Chapter 2
Catalog and Illustrative Examples of Lightweight Cryptographic Primitives

Aleksandra Mileva, Vesna Dimitrova, Orhun Kara, and Miodrag J. Mihaljević

Abstract The main objective of this chapter is to offer to practitioners, researchers and all interested parties a brief categorized catalog of existing lightweight symmetric primitives with their main cryptographic features, ultimate hardware performance, and existing security analysis, so they can easily compare the ciphers or choose some of them according to their needs. Certain security evaluation issues have been addressed as well. In particular, the reason behind why modern lightweight block cipher designs have in the last decade overwhelmingly dominated stream cipher design is analyzed in terms of security against tradeoff attacks. It turns out that it is possible to design stream ciphers having much smaller internal states.

2.1 Introduction

Lightweight cryptography aims to deploy cryptographic algorithms in resource-constrained devices such as embedded systems, RFID devices and sensor networks. The cryptographic community has done a significant amount of work in this area, including design, implementation and cryptanalysis of new lightweight cryptographic algorithms, together with efficient implementation of conventional cryptography algorithms in constrained environments (see the Lightweight Cryp-

A. Mileva (✉)
Universitet "Goce Delcev", Štip, Republic of Macedonia
e-mail: aleksandra.mileva@ugd.edu.mk

V. Dimitrova
University "Ss Cyril and Methodius", Skopje, Republic of Macedonia

O. Kara
Department of Mathematics, IZTECH Izmir Institute of Technology, Izmir, Turkey
e-mail: orhunkara@iyte.edu.tr

M. J. Mihaljević
Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Serbia

© The Author(s) 2021

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_2

tography Lounge,[1] [89, 260, 391]). Most recent cryptographic competitions such as NIST's SHA-3 Cryptographic Hash Algorithm Competition[2] and eSTREAM project[3] (with the Profile 2) had requirements that support implementations for highly constrained devices. Additionally, NIST currently is working on a special call[4] to create a portfolio of lightweight algorithms through an open standardization process.

The lightweightness of a given cryptographic algorithm can be obtained in two ways, by optimized implementations with respect to different constraints or by dedicated designs which use smaller key sizes, smaller internal states, smaller building blocks, simpler rounds, simpler key schedules, etc. There are several relevant metrics for assessing lightweight algorithms, such as power and energy consumption, latency, throughput and resource requirements [404]. Power and energy consumption are important for devices that are battery-oriented or energy harvesting. Latency is the time taken to perform a given task, and is important for applications where fast response time is necessary (e.g., Advanced Driver Assistance Systems), while throughput can be defined as the rate at which the plaintext is processed per time unit, and is measured in Bps.

Resource requirements are expressed differently in hardware and software implementations. In the hardware case, they are described as gate area, expressed by logic blocks for FPGAs or by Gate Equivalents (GEs) for ASIC implementations. However, these measures highly depend on the particular technology, so it is not possible to do a fair and relevant comparison of the lightweight algorithm implementations exactly across different technologies. In the software case, resource requirements are described as number of registers, RAM and ROM consumption in bytes. ROM consumption corresponds in fact with the code size.

Hardware implementations are suitable for highly constrained devices. For example, on the low end, low-cost passive RFID tags may have a total of 1000–10,000 gates, with only 200–2000 budgeted for security purposes [309]. Software implementations are suitable for less constrained devices, and they are optimized for throughput and energy consumption.

Some design choices related to dedicated lightweight cryptographic algorithms have influences on the security margins. For example, smaller key sizes such as 80 bits or 96 bits are in conflict with the current NIST minimum key size requirement of 112 bits. Smaller block and output sizes in some algorithms may lead to plaintext recovery or codebook attacks. Simpler key schedules may enable different attacks using related keys, weak keys, etc. Smaller internal state (IS) and digest sizes in hash functions may lead to collision attacks. Simpler rounds sometimes means that more iterations are required to achieve security.

[1] https://cryptolux.org/index.php/Lightweight_Cryptography.

[2] Part 2.B.2, Federal Register Notice (2 November 2007).

[3] http://www.ecrypt.eu.org/stream/call/.

[4] https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/Draft-LWC-Submission-Requirements-April2018.pdf.

