика на широки блокови за π64-Cipher256

6.2 Инкрементално својство

Еден од круцијалните фактори за обезбедување на брза и сигурна енкрипција на големи податоци е употребата на инкременталните крипто примитиви. Кога станува збор за податоци во мирување, сместени на некој податочен облак или складиште каде големината на датотеките е од неколку мега бајти, до стотици гига бајти, инкременталните крипто примитиви даваат брзина која значително се разликува од онаа кај стандардните примитиви. Се разбира дека нашиот модерен свет полека влегува во ерата каде што брзината на пресметување на операциите за ажурирање игра значајна улога и наоѓа се поголема практична примена. Како прво, цената за податочните складишта, не е повеќе проблем (на пр. [21]). Второ, според последните извештаи, глобалниот интернет сообраќај до крајот на оваа

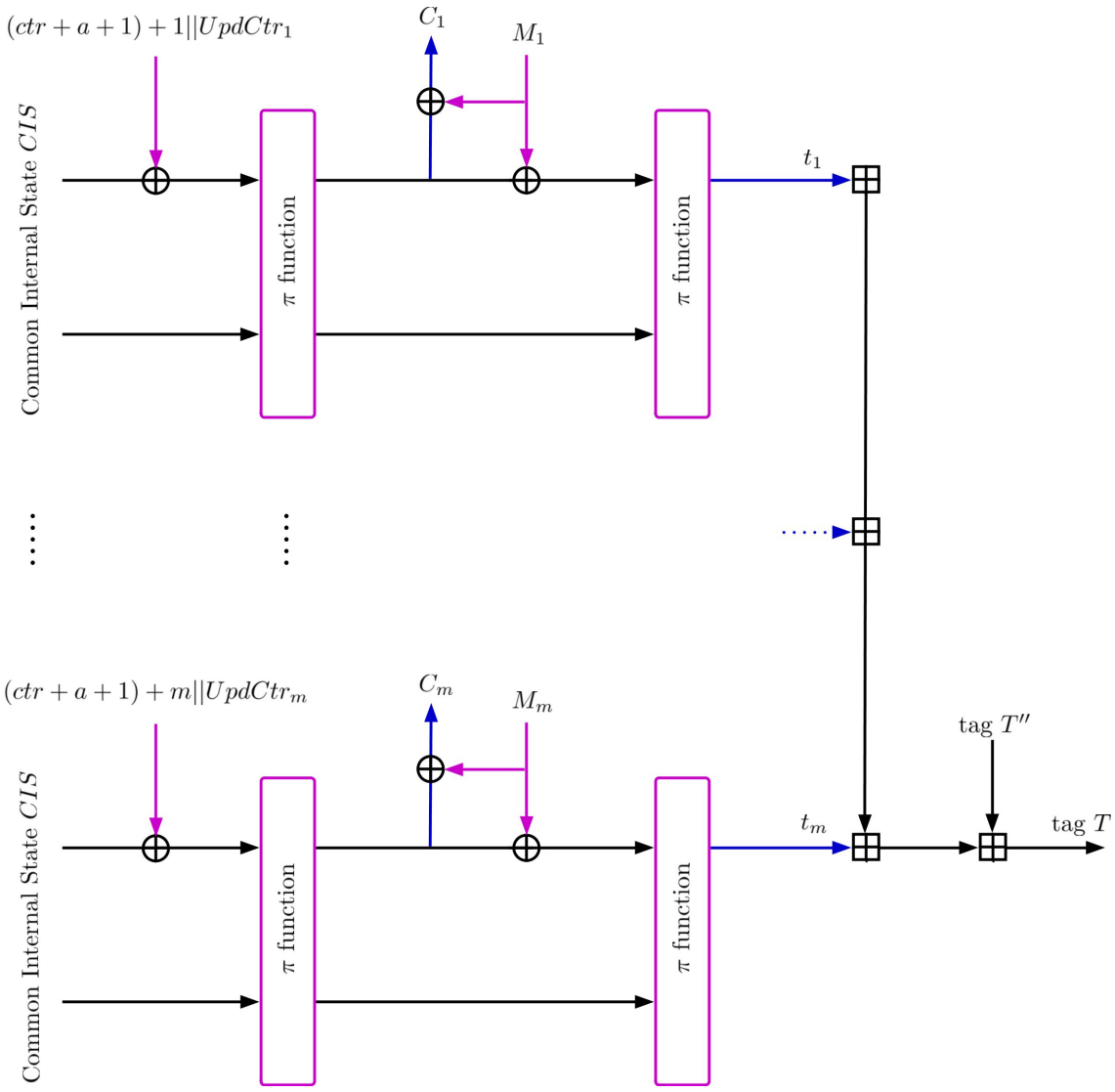
година ќе го надмине зета бајт прагот и се очекува да достигне 2.3 зета бајти по година во 2020 година [20]. Исто така според друго истражување се очекува до крајот на 2020 година дигиталниот универзум да достигне број до 44 зета бајти [94]. Според тоа степенот на податоците кои се обработуваат секојдневно од различни сервиси во облак, сензорски мрежи, дистрибуирани складишни системи, дигитални сервиси итн., веќе побарува нови решенија кои би ја користеле инкременталната парадигма.

Во суштина, идејата за инкременталната криптографија, е тоа што ако некој крипто алгоритам е применет на еден документ и овој документ е променет во само еден дел, тогаш би требало да можеме лесно да го ажурираме резултатот од алгоритамот за модифицираниот документ, без притоа истиот да го применуваме на целиот документ од ново. Идејата за инкеременталната криптографија е поставена од страна на Беларе, Голдрих и Голдвасер во трудот [9] и подобрена подоцна во [11]. Идејата за инкременталното својство на π-Cipher, потекнува од трудот [70] за инкременталноста на хеш функции со користење на операцијата собирање по компоненти во комутативна група $(\mathbb{Z}_{2^\omega}, \boxplus_\omega)$.

Според тоа ние дадовме алгоритам со кој π-Cipher се трансформира во сигурен инкрементален автентикациско енкрипциски шифрувач. Овој алгоритам доаѓа со дополнителни 64 бита на секој блок од пораката, кои се на некој начин цената која се плаќа за да се обезбеди ова својство. Дополнителните битови се претставени како бројач, кој ќе чува историја на сите ажурирања кои се случиле над тој блок.

По дефиниција големините на блоковите на π-Cipher се 128, 256 и 512 бита. Додавањето на 64 бита на секој од блоковите би значело многу, зошто претставено во проценти е $50\%, 25\%12.5\%$ од големината на блокот. Имајќи го во предвид претходното својство во Секција 6.1, каде се опишува дека π-Cipher може да поддржи блокови со произволна големина, за дефинираните вредности на блокови 512B, 2KB, 4KB, 8KB и 16KB, дополнителните битови за инкременталноста би значеле 1.5%, 0.39%, 0.19%, 0.09% и 0.04% од големината на блокот. За уште поголеми блокови, битови за бројачот би станале и незабележителни.

Да се напомене дека кога се користи π-Cipher во инкрементален мод, можни



Слика 6-1: Процесирање на оригиналната порака M во m паралелни блокови кај инкременталниот мод на π-Cipher. Тука $UpdCtr_i$ е бројачот за ажурирање.

се две сценарија и тоа: корисникот не сака да ги чува однапред пресметаните вредности за заедничката внатрешна состојба CIS и бројачот $ctr + a + 1$ или ги чува истите како тајни вредности. Доколку е избрано првото сценарио при секој инкрементален повик на π-Cipher, корисникот мора да ја изврши повторно иницијализациската фаза со тајниот клуч K и јавниот дел од нонсот PMN (повторувајќи го истиот). Наредно мора да се изврши фазата за автентикација на поврзаните податоци и енкриптирање на тајниот дел од нонсот SMN (повто-

рувајќи го истиот). Откако ќе се пресмета CIS и знае вредноста за $ctr + a + 1$, се врши енкрипција и автентикација само на ажурираниот блок M_i со употреба на бројачот за ажурирање $UpdCtr_i$, кој сега ја зголемува својата вредност за еден пред да се изврши енкрипцијата.

Алгоритам 1 - Операција на ажурирање со инкрементален таг
Влез. Заедничката интерна состојба CIS, $ctr + a + 1$, индекс за блокот i, стариот блок M_i, бројачот за ажурирање $UpdCtr_i$, новиот блок M'_i и старата вредност на тагот T.
Излез. Нов шифриран текст C_i и нов таг T.
1. За стариот блок M_i, пресметај: $IS \leftarrow \pi(CIS_{rate} \oplus ((ctr + a + 1) + i \parallel UpdCtr_i) \parallel\parallel\parallel CIS_{capacity})$; $C_i \leftarrow M_i \bigoplus IS_{rate}$; $t_i \leftarrow \pi(C_i \parallel\parallel\parallel IS_{capacity})_{rate}$; 2. Зголеми ја вредноста на бројачот за ажурирање $UpdCtr_i \leftarrow UpdCtr_i + 1$ 3. За новиот блок M'_i, пресметај: $IS \leftarrow \pi(CIS_{rate} \oplus ((ctr + a + 1) + i \parallel UpdCtr_i) \parallel\parallel\parallel CIS_{capacity})$; $C'_i \leftarrow M'_i \bigoplus IS_{rate}$; $t'_i \leftarrow \pi(C'_i \parallel\parallel\parallel IS_{capacity})_{rate}$; 4. Комбинирај ги T, t_i и t'_i преку групниот оператор за комбинирање $\boxplus_d$ за да се добие финалната вредност на автентикацискиот таг $T' = T \boxminus_d t_i \boxplus_d t'_i$; 5. Замени: $C_i \leftarrow C'_i$ и $T \leftarrow T'$; 6. Излез C_i и T.

Табела 6.2: Инкрементално ажурирање на автентикациска енкрипција со π-Cipher.

Како што може да се забележи, кога се применува π-Cipher во инкрементален мод се случува повторување на нонсот како пар (PMN, SMN). За да се избегне сценариото на повторна употреба на нонсот кај шифрувачи за автентикациска енкрипција, сега со употребата на бројачот за ажурирање $UpdCtr$ се добива нов вектор $(PMN, SMN, UpdCtr_i)$ кој формира единствен нонс за секој блок M_i. Процедурата за ажурирање на еден блок од порака со помош на инкрементална автентикациска енкрипција е опишана со Алгоритам 1. Притоа во чекорите 1 и 3

се гледа како е вметнат 64 битниот бројач за ажурирање $UpdCtr_i$, односно како тој се конкатенира со бројачот на блоковите и заедно како 128 битна вредност се инектираат во внатрешната состојба на пермутациската функција π. Графичка репрезентација на енкрипциската фаза кај инкременталниот мод на π-Cipher е дадена на Слика 6-1.

Глава 7

Споредбена анализа

Во оваа глава ќе ја споредеме нашата шема со веќе постоечкиот стандард за AEAD, AES-GCM [78]. Исто така ќе дадеме и детална табела на споредба на карактеристики со кои се одликуваат дизајните за AEAD на натпреварот CAESAR. Како споредбена анализа ќе ја претставиме и анализата на перформанси, која вклучува споредба на софтверски и хардверски решенија на соодветните шеми.

7.1 Споредба на функционални и сигурносни карактеристики

Една од главните цели на натпреварот за автентикациска енкрипција CAESAR, е да се најдат нови шифрувачи за автентикациска енкрипција кои ќе нудат предности над веќе постоечкиот стандард AES-GCM [78]. AES-GCM е шифрувач за автентикациска енкрипција со поврзани податоци кој што се базира на GCM модот на операции за блоковски шифрувачи, каде како блок шифрувач е искористен AES. Овој шифрувач се користи во протоколот TLS [25]. Избран е поради тоа што е целосно паралелизабилен, ефикасен и докажливо сигурен. Покрај добрите перформанси кои ги прикажува во софтвер, хардверските имплементации може да постигнат висока брзина, па истиот да се користи во апликации кои бараат голема пропустност на податоци. AES-GCM се користи како мод и во протоколите IPsec ESP [6] и 802.1AE Media Access Control Security.

Секој нов дизајн за AEAD покрај тоа што треба да ги исполнува скоро сите критериуми од AES-GCM, треба да понуди нешто ново со што ќе се издвои. За таа цел направивме споредба на нашиот кандидат π-Cipher со веќе постоечкиот стандард AES-GCM.

π-Cipher ги има следните предности над AES-GCM:

- **Реискористување на нонс**. AES-GCM е комплетно несигурен кога станува збор за повторување на нонсот. Во нашиот случај, со постоењето на тајниот дел од нонсот - SMN, π-Cipher обезбедува некое средно ниво на сигурност, поради тоа што ако јавниот дел од нонсот - PMN се повтори, а SMN е секогаш нова вредност, тогаш се задоволени во целост приватноста и интегритетот на пораката.

- **Можност за меѓу тагови**. π-Cipher има мод на операција со меѓу тагови кој овозможува работа на шифрувачот дури и на уреди со ограничена меморија без притоа да се случи ослободување на дешифрираната порака без да биде верифицирана.

- **Една крипто примитива**. Кај AES-GCM, блок шифрувачот AES е искористен за енкрипција на податоците, додека пак интегритетот е постигнат со користење на полиномно базирана хеш функција. Во нашиот случај π-Cipher обезбедува приватност и интегритет на податоците со користење на само една примитива, π функцијата.

- **Отсуство на слаби клучеви**. До сега не е направен напад на π-Cipher со слаби клучеви како што тоа беше случај на AES-GCM во [86]. π-Cipher воопшто не е подложен на овој тип на напад, поради тоа што не користи алгоритам за генерирање и распределување на подклучеви.

- **Променлива должина на тагот**. π-Cipher користи однапред определена должина на тагот која што е еднаква со должината на самиот клуч. Меѓутоа должината на тагот може да се манува во зависност од примената со максимална должина колку што е ратата на пермутациската функција.

Треба да се напомене и фактот дека должината на ратата на пермутациската функција може да се менува во зависност од потребата.

- **Блокови со променлива должина**. Благодарение на конструкцијата на π-Cipher, тој може да поддржи блокови со променлива должина на пораката во зависност од големината на параметарот N. За разлика од него, кај AES-GCM блоковите на пораката се со фиксна должина од 128 бита и не може да се променат.

- **Разноликост**. π-Cipher не се базира на AES инструкциското множество, додавајќи притоа разноликост во криптографските алгоритми.

π-Cipher ги има следните недостатоци спрема AES-GCM:

- **Асинхроно процесирање на поврзаните податоци и пораката**. AES-GCM дозволува пораката да се енкриптира пред или паралелно со фазата на процесирање на поврзаните податоци. Во случајот на π-Cipher, резултатот од фазата на процесирање на поврзаните податоци се користи за ажурирање на состојбата на функцијата π во фазата на енкрипција. Мешутоа во праксата големината на поврзаните податоци не надминува големина од 256 бита, па според тоа синхроно процесирање кај π-Cipher не влијае на перформансите на шемата.

Направени се неколку независни анализи на шифрувачите - кандидати на натпреварот CAESAR [1], [98], [5]. Целта на овие анализи е да се даде генерална слика за сите шеми натпреварувачи со нивните предности и недостатоци. Најдетален преглед на функционалните и сигурносните карактеристики на сите кандидати на натпреварот е даден во трудот [1]. Ние ќе ги издвоиме само кандидатите кои продолжуваат во третиот круг заедно со нашата шема π-Cipher иако не успеавме да продолжиме понатака во натпреварувањето. Комплетната анализа е дадена во Табела 7.1.

Анализата е направена според следните функционални и сигурносни карактеристики:

Паралелизам. За една крипто примитива велиме дека е паралелна, доколу при операцијата енкрипција/декрипција процесирањето на i-от блок не зависи од

излезот на j-от блок, за секој индекс $i \neq j$. Притоа операциите енкрипција и декрипција се разделени при разгледувањето на ова својство и се евалуираат посебно.

Онлајн. Еден шифрувач е онлајн ако енкрипцијата на i-от влезен блок M_i зависи само од претходно процесираните блокови $M_1, \ldots, M_{i-1}$. Доколку ова својство не е исполнето, шифрувачите се нарекуваат "offline"или шеми со две поминувања.