The main objective of this chapter is to offer to practitioners, researchers and all interested parties a short categorized catalog of existing symmetric lightweight primitives with their main features, some details about known software and hardware performance, and existing security analysis, to enable selection according to specific needs. These cryptographic primitives can be categorized into five areas: block and stream ciphers, hash functions, message authentication codes, and authenticated encryption schemes. As a consequence of the simplicity which provides lightweightness, the security evaluation of lightweight stream ciphers appears as an issue of top importance, and so a number of illustrative elements relevant for cryptanalysis of lightweight encryption techniques have been pointed out as well.

It can easily be observed that (see Sect. 2.2) almost all of the recently designed lightweight ciphers are block ciphers. The requirement for unnecessarily large internal states results in extra hardware area cost which definitely hinders designing ultralightweight stream ciphers. We analyze the arguments behind this criterion and propose to loosen it by justifying the security analysis in Sect. 2.3. We believe this adoption will promote the design and even the analysis of lightweight stream ciphers.

2.2 Catalog of Lightweight Cryptographic Primitives

The catalog of lightweight cryptographic primitives is divided in five categories: block and stream ciphers, hash functions, message authentication codes, and authenticated encryption schemes.

2.2.1 Block Ciphers

Block ciphers encrypt one block of plaintext bits at a time, to a block of ciphertext bits, through multiple rounds, and using a secret key. Each round is a sequence of several simple transformations, which provide confusion and diffusion [522]. In each round, a round key is used, which is derived from the secret key using a key schedule algorithm. According to the algorithm structure, block ciphers can be divided into several types:

- Substitution Permutation Network (SPN)—each round consists of substitution (S-) and permutation (P-) boxes. Usually, S-boxes are non-linear transformations and provide confusion, while P-boxes are linear and provide diffusion.
- Feistel Network (Feistel)—divides the input block into two halves, L_i and R_i, and in each round, the output block is $(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus F(R_i, K_{i+1}))$, where F is the round-function (introduced by H. Feistel [209]).

- Add-Rotate-XOR (ARX)—only three operations are used: modular addition, rotation and XOR.
- Generalized Feistel Network (GFN)—divides the input block into n parts, and each round consists of a round-function layer and a block-permutation layer, which usually is a cyclic shift. If the round-function is applied only to one part, we speak about Type-1, and if it is applied on the $n/2$ parts, we speak about Type-2 GFN. If there is an additional linear layer between the two layers, we speak about Extended GFN [78].
- LFSR-based—in the round function they use one or more Linear Feedback Shift Registers (LFSRs) in combination with non-linear functions.
- LS-design—each round combines linear diffusion L-boxes with non-linear bitslice S-boxes, and they are aimed at efficient masked implementations against side-channel analysis [247].
- XLS-design—a variation of the LS-design, that uses the additional ShiftColumns operation, and Super S-boxes [306].

There are also tweakable block ciphers, which in addition to the key and the message have a third input named tweak, and they must be secure even if the attacker is able to control the tweak input. Each tweakable block cipher can be seen as a family of permutations in which each (key, tweak) pair selects one permutation.

The standard block cipher approach can be made lightweight by using smaller key sizes (e.g., 80 or 96 bits), smaller block sizes (e.g., 64 bits), smaller or special building blocks (e.g., 4-bit S-boxes, no S-boxes at all, or recursive diffusion layers), simpler key schedules (e.g., selecting a key schedule where bits from the master key are selected as round keys), smaller hardware implementation, involutive encryption, etc. AES-128 belongs in this group also, because there are ASIC implementations of it with an area of just 2400 GE[426] on 0.18 μm technology, but it cannot be applied in every scenario. In Table 2.1, we give a summary of the known lightweight block ciphers, sorted in alphabetical order, with their type, key and block size in bits, number of rounds, used technology and number of GEs if known, and we give the best known attacks in Table 2.2. KASUMI used in UMTS, GSM, and GPRS mobile communications systems, 3-Way and MANTIS are considered insecure. Additionally, CLEFIA and PRESENT are part of the ISO-29192-2 standard, while HIGHT, MISTY1 and AES are part of the ISO/IEC 18033-3:2010 standard.