Без инверзна функција. AE шемите кои користат само една функција за да обезбедат енкрипција и декрипција на податоците, заштедуваат многу мемориски простор и ресурси. Таквите шеми се наречени "inverse-free"и не го користат стандардниот модул од крипто примитивата за декрипција на податоците.

Меѓу тагови. Ова својство на AEAD шемите со посебен осврт на π-Cipher го опишавме детално во Глава 5. Идејата тука е да се открие неправилност во шифрираните блокови без да се чека да се испроцесира целата порака. Ова својство е важно кога се работи со долги пораки.

Доказ за сигурност. За една AEAD шема велиме дека е сигурна ако истата истовремено обезбедува приватност и интегритет на пораката. Одосно во јазикот на докажливата сигурност, дали соодветната AEAD шема задоволува IND-CPA и INT-CTXT сигурност во услови кога основната крипто примитива е заменета со идеална случајна функција/пермутација.

Реискористен нонс. Кај енкрипциските шеми кои се базирани на нонс, повторувањето на нонсот може да се случи во пракса и притоа да се компромитира сигурноста на шемата. Комплетна дефиниција за автентикациска енкрипција која што е отпорна на повторување на нонс, дадена е во [89] и според неа, цената за да една шема биде комплетно сигурна на реискористување на нонсот, се балансира со перформансите, односно истата станува далеку понеефикасна и побарува две поминувања над податоците.

Исто така во дадената анализа наведен е и типот на конструкција на која се базира соодветната AEAD шема на кандидатите - натпреварувачи. Најголем дел од нив се базирани на блок шифрувачи, особено блок шифрувачот AES, потоа следуваат шемите кои користат сунѓер конструкции во комбинација со некоја пермутациска функција и дел од нив се базирани на познати проточни шифру-

вачи и специјално дизајнирани шеми. Оние шеми кои користат блок шифрувачи за енкрипција на блоковите, користат и специјално дефиниран мод на операции кој во суштина ја дефинира сигурната трансформација на податоците поголеми од еден блок. Некои од овие модови ги споменавме во Глава 1. Тука ќе ги дадеме само имињата на модовите кои се користат во соодветните шеми заедно со нивните референци, без притоа истите да се објаснуваат. Листата на модови на операции кои се користат во кандидатите во третиот циклус на натпреварот CAESAR се: **CFB** Ciphertext feedback mode [75], **EME** Encrypt-Mix-Encrypt mode [41], **OFB** Output feedback mode [75], **OTR** Two-branch two-round Feistel [72], **XEX** XOR-encrypt-XOR [88].

Според дадената анализа во Табела 7.1, јасно се гледа дека π-Cipher во ниту една карактеристика не изостанува зад кандидатите кои го продолжија натпреварувањето во третиот циклус на натпреварот.

7.2 Споредба на перформанси

Софтверските перформанси на криптографските шеми се значаен фактор во одлуката да се вклучи дадената примитива во реалните протоколи како што се: TLS, SSH или IPSec. Едно од главните барања за влез на натпреварот CAESAR беше сите кандидати да имаат имплементирано референтни софтверски решенија од нивните алгоритми во програмскиот јазик C. Во текот на натпреварот сите кандидати ги оптимизираа своите решенија трудејќи се да постигнат што е можно подобри перформанси. Во правилата на натпреварот стои дека кандидатите кои ќе влезат во вториот круг, мора да приложат и хардверска имплементација на своите шеми.

π-Cipher беше дизајниран да може добро да се однесува во софтвер и хардвер. Начинот на дизајн на главната компонента како и бројот на рунди беа сведени на минимално дозволено сигурносно ниво, за да може да се обезбедат подобри перформанси.

Во продолжение ќе дадеме анализа на генералните перформанси на π-Cipher како што се: број на операции, меморија, паралелизам и софтверска и хардверска

Конструкција	Кандидат	Дизајн	Карактеристики					Сигурност	
			Паралелна Енк/Дек	Онлајн	Без инверзна функ.	Меѓу тагови	Инкременталност	Доказ за сигурност	Реискористен нонс
базиран на проточен шифрувач	ACORN [45]	LFSR	•/•	•	•	-	-	-	-
специјален дизајн	AEGIS [46]	AES	•/–	•	•	-	-	-	-
базиран на блок шифрувач	AES-JAMBU [48]	OFB	- / -	•	•	-	-	-	•
базиран на блок шифрувач	AES-OTR [57]	OTR	•/•	•	•	-	•	•	-
базиран на блок шифрувач AES	AEZ [99]	OTR	•/•	-	•	-	•	•	•
базиран на сунѓер конструкции	Ascon [19]	SPN, Duplex	- / -	•	•	-	-	•	•
базиран на блок шифрувач AES	CLOC [97]	CFB	- / -	•	•	-	-	•	-
базиран на блок шифрувач AES	SILC [97]	CFB	–/•	•	•	-	-	•	-
базиран на блок шифрувач AES	COLM [28]	EME	•/•	•	•	•	•	•	•
базиран на блок шифрувач AES	Deoxys [54]	EME	- / -	•	-	-	-	•	•
базиран на сунѓер конструкција	Ketje [39]	Duplex	- / -	•	•	•	-	•	-
базиран на сунѓер конструкција	Keyak [38]	Duplex	•/•	•	•	•		•	-
специјален дизајн	MORUS [47]	LRX	- / -	•	•	-	-	-	-
базиран на сунѓер конструкција	NORX [53]	LRX, Duplex	•/•	•	•	-	-	•	-
базиран на блок шифрувач AES	OCB [96]	XEX	•/•	•	-	-	-	•	-
специјален дизајн	Tiaoxin [51]	AES рунда	•/•	•	•	-	-	-	-
базиран на сунѓер конструкција	π-Cipher [22]	ARX, Duplex	•/•	•	•	•	•	•	•

Табела 7.1: Функционални и сигурносни карактеристики на кандидатите во втората рунда на натпреварот CAESAR

имплементација.

7.2.1 Број на операции

Табела 7.2 го прикажува бројот на операции потребни на главната компонента на π-Cipher, функцијата π. Во основа како што и наведовме во претходните глави, операциите на шифрувачот π-Cipher се сведуваат на трите основни операции на ARX диазјните и тоа: собирање по компоненти (ADD), ротација во лево за r бита (ROTL) и логичката операција исклучиво или (XOR).

	AND	ROTL	XOR	Вкупно
операција $*$	28	8	16	52
1 рунда од π функ.	204	64	128	396
π функ.	612	196	384	1192
еден блок од порака [1]	1226	392	770	2388

Табела 7.2: Преглед на бројот на операции за π-Cipher

7.2.2 Меморија

Во зависност од тоа која верзија на π-Cipher е употребена, потребни се различни побарувања од интерна меморија. За секоја од верзиите потребни се 8 константи за извршување на операцијата $*$ и уште 6 рундовски константи за извршување на функцијата π. Константите вкупно побаруваат $8 \times \omega$ бита и $6 \times 4 \times \omega$ бита, за $\omega = 16, 32, 64$ или бкупно 64 бајти за π16-Cipher, 128 бајти за π32-Cipher и 256 бајти за π64-Cipher.

Покрај константите потребно е да се чува и вредноста на заедничката интерна состојба CIS на пермутациската функција π. Односно, тоа побарува за $N = 4$, 32 бајти за π16-Cipher, 64 бајти за π32-Cipher и 128 бајти за π64-Cipher дополнителна меморија. И на крај потребно е да се резервира и мемориски простор за

[1] Под блок од порака се подразбира енкрипција на пораката и нејзина автентикација без претходно да се земе во предвид иницијализациската фаза.

генерирање на тагот, а тој изнесува 16 бајти за π16-Cipher, 32 бајти за π32-Cipher и 64 бајти за π64-Cipher.

7.2.3 Паралелизам

Главната компонента, π функцијата по својата природа не е паралелна и не може тоа да се постигне заради тоа што се користени два различни типови на квазигрупни трансформации. Доколку беше искористен еден тип на квазигрупна трансформација би можело во внатрешноста на функцијата да се дефинира проточен паралелизам. Но, поради тоа, π-Cipher нуди поголемо ниво на паралелизам со користење на современите процесорски системи. Дизајнот на нашата шема е таков што независно може да се обработуваат онолку блокови колку што на располагање имаме јадра на процесорот. Така на пример, на денешните графички процесорски единици, каде што бројот на проточни процесори може да достигне и до неколку илјади [32], забрзувањето на времето на извршување, а со тоа и на пропустноста драстично се зголемува дури и до фактор 10 [92] [80].

7.2.4 Софтвер

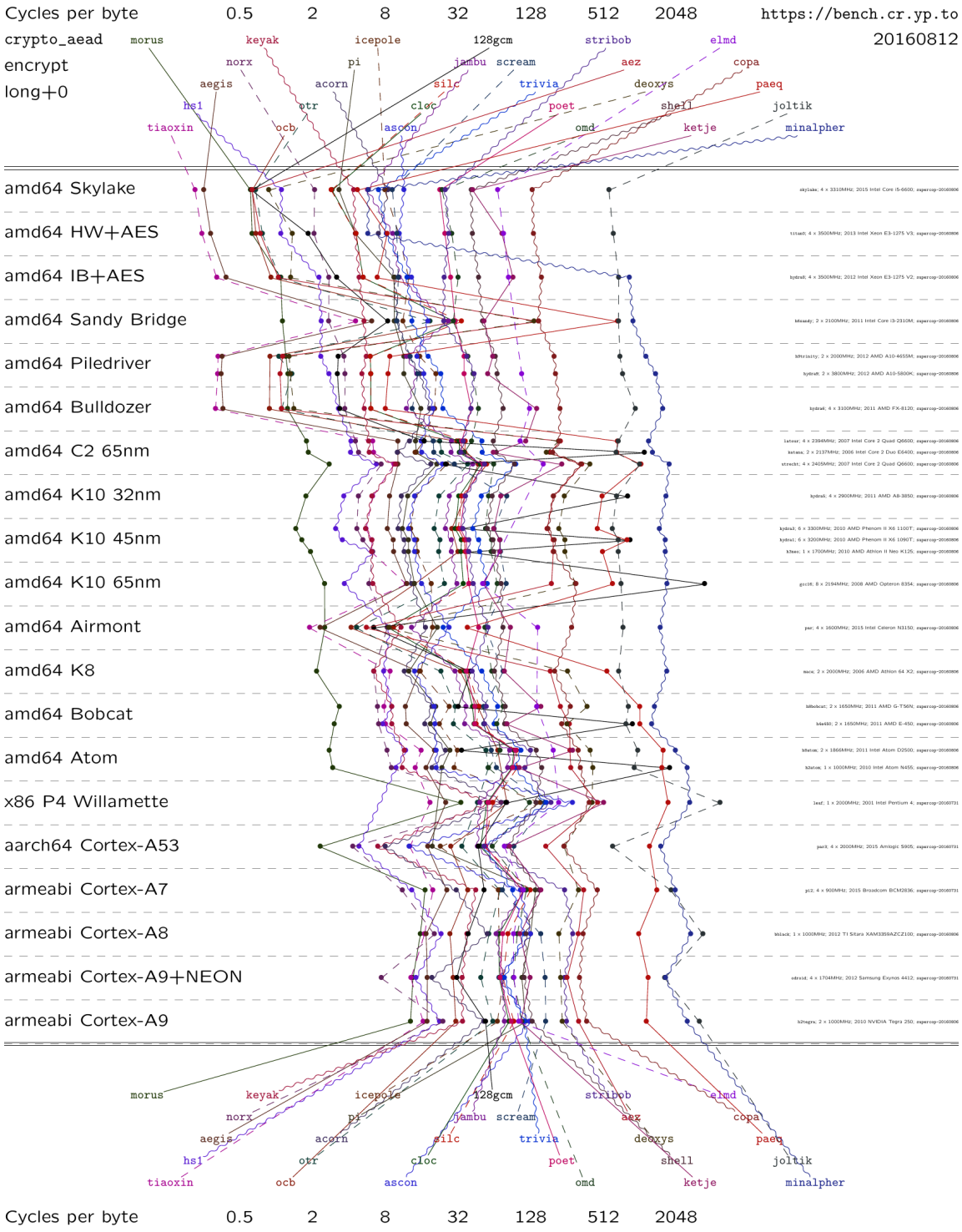
π-Cipher е дизајниран да може лесно да се имплементира на различни архитектури на процесори 16, 32 или 64 битни, во зависност од должината на зборовите која што се користи $\omega = 16, 32, 64$. Спецификацијата може директно да биде преведена во код и притоа нема потреба од дополнителни специфични техники, како што се lookup-табели или bitslicing (како во случајот на блок шифрувачи). Јадрото на шемата на π-Cipher се состои од повторување на употребата на π функцијата, што дополнително ја поедноставува имплементацијата, т.е доволна е една имплементација на π функцијата. Во нашиот случај π функцијата се користи и во процесот на енкрипција и во процесот на декрипција без употреба на нејзината инверзна форма.

Референтната имплементација на π-Cipher за сите негови варијанти може да се најде на официјалниот сајт на натпреварот CAESAR [13] и официјалната веб страна на π-Cipher [95]. Мерено според референтната имплементација, брзината

на π-Cipher не е задоволителна. Имено за многу мали пораки како на пример SSH пакети (46 бајти), брзината на енкрипција на пакет изнесува околу 132 cpb (Cycles per Byte) кај π32-Cipher. Додека пак за TLS пакети (1.5KB), брзината на енкрипција изнесува околу 20 cpb кај π64-Cipher.

Направена е и оптимизирана варијанта за 64-битната верзија на π-Cipher [82] која за енкрипција на големи пораки ($>$ 2KB) достигнува брзина до околу 4 cpb. Лесната имплементација на π16-Cipher за мали уреди е имплементирана на Atmel AVR 8-bit MCU и се карактеризира со големина на код од 2.26 KB и брзина од 741.7 cpb.

Преглед на тоа каде се наоѓа нашата шема во множеството од сите кандидати во втората рунда на натпреварот е дадена на Слика 7-1. На оваа слика може јасно да се види дека по брзина ние сме во групата на 10-те најбрзи кандидати. Меѓутоа треба да се напомене дека имплементацијата на π-Cipher на системот за тестирање Supercop, е секвенцијална.



Слика 7-1: Споредба на најдобрите имплементации на секој од втората рунда кандидати на рамката за тестирање Supercop. Поврзаните податоци се 0 бајти, а пораката е 2048 бајти. π-Cipher е означен со pi

7.2.5 Хардвер

Со оглед на фактот што π-Cipher е нов дизјан и не користи во својата структура некои познати крипто примитиви, хардверската архитектура што е изведена е комплетно нова. Најголем проблем во дизјанот претставува функцијата π заради секвенцијалноста во својата структура, со што се намалува брзината на процесирање, а се зголемува зафатнината.

π-Cipher нуди 64-битна и 16-битна варијанта. Првата е наменета за пресметување со високи перформанси, додека пак последната за лесни решенија. Направен е 16-битен FPGA процесор за π16-Cipher кој всушност е првата лесна имплементација на овој шифрувач [26], [69]. За потребите на натпреварот дизајнирано е специјално API од ATHENa тимот на Универзитетот George Mason во САД [60], според кое би се дале најреални резултати при тестирањето на хардверските перформанси на шифрувачите. Според тоа API изработена е имплементација за π-Cipher од страна на еден од членовите на тимот и истата е достапна на нивниот официјален сајт [44]. На Слика 7-2 даден е пресек од резултатите добиени со тестирање на кандидатите од втората рунда над специфичната хардверска платформа.

7.3 Рецензија за π-Cipher од анонимен рецензент на натпреварот CAESAR

Во продолжение детално е дадена рецензијата на еден од членовите на стручната комисија на натпреварот CAESAR. Иако во правилата на натпреварот стоеше дека комисијата нема да ги коментира алгоритмите, по барање на анонимниот рецензент да ни се достави на увид неговата рецензија, од страна на главниот организатор на овој натпревар Dan Bernstein по пат на електронска пошта го добивме мислењето на рецензентот.