For fair and consistent evaluation and comparison of software implementations of lightweight block and stream ciphers, one can use a free and open-source benchmarking framework FELICS (Fair Evaluation of Lightweight Cryptographic Systems) [182]. Currently, the assessment can be done on three widely used microcontrollers: 8-bit AVR, 16-bit MSP and 32-bit ARM, and extracted metrics are the execution time, RAM consumption and binary code size, from which one single value “Figure Of Merit” (FOM) is calculated. Table 2.3 presents some details about software performance of some lightweight block ciphers with the current best FELICS results for encryption of 128 bytes of data in CBC mode (scenario 1 in [182]), sorted according to the FoM measure, where the lowest result is the best.

Table 2.1 Lightweight block ciphers (characteristics)

Name	Ref	Type	Key size (bits)	Block size (bits)	No. of rounds	Techno. (μm)	No. of GEs
3-Way	[164]	SPN	96	96	11	–	–
AES-128	[166]	SPN	128	128	10	0.18	2400
CLEFIA	[527]	Type-2 GFN	128/192/256	128	18/22/26	0.13	2604 [16] (CLEFIA-128)
DESL/DESLX	[361]	Feistel	56/184	64	16	0.18	1848/2168
Fantomas	[247]	SPN+LS-design	128	128	12	–	–
FLY	[317]	SPN	128	64	20	–	–
GOST revisited	[487]	Feistel	256	64	32	0.18	651
GRANULE	[54]	Feistel	80/128	64	32	0.18	1288/1577
HIGHT	[283]	ARX+Type-2 GFN	128	64	32	0.25	3048
ICEBERG	[541]	SPN	128	64	16	–	–
ITUbee	[315]	Feistel	80	80	20	–	–
KASUMI	[1]	Feistel	128	64	8	0.11	2990 [586]
KATANn / KTANTANn	[126]	LFSR-based	80	$n \in \{32, 48, 64\}$	254	0.13	1054 ($n = 64$)
						0.13	462 ($n = 32$)
KLEIN	[239]	AES-like SPN	64/80/96	64	12/16/20	–	–
LBlock	[583]	Feistel	80	64	32	0.18	1320
LEA	[282]	ARX+GFN	128/192/256	128	24/28/32	0.13	3826

(continued)

Table 2.1 (continued)

Name	Ref	Type	Key size (bits)	Block size (bits)	No. of rounds	Techno. (μm)	No. of GEs
LED	[252]	AES-like SPN	64/128	64	32/48	0.18	966/1265
Lilliput	[78]	Extended GFN	80	4	30	0.065	1581
MANTIS$_r^a$	[68]	SPN	128+64 tweakey	64	$r \in \{5, 7\}$	–	–
mCrypton	[372]	SPN	64/96/128	64	12	–	–
MIBS	[299]	Feistel	64/80	64	32	0.18	1396
Midori	[51]	AES-like SPN	128	64/128	16/20	0.09	2450/3661
MISTY1	[398]	Feistel	128	64	8	–	–
Mysterion	[306]	SPN+XLS-design	128/256	128/256	12/16	–	–
Noekeon	[165]	SPN	128	128	16	–	–
PICARO	[485]	Feistel	128	128	12	–	–
Piccolo	[526]	GFN	80/128	64	25/31	–	–
PRESENT	[101]	SPN	80/128	64	31	0.18	1075/1391
PRIDE	[17]	SPN	128	64	20	–	–
PRINCE	[105]	SPN	128	64	12	0.13	3491
PRINTcipher	[333]	SPN	80/160	48/96	48/96	0.18	402/726
PUFFIN2	[569]	SPN	80	64	34	0.18	1083
RC5-12	[502]	ARX+Feistel	128	64	12	–	–
RECTANGLE	[598]	SPN	80/128	64	25	0.13	1599.5/2063.5
RoadRunneR	[63]	Feistel	80/128	64	10/12	–	–
Robin	[247]	SPN+LS-design	128	128	16	–	–
SEA	[542]	Feistel	$n = m(6b)$	n	odd[b]	–	–