Result ID	Algorithm (Double Click)	Key Size [bits]	Impl Approach	Hardware API	Arch Type	Primary Opt Target	(Enc/Auth TP)/Slice [(Mbits/s)/Slice]	Auth-Only TP [Mbits/s]	Enc/Auth TP [Mbits/s]	Dec/Auth TP [Mbits/s]	Impl Freq [MHz]	CLB Slices	BRAMs	DSPs	Primary Designer Affiliation	Primary Designer Name(s)	Group	Megafunctions or Primitives
491	MORUS	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	66.911	93,140	93,140	93,140	363.8	1,392	0	0	CCRG, NTU Singapore	Tao Huang	CAESAR Round 2	No
639	ICEPOLE	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	37.311	57,981	57,981	57,981	396.4	1,554	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
454	ACORN	128	RTL	CAESAR Hardware API v1	32-bit	Area	36.469	11,269	11,269	11,269	352.2	309	0	0	CCRG, NTU Singapore	Tao Huang	CAESAR Round 2	No
455	AEGIS	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	32.297	78,287	78,287	78,287	305.8	2,424	0	0	CCRG, NTU Singapore	Tao Huang	CAESAR Round 2	No
486	Ketje	96	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	24.838	8,718	8,718	8,718	272.4	351	0	0	Ketje-Keyak Team	Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer	CAESAR Round 2	No
516	Trivia-ck	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	21.713	14,852	14,852	14,852	232.1	684	0	0	CERG, GMU USA	Michael X. Lyons	CAESAR Round 2	No
494	NORX	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	15.059	15,134	15,134	15,134	157.7	1,005	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
465	ASCON	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	12.319	5,371	5,371	5,371	377.6	436	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
515	STRIBOB	192	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	5.673	6,881	6,881	6,881	349.4	1,213	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
507	PRIMATEs-GIBBON	80	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	4.697	1,987	1,987	1,987	347.7	423	0	0	CERG, GMU USA	Ahmed Ferozpuri	CAESAR Round 2	No
488	Keyak	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	4.099	8,203	8,203	8,203	180.9	2,001	0	0	Ketje-Keyak Team	Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer	CAESAR Round 2	No
519	AES-GCM	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	4.043	3,837	3,837	3,837	329.7	949	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	Standards	No
612	ELmD	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	3.082	4,025	4,025	4,025	314.5	1,306	0	0	Lab Hubert Curien, St. Etienne, France	Cuauhtemoc Mancillas Lopez	CAESAR Round 2	No
490	Minalpher	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	3.051	2,447	2,447	2,447	372.7	802	0	0	Mitsubishi Electric Corporation	Takeshi Sugawara	CAESAR Round 2	No
674	Deoxys	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.981	2,844	2,844	2,844	333.3	954	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
498	OCB	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.936	3,608	3,608	3,608	338.3	1,229	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
509	PRIMATEs-HANUMAN	80	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.897	1,072	1,072	1,072	348.6	370	0	0	CERG, GMU USA	Ahmed Ferozpuri	CAESAR Round 2	No
457	AES-OTR	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.647	3,187	3,187	3,187	298.8	1,204	0	0	NEC, Japan	Kazuhiko Minematsu	CAESAR Round 2	No
520	CLOC	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.466	2,831	2,831	2,831	243.3	1,148	0	0	CERG, GMU USA	Farnoud Farahmand	CAESAR Round 2	No
501	PAEQ	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.342	3,235	4,960	4,960	283.1	2,118	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol, Ahmed Ferozpuri	CAESAR Round 2	No
502	Pi-Cipher	128	RTL	CAESAR Hardware API v1	Folded /4v	Throughput/Area	2.086	2,094	2,094	2,094	204.5	1,004	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
458	AEZ	384	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	2.047	8,421	3,368	3,368	329.0	1,645	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
513	SILC	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	1.883	3,565	1,860	1,860	334.2	988	0	0	CLOC-SILC Team	Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, Eita Kobayashi	CAESAR Round 2	No
510	SCREAM	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	1.580	1,100	1,100	1,100	180.4	696	0	0	CERG, GMU USA	William Diehl	CAESAR Round 2	No
675	Joltik	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	1.533	759	759	759	391.2	495	0	0	CERG, GMU USA	Ekawat "Ice" Homsirikamol	CAESAR Round 2	No
505	POET	128	RTL	CAESAR Hardware API v1	Basic Iterative	Throughput/Area	1.458	3,154	3,154	3,154	246.4	2,163	0	0	CERG, GMU USA	William Diehl	CAESAR Round 2	No

Слика 7-2: Позиција на π-Cipher според тестирањата на GMU API хардверските имплементации

π-Cipher е автентикациско енкрипциска примитива базирана на сунѓер конструкција. Постои еден сунѓер (Common Internal State) со двојно повеќе битови од блокот што се процесира. Поврзаните податоци и оригиналниот текст, имаат влијание над CIS во текот на процесирањето. Можеби најинтересната карактеристика на оваа конструкција е дека блоковите на AD и пораката може да се процесираат со произволен паралелизам, иако процесирањето на AD мора да заврши пред да почне пораката да се процесира. Не постои зголемување на пораката со исклучок на додавањето на финалниот таг.

Коментари за документацијата:

Со малата промена во верзија 2 се промени правилото за пополнување на поврзаните податоци и пораката, оставајќи ја спецификацијата за a и m во секцијата 1.1 малку двосмислена. Тоа би требало да разјасни дека бројот на податочни блокови може да биде за еден поголем отколку големината на

поврзаните податоци. (Ова е веќе јасно во описот на малата промена и подоцна во текстот, па не е голем проблем).

Процесирањето на SMN е избегнато ако $|SMN| = 0$. Доволно фер. Но треба да појаснете дека T'' е едноставно T' ако процесирањето е избегнато, во спротивно пресметувањето на финалниот таг на Слика 1.6 нема дефиниран таг. Алгоритмите 1 и 2 не одразуваат дека е возможно да се избегне ова процесирање. Буквално читање на алгоритмите би повлекло дека постои дополнителен повик на π ако нема SMN.

Функционалност:

*Функциите за половина рунда E_1 и E_2, секоја започнува со примена на операцијата * со константна вредност како еден од операндите. Сега, μ и ν оперирајќи над константите ќе произведат константи исто така. Па според тоа оптимизацијата би била едноставно да се додаде константата на J_1 и J_N соодветно пред да се примени σ. Додека влезните константи се избрани да бидат балансирани, тоа речиси е случајот кога овие подфункции ќе се применат над константите, тие нема повеќе да се балансирани, па ако оптимизирате, може подобро е само да ги додадете овие оригинални константи директно.*

Објавени напади:

Публикуван е напад на откривање на клучот за 2.5 рунди на π-Cipher. Комплетниот шифрувач е специфициран со 3 рунди во својата пермутација π. Секоја половина рунда ги проследува податоците на CIS во една насока напред, а потоа наназад итн. Според тоа, со ограничување на π функцијата на тој начин што ќе се исфрли последната половина рунда наназад, дозволува стил на напад што е погоди и одгатни. Лично, јас бев воодушевен од аналогијата на 5 рунди наспроти 6 рунди кај Luby-Rackoff конструкција. Овој напад не ми личи дека може да се изведе за цел шифрувач или да води кон таа насока. Меѓутоа трудот покажува доста убедливо дека предложената варијанта со 2 рунди од π-Cipher не треба да се користи.

Перформанси:

Карактеристиките за перформансите на π-Cipher не ме воодушевија како добри; со оглед на едноставноста и структурата јас очекував нешто подобро.

По размислување, Сватив дека тоа е ценаша што треба да се плаши при паралелно процесирање. За секој податочен блок, CIS мора да биде независно процесирана, а после комбинирана со оригиналниот текст и повторно процесирана за производство на шифрираниот текст. Според тоа за секој податочен блок, постојат всушност 6 рунди на функцијаша π (каде што е завршена вистинскаша работа). Наспроти ова, Ascon, кој користи синџир конструкција, прави некое дополнително процесирање на почетокот и крајот, но само неколку рунди помеѓу блоковиште.

– – – – –

Опис на π-Cipher

- *без повторување на вредноста на нонсот*
- *постојат три можни големини на состојбаша: 256 бита, 512 бита и 1024 бита*
- *комплетно паралелен*
- *базиран на пермутацијаша π*
- *секој блок од пораката е процесиран со користење на две π пермутации. За секој блок од пораката, прваша пермутација π како влез ја користи тајнаша интерна состојба и 64 битен бројач. Дел од излезот од прваша пермутација може да се искористи за енкрипција. Пораката се XOR-ира со излезот од прваша пермутација, а потоа се користи како влез во втораша пермутација. Излезиште од вториште пермутации се собираат заедно за да го генерираат автентикацискиот таг (прикажано на Слика 1.4 и 1.5).*
- *π пермутација: оваа пермутација се состои од три рунди (4 рунди во прваша верзија). Секоја рунда (прикажано на Слика 1.8) се состои од 8 последователни * операции. Секоја * операција се состои од ARX операции (прикажано на Слика 1.7).*

Сигурност

- *Постои труд "Key Recovery Attack against 2.5-round π-Cipher" http://eprint.iacr.org/2016/502.pdf. Тој разбива 2.5 рунди од π-Cipher.*

- *Дизајнерите оспориле на официјалната листа со адреси дека само 2 рунди може да се разбијат.*
  http : //pi − cipher.org/upload/Response − Pi − Cipher − Attack.pdf.
  Во секој случај тоа е како да се бројат рундите.

Перформанси

- На процесорот Intel Skylake CPU брзината на π-Cipher (pi64cipher256, најголемата варијанта) е 3.3 cpb.
- Хардверските перформанси на π-Cipher беа дадени од страна на авторите на: *http : //pi − cipher.org/upload/measurement_table.pdf*. Изгледа дека овие имплементации се многу неефикасни. Дури и најмалата верзија (256-bit состојба) побарува повеќе од 1000 слајсови и протокот е исто така мал. Овие податоци не се корисни при евалуацијата.

Коментари

π-Cipher се состои од мала и голема варијанта, па така големата варијанта може да се смета во категоријата за високи перформанси, додека пак малата варијанта во категоријата за лесни шифрувачи. π пермутацијата е доста комплицирана и не може да биде ефикасна. Паралелното пресметување кај π-Cipher овозможува многу блокови да се пресметуваат паралелно, како и да се употреби SSE или AVX.

Глава 8

Заклучок

8.1 Преглед

Разгледувањата и резултатите изнесени во оваа докторска дисертација се однесуваат на дизајнот и анализата на една нова шема за автентикациска енкрипција π-Cipher која што е дизајнирана за широка употреба. Потврда за успешниот дизајн на оваа примитива е влезот во втората фаза на натпреварот за автентикациска енкрипција CAESAR, заедно со 20 други кандидати од целиот свет, како и релативно позитивната анализа добиена од CAESAR натпреварот. За разлика од поголемиот дел од нив, π-Cipher се карактеризира со оригиналност и иновативност. Наспроти користење на веќе постоечките примитиви во симетричната криптографија и нивно комбинирање во модови на операции, π-Cipher е оригинален и по тоа што користи нов ARX дизајн на основната крипто примитива и нова техника на обработка на тагот без учество на дополнителни скапи операции од типот на множење во поле. За прв пат на овој натпревар беше даден предлог за користење на нонсот како пар од јавен и таен број. Притоа оние дизајни кои не сакаат да го користат тајниот дел од нонсот со негово поставување на 0, ја добиваат старата и веќе позната конструкција на нонсот (да се состои од еден број кој што е јавен). Идејата на тајниот дел од нонсот, SMN не е, тој да игра улога на таен клуч, туку истиот да се енкриптира и автентицира и соодветно да се декриптира и верифицира. Со користењето на SMN, се дава можност за

рано откривање на фалсификат со користење на некои дополнителни техники за проверка на истото. Од сите кандидати кои влегоа во втората рунда на натпреварот, само π-Cipher и Icepole го користат SMN, како дел од нонсот во дизајнот на нивната шема.

Сепак, како што споменавме, дизјанот на π-Cipher се одликува со ARX дизајн на основната крипто примитива кој што е постигнат со употреба на квазигрупи и квазигрупни трансформации. Начинот на конструкција на една таква примитива (π функција) дава само доказ за употребливоста на овие алгебарски структури во крипто примитивите. Исто така како што беше опишано и во Глава 3 теоријата на квазигрупи отвара нови истражувачки проблеми во докажливоста на ARX дизајните.

Во Глава 4 е даден комплетен теориски доказ за сигурноста на модот на работа на π-Cipher со користење на SMN. Притоа јасно е назначено дека π-Cipher како и другите шеми кои користат сунѓер конструкција достигнува сигурност од $min\{2^{b/2}, 2^c, 2^k\}$ што е значително подобрување наспроти традиционалната граница дефинирана од сунѓер конструкциите $min\{2^{c/2}, 2^k\}$.

Покрај дизајнот на AEAD шемата, во оваа докторска дисертација предложени се и различни модови на работа на π-Cipher во зависност од намената. Еден таков мод е со користење на меѓу тагови. Главната идеја позади овој мод е да се постигне онлајн карактеристика на шифрувачот не само во фазата на енкрипција/автентикација, туку и во фазата на декрипција и верификација. Имено онлајн шифрувачите наоѓаат голема примена во лесната криптографија каде уредите се со ограничени ресурси. Во услови на ограничена меморија, мора да се внимава колкава е пораката што се обработува и дали верификацијата ќе заврши пред да се ослободи дешифрираниот текст. Токму шифрувачите со меѓу тагови ги решаваат овие проблеми. Друго својство со кое се карактеризира нашиот шифрувач што го одделува од другите кандидати на натпреварот CAESAR е можноста за работа со блокови со произволна должина. Благодарение на конструкцијата на π-функцијата која се базира на квазигрупи и квазигрупни трансформации, овозможено е π-Cipher да може да обработува блокови од 128 бита и до неколку мега бајти со што уште еднаш се потврдува неговата широка употреба. Кога

станува збор за големи податоци, тука, неминовен е ѝ фактот за инкрементално својство. Паралелниот дизајн и операцијата - собирање по компоненти која се користи за производство на тагот му овозможуваат на π-Cipher да се закити и со инкременталното својство.

За крај, во Глава 7 ги дискутираме предностите и недостатоците на нашата шема со веќе постоечкиот и референтен дизајн AES-GCM. Исто така дадена е комплетна споредбена анализа на сите кандидати кои го продолжија своето натпреварување во третата-финална рунда и нашиот дизајн. Во однос на оваа анализа може да се забележи дека π-Cipher во ниту една карактеристика не заостанува зад другите кандидати кои го продолжија своето натпреварување. Како потврда за ова е и добиената потврда од анонимен рецензент, кој во ниту еден момент рецензентот не смета дека дизајнот на π-Cipher е разбиен како причина з анегово не продолжување во следната фаза. Околу софтверските и хардверските перформанси, π-Cipher можеби не е најбрзиот или најлесниот дизајн, но секогаш се наоѓа во првите 10 нај кандидати.

Како заклучок може да се каже дека дизајнот на една ваква поразлична крипто примитива со сите свои предности ќе ги отвори вратите на новите дизајни во криптографијата. Направените истражувања дадоа навистина многу нови и оригинални резултати и доведоа до многу отворени прашања, чие што решавање би имало голема практична вредност и би допринело во решавање на разни видови проблеми првенствено поврзани со сигурноста и безбедноста на податоците во новиот тренд *Интернет на нештата*. Исто така, би напоменале дека резултатите добиени од овие истражувања се компатибилни со најсовремените истражувања во оваа област за што доказ е влезот во втората рунда на натпреварот за автентикациска енкрипција CAESAR и опстојувањето на овој шифрувач без некои посебни и посериозни криптоанализи кои би ја нарушиле неговата сигурност.