SKINNY[a]	[68]	SPN	(64, 128, 192)/(128, 256, 384) tweakey	64/128	(32, 36, 40)/(40, 48, 56)	0.18	(1223, 1696, 2183)/(2391, 3312, 4268)
Simeck	[588]	Feistel	64/96/128	32/48/64	32/36/44	0.13	549/778/1005
SIMON	[65]	Feistel	64/(72, 96)/(96, 128)/(96, 144)/(128, 192, 256)	32/48/64/96/128	32/36/(42, 44)/(52, 54)/(68, 69, 72)	0.13	1234 (SIMON 128/128)
SPARX	[181]	ARX+SPN	128/128/256	64/128/128	24/32/40	–	–
SPECK	[65]	ARX+Feistel	64/(72, 96)/(96, 128)/(96, 144)/(128, 192, 256)	32/48/64/96/128	22/(22, 23)/(26, 27)/(28, 29)/(32, 33, 34)	0.13	1280 (SPECK 128/128)
TWINE	[544]	Type-2 GFN	80/128	64	36	0.09	1799
QARMA[a]	[39]	SPN	128/256	64/128	16/24	–	–
XTEA	[436]	Feistel	128	64	64	0.13	3490
Zorro	[227]	AES-like SPN	128	128	24	–	–

[a]Indicate tweakable block ciphers
[b] $\frac{3n}{4} + 2(\frac{n}{2b} + \lfloor b/2 \rfloor)$

Table 2.2 Lightweight block ciphers (best known attacks)

Name	Ref	Best known attack: data complexity/memory/time complexity
3-Way	[164]	**Practical** related-key attack [320], 1 related key pair, 2^{22} CPs
AES-128	[166]	Biclique key-recovery attack [545]: 2^{56} / – /$2^{126.13}$
CLEFIA	[527]	Impossible differential attack [106]: $2^{114.58}$ / $2^{83.16}B$ /$2^{116.16}$
DESL/	[361]	Linear cryptanalysis on DES [311]: $2^{39} - 2^{41}$ DES evaluations
DESLX		Related-key attack on DESX[474]:$2^{3.5}$ KPs/– / 2^{56} DES evaluations
Fantomas	[247]	–
FLY	[317]	–
GOST revisited	[487]	Single-key KP differential attack [159]: 2^{64} / $2^{70}B$ /2^{179}
GRANULE	[54]	–
HIGHT	[283]	Biclique cryptanalysis [15]: 2^{8} / _ /$2^{126.07}$
ICEBERG	[541]	Differential cryptanalysis [543]: 2^{63} CPs /2^{96} enc. on 8 rounds
ITUbee	[315]	–
KASUMI	[1]	**Practical** related-key attack [192]: 4 related keys, 2^{26} / 2^{30} B / 2^{32}
KATAN*n*/	[126]	Meet-In-The-Middle attack on KTANTAN*n* [104]
KTANTAN*n*		(3, 2, 2) pairs/ –/($2^{75.17}$, $2^{75.044}$, $2^{75.584}$)
KLEIN	[239]	Truncated differential attack [497]: $2^{48.6}$ / 2^{32} /$2^{54.9}$ on KLEIN-64
LBlock	[583]	CP related-key impossible differential attack[584]: 2^{63} / – /$2^{75.42}$ on 24 rounds
LEA	[282]	–
LED	[252]	Random-difference distinguishers [443]: – / $2^{60}B$/$2^{60.3}$ on 40 rounds LED-128
Lilliput	[78]	Key-recovery attack with the division property [512]: 2^{63} / – /2^{77} on 17 rounds
MANTIS$_r$	[68]	**Practical** key-recovery attack [185]: 2^{28} / – /2^{38} enc. on MANTIS$_5$
mCrypton	[372]	Related-key impossible differential cryptanalysis [388]: ($2^{59.9}$, $2^{59.7}$) / ($2^{63.9}$, $2^{55.7}$)B /($2^{74.9}$, $2^{66.7}$) on 9 rounds
MIBS	[299]	Biclique cryptanalysis [519] (MIBS-80): 2^{52}/– /$2^{78.98}$
Midori	[51]	Key-recovery attack for the class of 2^{32} weak keys in Midori64 [250]: 2/– /2^{16}
MISTY1	[398]	Single-key integral attack [56]: 2^{64}/– /$2^{69.5}$
Mysterion	[306]	–

(continued)

Table 2.2 (continued)

Name	Ref	Best known attack: data complexity/memory/time complexity
Noekeon	[165]	Many related keys (weakness) [334]
PICARO	[485]	Related-key attack [129]: $2^{99}/2^{22}B$ /$2^{107.4}$
Piccolo	[526]	Biclique cryptanalysis [15]: 2^4/– /($2^{79.07}$, $2^{127.12}$)
PRESENT	[101]	Biclique cryptanalysis (PRESENT-80) [15]: 2^{22}/– /$2^{79.37}$
PRIDE	[17]	Multiple related-key differential attack [167]: $2^{41.6}$/– /$2^{42.7}$
PRINCE	[105]	Multiple differential attack [128]: $2^{57.94}$/$2^{61.52}$ /$2^{60.62}$ on 10 rounds
PRINTcipher	[333]	Invariance subspace attack [359] applicable to 2^{52}/ 2^{102} weak keys: 5 CPs/ –/ negligible
PUFFIN2	[569]	Differential attack [95]: $2^{52.3}$ CPs/– /$2^{74.78}$
RC5-12	[502]	Differential attack [88]: 2^{44} CPs
RECTANGLE	[598]	Related-key differential attack [521]: $2^{62}/2^{72}B/2^{67.42}$ on 19 rounds
RoadRunneR	[63]	–
Robin	[247]	Key-recovery attack for the weak key set of density 2^{-32} [360]: 1 CP/– /2^{64}
SEA	[542]	–
SKINNY	[68]	Related-tweakey impossible differential attacks [23]: $2^{71.4}/2^{64}$ /2^{79} up to 23 rounds
Simeck	[588]	Linear hull attack with dynamic key-guessing techniques [491]: ($2^{31.91}$, $2^{47.66}$, $2^{63.09}$)/ –/($2^{61.78}$, $2^{92.2}$, $2^{111.44}$) add. and ($2^{56.41}$, $2^{88.04}$, $2^{121.25}$) enc.
SIMON	[65]	Differential cryptanalysis on 12/16/19/28/37 reduced-round SIMON-32/48/64/96/128
SPARX	[181]	Truncated-differential attack [24]: 2^{32} /2^{61}/2^{93} on 16 rounds (SPARX-64/128)
SPECK	[65]	Differential cryptanalysis [537]: $2^{125.35}/2^{22}/2^{125.35}$ on 23 rounds of the SPECK-128/128
TWINE	[544]	Impossible differential and multidimensional zero correlation linear attack [373]: $2^{62.1}$ KPs/ $2^{60}B$ / 2^{73} (TWINE-80)
QARMA	[39]	–
XTEA	[436]	Related-key rectangle attack [380]: $2^{63.83}$ / – / $2^{104.33}$ on 36 rounds
Zorro	[227]	Differential attack [55]: $2^{41.5}$ / 2^{10} / 2^{45}

KP—Known Plaintext
CP—Chosen Plaintext

Table 2.3 The current best FELICS results for scenario 1: Encrypt 128 bytes of data in CBC mode

	AVR			MSP			ARM			
Cipher	Code (B)	RAM (B)	Time (Cyc.)	Code (B)	RAM (B)	Time (Cyc.)	Code (B)	RAM (B)	Time (Cyc.)	FoM
Speck	966	294	39,875	556	288	31,360	492	308	15,427	5.1
Speck	874	302	44,895	572	296	32,333	444	308	16,505	5.2
Simon	1084	363	63,649	738	360	47,767	600	376	23,056	7.0
Simon	1122	375	66,613	760	372	49,829	560	392	23,930	7.2
RECTANGLE	1152	352	66,722	812	398	44,551	664	426	35,286	8.0
RECTANGLE	1118	353	64,813	826	404	44,885	660	432	36,121	8.0
LEA	1684	631	61, 020	1154	630	46,374	524	664	17,417	8.3
SPARX	1198	392	65,539	966	392	36,766	1200	424	40,887	8.8
SPARX	1736	753	83,663	1118	760	53,936	1122	788	67,581	13.2
HIGHT	1414	333	94,557	1238	328	120,716	1444	380	90,385	14.8
AES	3010	408	58,246	2684	408	86,506	3050	452	73,868	15.8
Fantomas	3520	227	141,838	2918	222	85,911	2916	268	94,921	17.8
Robin	2474	229	184,622	3170	238	76,588	3668	304	91,909	18.7
Robin⋆	5076	271	157,205	3312	238	88,804	3860	304	103,973	20.7
RC5-20	3706	368	252,368	1240	378	386,026	624	376	36,473	20.8
PRIDE	1402	369	146,742	2566	212	242,784	2240	452	130,017	22.8
RoadRunneR	2504	330	144,071	3088	338	235,317	2788	418	119,537	23.3
RoadRunneR	2316	209	125,635	3218	218	222,032	2504	448	140,664	23.4
LBlock	2954	494	183,324	1632	324	263,778	2204	574	140,647	25.2
PRESENT	2160	448	245,232	1818	448	202,050	2116	470	274,463	32.8
PRINCE	2412	367	288,119	2028	236	386,781	1700	448	233,941	34.9
Piccolo	1992	314	407,269	1354	310	324,221	1596	406	294,478	38.4
TWINE	4236	646	297,265	3796	564	387,562	2456	474	255,450	40.0
LED	5156	574	2,221,555	7004	252	2,065,695	3696	654	594,453	138.6