8.2 Отворени истражувачки проблеми

Интернет на нештата. Предложениот развој на Интернетот каде што секојдневните објекти прибираат или разменуваат податоци преку мрежна поврзаност се нарекува Интернет на нештата (анг. Internet of things). Во ваквата голема мрежа сè позначајно место заземаат паметните домови, интелигентниот транспорт, паметните градови итн. Секој објект е на некој начин идентификуван преку неговиот вграден мини компјутерски систем, а има можност да комуницира преку постоечката Интернет мрежа. Се разбира дека неминовно во овој голем свет на објекти е и безбедноста на податоците кои се разменуваат. Од тие причини, симетричната криптографија зазема значајно место во новиот тренд на Интернет на нештата, со посебен осврт на автентикациската енкрипција. Во оваа смисла предложената лесна шема на π-Cipher која користи 16-битни зборови, заедно со предностите на тајниот дел од нонсот може да се прилагоди за потребите на овој Интернет предизвик.

Големи податоци. Со оглед на самиот факт дека π-Cipher е паралелизабилен, адаптибилен кога станува збор за големината на блоковите кои треба да се процесираат и инкрементален, голема практична примена би можел да најде за енкриптирање на големи податоци кои се во мирување. За да може практично да се потврди тоа, во следниот период планираме да се направи комплетно паралелна имплементација на алгоритмот со користење на современите графички процесорски единици GPU, кои имаат големо ниво на паралелизам и пресметковна моќ. На тој начин ќе може и реално да се измерат перформансите на нашиот алгоритам, кои би достигнале и до 10 пати поголемо софтверско забрзување од досега понуденото.

Библиографија

[1] Farzaneh Abed, Christian Forler, and Stefan Lucks. General Overview of the Authenticated Schemes for the First Round of the CAESAR Competition. IACR ePrint 2014/792, 2014. `https://eprint.iacr.org/2014/792`.

[2] AES. *Advanced Encryption Standard*. FIPS PUB 197, Federal Information Processing Standards Publication, 2001.

[3] Joseph Alley and Josef Pieprzyk. State Recovery Attacks Against pi-Cipher. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '16, pages 43:1–43:6. ACM, 2016.

[4] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In *Advances in Cryptology – ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 105–125. Springer Berlin Heidelberg, 2014.

[5] Ralph Ankele and Robin Ankele. Software Benchmarking of the 2nd round CAESAR Candidates. IACR ePrint 2016/740, 2016. `https://eprint.iacr.org/2016/740`.

[6] Randall Atkinson and Dr. Stephen T. Kent. IP Encapsulating Security Payload (ESP). RFC 1827, RFC Editor, August 1995. `http://www.rfc-editor.org/rfc/rfc1827.txt`.

[7] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A Lightweight Hash. *Journal of Cryptology*, 26(2):313–339, 2013.

[8] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. `http://eprint.iacr.org/2013/404`.

[9] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. In *Advances in Cryptology* -

CRYPTO '94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings, pages 216–233, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[10] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In *Advances in Cryptology - CRYPT0' 95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings*, pages 15–28, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[11] Mihir Bellare and Daniele Micciancio. A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. In *EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 163–192. Springer, 1997.

[12] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology — ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings*, pages 531–545, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[13] Dan J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. CAESAR web page, 2013. `http://competitions.cr.yp.to/index.html`.

[14] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 320–337, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic Sponge Functions, 2011. `http://sponge.noekeon.org/CSF-0.1.pdf`.

[16] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. SPONGENT: A Lightweight Hash Function. In *Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*, pages 312–325, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[17] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Alex Poschmann, Matt J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware*

and Embedded Systems - CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings, pages 450–466. Springer Berlin Heidelberg, 2007.

[18] Christina Boura, Avik Chakraborti, Gaëtan Leurent, Goutam Paul, Dhiman Saha, Hadi Soleimany, and Valentin Suder. Key Recovery Attack Against 2.5-Round pi-Cipher. In *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 535–553, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel and Martin Schläffer. Ascon v1.2. CAESAR web page, 2016. `https://competitions.cr.yp.to/round3/asconv12.pdf`.

[20] CISCO. The Zettabyte Era—Trends and Analysis. White Paper, June, 2016. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`.

[21] Historical cost of computer memory and storage. hblok.net Freedom, Electronics and Tech, December, 2015. `https://hblok.net/blog/storage/`.

[22] Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, Rune Erlend Jensen, and Daniel Otte. π-Cipher v2.0. CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/picipherv20.pdf`.

[23] Christophe De Cannière and Bart Preneel. Trivium. In *New Stream Cipher Designs: The eSTREAM Finalists*, pages 244–266, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[24] József Dénes and Donald Keedwell. *Latin Squares and Their Applications*. Academic Press, 1974.

[25] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol. Technical report, IETF RFC 5246, 2008. `https://www.ietf.org/rfc/rfc5246.txt`.

[26] Mohamed El-Hadedy, Hristina Mihajloska, Danilo Gligoroski, Amit Kulkarni, Dirk Stroobandt, and Kevin Skadron. A 16-Bit Reconfigurable Encryption Processor for pi-Cipher. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 162–171, 2016.

[27] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. AES-COPA v2.0. CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/aescopav2.pdf`.

[28] Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser and Kan Yasuda. COLM v1. CAESAR web page, 2016. `https://competitions.cr.yp.to/round3/colmv1.pdf`.

[29] Elena Andreeva, Begul Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1.02. CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/primatesv102.pdf`.

[30] Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. The POET Family of On-Line Authenticated Encryption Schemes. CAESAR web page, 2015. `http://competitions.cr.yp.to/round2/poetv20.pdf`.

[31] Pierre-Alain Fouque, Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Authenticated On-Line Encryption. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003. Revised Papers*, pages 145–159, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[32] Kane Fulton. The 10 best graphics cards in the world. Open report in techradar.com, 2016. `http://www.techradar.com/news/computing-components/graphics-cards/best-graphics-cards-1291458`.

[33] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. The Stream Cipher Edon80. In *New Stream Cipher Designs: The eSTREAM Finalists*, pages 152–169, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[34] Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, and Rune Erlend Jensen. π-Cipher v2. Cryptographic competitions: CAESAR, 2014. `http://competitions.cr.yp.to/caesar-submissions.htmls`.

[35] Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Håkon Jacobsen, Rune Erlend Jensen, and Mohamed El-Hadedy. π-Cipher: Authenticated Encryption for Big Data. In *Secure IT Systems: 19th Nordic Conference, NordSec 2014, Tromsø, Norway, October 15-17, 2014, Proceedings*, pages 110–128, Cham, 2014. Springer International Publishing.

[36] Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, and Vlastimil Klima. Cryptographic Hash Function EDON-$\mathcal{R}'$. In *1st International Workshop on Security and Communication Networks*, pages 85–95, Trondheim, Norway, May 2009. IEEE.

[37] Emmanuel Goossaert. Coding for SSDs – Part 3: Pages, Blocks, and the Flash Translation Layer, February, 2014. http://codecapsule.com/2014/02/12/coding-for-ssds-part-3-pages-blocks-and-the-flash-translation-layer/.

[38] Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche, and Ronny Van Keer. Keyak v2. CAESAR web page, 2015. https://competitions.cr.yp.to/round2/keyakv2.pdf.

[39] Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche, and Ronny Van Keer. Ketje v2. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/ketjev2.pdf.

[40] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON Family of Lightweight Hash Functions. In *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 222–239, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[41] Shai Halevi and Phillip Rogaway. A Parallelizable Enciphering Mode. In *Topics in Cryptology – CT-RSA 2004: The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, pages 292–304, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[42] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a Stream Cipher for Constrained Environments. *Int. J. Wire. Mob. Comput.*, 2(1):86–93, May 2007.

[43] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 493–517. Springer Berlin Heidelberg, 2015.

[44] Ekawat Homsirikamol, William Diehl, Ahmed Ferozpuri, Farnoud Farahmand, Malik Umar Sharif, and Kris Gaj. A Universal Hardware API for Authenticated Ciphers. In *Proc. 2015 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2015*. IEEE, Dec 2015.

[45] Hongjun Wu. ACORN: A Lightweight Authenticated Cipher v3. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/acornv3.pdf.