2.2.2 Stream Ciphers

Stream ciphers encrypt small portions of data (one or several bits) at a time. By using a secret key, they generate a pseudorandom keystream, which is then combined with the plaintext bits to produce the ciphertext bits. Very often the combining function is bitwise XORing, and in that case we speak about binary additive stream ciphers. The basic security rule for stream ciphers is not to encrypt two different messages with the same pair of key/IV. So, stream ciphers usually have a large keystream period, and a different key and/or IV should be used after the period elapses. Each stream cipher usually has an initialization phase with some number of rounds (or clock-cycles), followed by an encryption phase. A fast initialization phase makes a given cipher suitable for encrypting many short messages, while when several large messages need to be encrypted, stream ciphers with a fast encryption phase are more appropriate.

The standard stream cipher approach can be made lightweight by using: smaller key sizes (e.g., 80 bits), smaller IV/nonce sizes (e.g., 64 bits), a smaller internal state

(e.g., 80 or 100 bits), simpler key schedules, a smaller hardware implementation, etc. Table 2.4 lists the known lightweight stream ciphers in alphabetical order, with their main parameters and details about hardware implementation, and Table 2.5 provides the best known attacks. One can notice that all eSTREAM Profile 2 candidates that were not selected as finalists are not in the table. Also, according to the hardware implementations, ZUC, ChaCha and Salsa20 cannot really be considered as lightweight. While Lizard uses 120 bit keys, its designers claim only 80-bit security against key-recovery attacks. A5/1 used in GSM protocol, E0 used in Bluetooth, A2U2, and Sprout are considered insecure.

Additionally, Enocoro and Trivium are part of the ISO/IEC 29192-3:2012 standard, and Rabbit is part of ISO/IEC 18033-4:2011. SNOW 3G was chosen for the 3GPP encryption algorithms UEA2 and UIA2, while ZUC was chosen for the 3GPP algorithms 128-EEA3 and 128-EIA3. The profile 2 eSTREAM portfolio includes Grain v1, MICKEY 2.0 and Trivium. There is an IETF implementation of the ChaCha20, published in RFC 7539, with 96-bit nonce and maximum message length up to $2^{32} - 1$B that can be safely encrypted with the same key/nonce, as a modification.

2.2.3 Hash Functions

A hash function is any function that maps a variable length input message into a fixed length output. The output is usually called a hashcode, message digest, hash value or hash result. Cryptographic hash functions must be preimage (one-way), second preimage and collision resistant.

Usually the message is first padded and then divided into blocks of fixed length. The most common method is to iterate over a so-called compression function, that takes two fixed size inputs, a message block and a chaining value, and produces the next chaining value. This is known as a Merkle-Damgård (MD) construction. The sponge construction is based on fixed-length unkeyed permutation (P-Sponge) or random function (T-Sponge), that operates on b bits, where $b = r + c$. b is called the width, r is called the rate (the size of the message block) and the value c the capacity. The capacity determines the security level of the given hash function. There is also a JH-like sponge in which the message block is injected twice.

The main problem of using conventional hash functions in constrained environments is their large internal state. SHA-3 uses a 1600 bit IS, and its most compact hardware implementation needs 5522 GE [471] on 0.13 µm technology. On the other hand, SHA-256 has a smaller IS (256 bit), but one of its smaller hardware implementations uses 10,868 GE [211] on 0.35 µm technology.

Lightweight hash functions can have smaller internal state and digest sizes (for applications where collision resistance is not required), better performance on short messages, small hardware implementations, etc. In some cases, for example tag-based applications, there is a need only for the one-way property. Also, most tag protocols require hashing of small messages, usually much less than 256 bits.