[46] Hongjun Wu and Bart Preneel. AEGIS: A Fast Authenticated Encryption Algorithm. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/aegisv11.pdf.

[47] Hongjun Wu and Tao Huang. The Authenticated Cipher MORUS. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/ketjev2.pdf.

[48] Hongjun Wu and Tao Huang. The JAMBU Lightweight Authentication Encryption Mode. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/jambuv21.pdf.

[49] Hristina Mihajloska and Danilo Gligoroski. Construction of Optimal 4-bit S-boxes by Quasigroups of Order 4. SECURWARE 2012 : The Sixth International Conference on Emerging Security Information, Systems and Technologies, 2012. https://www.thinkmind.org/download.php?articleid=securware_2012_6_30_30103.

[50] IDEMA. The Advent of Advanced Format. *idema.org*, 2013. http://www.idema.org/?page_id=2369.

[51] Ivca Nikolić. Tiaoxin - 346. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/tiaoxinv21.pdf.

[52] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. In *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 31–49, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[53] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX v2.0. CAESAR web page, 2015. https://competitions.cr.yp.to/round2/norxv20.pdf.

[54] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.4. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/deoxysv14.pdf.

[55] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In *Advances in Cryptology – ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 85–104, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[56] Charanjit S. Jutla. Encryption Modes with Almost Free Message Integrity. *Journal of Cryptology*, 21(4):547–578, 2008.

[57] Kazuhiko Minematsu. AES OTR v3. CAESAR web page, 2016. https://competitions.cr.yp.to/round3/aesotrv31.pdf.

[58] Lars R. Knudsen. Contemporary block Ciphers. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 105–126. Springer-Verlag, 1999.

[59] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 310–331, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[60] CERG: Cryptographic Engineering Research Group Kris Gaj. ATHENa: Automated Tools for Hardware EvaluatioN, 2016. `https://cryptography.gmu.edu/athena/index.php?id=CAESAR_source_codes`.

[61] Charles F. Laywine and Gary L. Mullen. *Discrete Mathematics Using Latin Squares*. 1484 Series. Wiley, 1998.

[62] Pierre L'Ecuyer and Richard Simard. TestU01: A C Library for Empirical Testing of Random Number Generators. *ACM Trans. Math. Softw.*, 33(4):22:1–22:40, August 2007.

[63] Gaëtan Leurent. Analysis of Differential Attacks in ARX Constructions. In *Advances in Cryptology – ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 226–243, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[64] Markku-Juhani O. Saarinen and Billy B. Brumley. STRIBOBr2: "WHIRLBOB". CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/stribobr2.pdf`.

[65] Smile Markovski, Danilo Gligoroski, and Verica Bakeva. Quasigroup String Processing. In *Part 1, Contributions, Sec. Math. Tech. Sci., MANU, XX*, pages 1–2, 1999.

[66] Smile Markovski, Danilo Gligoroski, and Ljupco Kocarev. Unbiased Random Sequences from Quasigroup String Transformations. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, pages 163–180, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[67] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004. Proceedings*, pages 343–355, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[68] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[69] Hristina Mihajloska, Mohamed El-Hadedy, Danilo Gligoroski, and Kevin Skadron. Lightweight version of pi-cipher. NIST Lightweight Cryptography Workshop 2015, July 2015.

[70] Hristina Mihajloska, Danilo Gligoroski, and Simona Samardjiska. Reviving the Idea of Incremental Cryptography for the Zettabyte Era. Use Case: Incremental Hash Functions Based on SHA-3. In *Open Problems in Network Security: IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers*, pages 97–111. Springer International Publishing, 2016.

[71] Ran Canetti Mihir Bellare and Hugo Krawczyk. Message Authentication Using Hash Functions: The HMAC Construction. *RSA Laboratories' CryptoBytes Technical Newsletter*, 2(1), 1996.

[72] Kazuhiko Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In *Advances in Cryptology – EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 275–292, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[73] Nicky Mouha, Vesselin Velichkov, Christophe De Cannière, and Bart Preneel. The Differential Analysis of S-Functions. In *Selected Areas in Cryptography: 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, pages 36–56, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[74] Chanathip Namprempre, Phillip Rogaway, and Tom Shrimpton. A5 Security Notions: Definitions Implicit in the CAESAR Call. Cryptology ePrint Archive, Report 2013/242, 2013. `http://eprint.iacr.org/2013/242`.

[75] National Bureau of Standards (USA). *DES Modes of Operation*. FIPS PUB 81, Federal Information Processing Standards Publication, 1981.

[76] National Institute of Standards and Technology (NIST). *Recommendation for Block Cipher Modes of Operation*. NIST Special Publication SP 800-38A, 2001.

[77] National Institute of Standards and Technology (NIST). *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication SP 800-38B, 2005.

[78] National Institute of Standards and Technology (NIST). *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST Special Publication SP 800-38D, 2007.

[79] National Institute of Standards and Technology (NIST). *The Keyed-Hash Message Authentication Code (HMAC)*. FIPS PUB 198-1, Federal Information Processing Standards Publication, 2008.

[80] Samuel Neves. Cryptography in GPUs. Master's thesis, Faculty of Sciences and Technology of the University of Coimbra, Coimbra, Portugal, July 2009.

[81] Nilanjan Datta and Mridul Nandi. ELmD v2.0. CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/elmdv20.pdf`.

[82] Daniel Otte. Reference Implementation of Pi-Cipher. GitHub, 2016. `https://github.com/nerilex/pi-cipher-ref`.

[83] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wojcik. ICPOLE v2. CAESAR web page, 2015. `https://competitions.cr.yp.to/round2/icepolev2.pdf`.

[84] Petr Švenda. Basic comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS), 2004. `https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf`.

[85] Bart Preneel. CBC-MAC and Variants. In *Encyclopedia of Cryptography and Security*, pages 184–188, Boston, MA, 2011. Springer US.

[86] Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes. *J. Cryptol.*, 28(4):769–795, October 2015.

[87] Phillip Rogaway. Authenticated-encryption with Associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 98–107, New York, NY, USA, 2002. ACM.

[88] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *Advances in Cryptology - ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, pages 16–31, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[89] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings*, pages 373–390, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[90] Kenneth H. Rosen. *Handbook of Discrete and Combinatorial Mathematics, Second Edition*, chapter 12. Chapman & Hall/CRC, 2nd edition, 2010.

[91] Simona Samardjiska and Hristina Mihajloska. Towards Cryptanalysis of ARX based ciphers. The 13th Conference for Informatics and Information Technology (CIIT 2016), Bitola, Macedonia, 2016.

[92] Peter Schwabe. Graphics Processing Units. White paper, 2013. `https://cryptojedi.org/papers/gpus-20130310.pdf`.

[93] Jonathan D. H. Smith. *An Introduction to Quasigroups and Their Representations (Studies in Advanced Mathematics)*. Chapman and Hall/CRC, 2007.

[94] EMC Digital Universe Study. Discover the Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. Open report, 2014. `http://www.emc.com/leadership/digital-universe/index.htm?pid=home-dig-uni-090414`.

[95] Pi-Cipher Team. π-Cipher - Authenticated Encryption Cipher with Associated Data. Web site, 2016. `http://pi-cipher.org/`.

[96] Ted Krovetz and Phillip Rogaway. OCB v1.1. CAESAR web page, 2016. `https://competitions.cr.yp.to/round3/ocbv11.pdf`.

[97] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka and Eita Kobayashi. CLOC & SILC v3. CAESAR web page, 2016. `https://competitions.cr.yp.to/round3/clocsilcv3.pdf`.

[98] `https://aezoo.compute.dtu.dk/doku.php`.

[99] Viet Tung Hoang, Ted Krovetz and Phillip Rogaway. AEZ v4.2: Authenticated Encryption by Enciphering. CAESAR web page, 2016. `https://competitions.cr.yp.to/round3/aezv42.pdf`.

[100] David Wagner. A Generalized Birthday Problem. In *CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

[101] Doug Whiting, Russ Housley, and Niels Ferguson. IEEE 802.11-02/001r2: AES Encryption and Authentication Using CTR Mode and CBC-MAC. March 2002.