Table 2.4 Lightweight stream ciphers (characteristics)

Name	Ref	Key size (bits)	IV/nonce (bits)	IS (bits)	Output size (bits)	Max. keystream bits per (key, IV/nonce)	No. of init. rounds/cycles	Techno (μm).	No. of GEs
A2U2	[173]	61	64	95	1		var.	–	283 estimated
A5/1	[92]	64	22	64	1	228	86 + 100	–	–
BEAN	[350]	80	64	160	2		81	–	–
CAR30	[172]	128	120	256	128	$>2^{122}$	160	–	–
CAvium	[511]	80	80	288	1	–	144	–	–
ChaCha	[79]	256	64	512	512	2^{73}	8/12/20	0.18	9110 [270]
E0	[96]	8−128	26 + 48	132	1		240	–	–
Enocoro	[574, 575]	80/ 128(v2)	64	176/ 272	8	2^{35} 2^{67}	40/ 96	0.18/ 0.09	2700/ 4100
Fruit-80	[228]	80	70	80	1	2^{43}	160	0.18	960
Grain	[266, 267]	80(v1)/128	64/96	160/256	1	2^{43}	160	0.13	1294/1857 [240]
LILLE	[53]	80	80	80/100/120	40	$2^{32} \cdot 40$	720	0.09	911/991.6/1076.4
LIZARD	[253]	120	64	121	1	2^{18}	128+128	0.18	1161
MICKEY 2.0	[48]	80/ 128	80/ 128	200/ 320	1	2^{40}/ 2^{64}	260/ 416	0.13	3188/ 5039 [240]
Plantlet	[421]	80	90	110	1	2^{30}	320	0.18	928

Rabbit	[98]	128	64	513	128	2^{71}	4+4	0.18	3800
RAKAPOSHI	[148]	128	192	320	1	2^{64}	448	–	–
Salsa20	[80]	256	64	512	512	2^{73}	20	0.18	9970 [270]
SNOW 3G	[204]	128	128	576	32		32	–	–
Sprout	[27]	80	70	89	1	2^{40}	320	0.18	813
Trivium	[127]	80	80	288	1	2^{64}	1152	0.35	749 [409]
Quavium	[555]	80	80	288	1	2^{64}	1152	–	3496 estimated
WG-8	[207]	80	80	160	1	2^{160}	40	0.065	1786 [587]
ZUC (v 1.6)	[205]	128	128	560	32		32	0.065	12,500 [378]

Table 2.5 Lightweight stream ciphers (best known attacks)

Name	Ref	Best known attack: data complexity/memory/time complexity
A2U2	[173]	**Practical** key-recovery attack [524] under the KP attack model 210/−/$2^{24.7}$
A5/1	[92]	**Practical** Time-Memory tradeoff attack [92] 2sec KPs/ 2^{48} preprocessing steps to compute $300GB$/ 2^{24}
BEAN	[350]	Distinguishing attack [13] with 2^{17} keystream bits
CAR30	[172]	–
CAvium	[511]	–
ChaCha	[79]	Multi-bit differential attack [143]: 2^{28} / −/ 2^{233} on 7 rounds
E0	[96]	**Practical** key-recovery attack [381] using the first 24 bits of $2^{23.8}$ frames and 2^{38} computations
Enocoro	[574, 575]	–
Fruit-80	[228]	–
Grain	[266, 267]	Fast near collision attack [595]: 2^{19} / 2^{28}/ $2^{75.7}$ on Grainv1
LILLE	[53]	–
LIZARD	[253]	Distinguishing attack [52]: −/$2^{76.6}$/$2^{51.5}$ random IV enc
MICKEY 2.0	[48]	Practical related key attack [179] with 65/113 related (K,?IV) pairs and 0.9835/0.9714 success rate
Plantlet	[421]	Distinguishing attack [422]
Rabbit	[98]	Differential fault analysis [330] with 128 − 256 faults: −/$2^{41.6}$ B/2^{38}
RAKAPOSHI	[148]	Related key attack [297]: 2^{38} chosen IVs/−/ 2^{41}
Salsa20	[80]	Multi-bit differential attack [143]: 2^{96} / −/ $2^{244.9}$ on 8 rounds
SNOW 3G	[204]	Multiset distinguisher [90]: 2^{8} on 13 rounds
Sprout	[27]	Many, e.g., key recovery attack [50]: −/−/$2^{66.7}$ enc.
Trivium	[127]	Key-recovery attack [224]: 2^{77} on 855 rounds
Quavium	[555]	–
WG-8	[207]	Related key attacks [180] with one related key 2^{52} chosen IVs/−/ $2^{53.32}$
ZUC (v 1.6)	–	

KP—Known Plaintext

Tables 2.6 and 2.7 list the cryptographic and implementation properties of the known lightweight hash functions. ARMADILLO is considered insecure. Lesamnta-LW, PHOTON, and SPONGENT are part of the ISO/IEC 29192-5:2016 standard.

2.2.4 Message Authentication Codes

A message authentication code (MAC) protects the integrity and authenticity of a given message, by generating a tag from the message and a secret key. MAC

Table 2.6 Lightweight hash functions (cryptographic properties)

Name	Ref	Construction	Type of compression function	Message digest (bits)	IS (bits)	Rate (bits)	Preimage	Second preimage	Collisions	Best known attack
ARMADILLO2	[49]	MD	BC with data-depend. bit transpositions	80/128 /160/192/256	256/384 /480/576/768	48/64 /80/96/128	$2^{80}/2^{128}$ $/2^{160}/2^{192}$ $/2^{256}$	$2^{80}/2^{128}$ $/2^{160}/2^{192}$ $/2^{256}$	$2^{40}/2^{64}$ $/2^{80}/2^{96}$ $/2^{128}$	**Practical** free-start collision attack [435] $2^{8.9}/2^{10.2}/2^{10.2}/$ $2^{10.2}/2^{10.2}$
DM-PRESENT	[102]	MD	PRESENT in Davies-Meyer mode	64	64	80 / 128	2^{64}	2^{64}	2^{32}	Multi-differential collision attack [343] $2^{29.18}$ hash comp. on 12 rounds
H-PRESENT	[102]	MD	PRESENT in double-block-length c.	128	128	64	2^{128}	2^{128}	2^{64}	–
GLUON	[77]	T-sponge	Based on Feedback with Carry Shift Register	128/160/224	136/176/256	8/16/32	$2^{128}/2^{160}$ $/2^{224}$	$2^{64}/2^{80}$ $/2^{112}$	$2^{64}/2^{80}$ $/2^{112}$	Preimage attack [469] 2^{105} complexity
Lesamnta-LW	[281]	MD	Type-1 GFN 64−round BC in LW1 mode	256	256	128	2^{120}	2^{120}	2^{120}	–
LHash	[582]	P-Sponge	18-round Feistel-PG	80/96	96/96	16/16	$2^{64}/2^{80}$	$2^{40}/2^{40}$	$2^{40}/2^{40}$	–

(continued)

Table 2.6 (continued)

Name	Ref	Construction	Type of compression function	Message digest (bits)	IS (bits)	Rate (bits)	Preimage	Second preimage	Collisions	Best known attack
PHOTON	[251]	P-Sponge	12 round AES-like permutation	80/128/160/ 224/256	100/144/196/ 256/288	(20,16)/16/ 36/32/32	$2^{64}/2^{112}/$ $2^{124}/2^{192}$ 2^{224}	$2^{40}/2^{64}/$ $2^{80}/2^{112}$ 2^{128}	$2^{40}/2^{64}/$ $2^{80}/2^{112}$ 2^{128}	–
QUARK	[33]	P-Sponge	Grain-like permutation 544/704/1024 rounds	136(u)/176(s) /256(d)	136/176/256	8/16/32	$2^{128}/2^{160}$ 2^{224}	$2^{64}/2^{80}$ 2^{112}	$2^{64}/2^{80}$ 2^{112}	–
sLiSCP	[20]	P-Sponge	Type 2 GFN Simeck	160/160/192	192/256/256	32/64/64	$2^{128}/2^{128}$ 2^{160}	$2^{80}/2^{96}$ 2^{96}	$2^{80}/2^{96}$ 2^{96}	–
SPN-Hash	[144]	P-Sponge	SPN permutation in JH mode 10 rounds	128/256	256/512	128/256	$2^{128}/2^{256}$	$2^{128}/2^{256}$	$2^{64}/2^{128}$	–
SPONGENT	[100]	P-Sponge	PRESENT-like permutation 45/70/90 /120/140 r.	80/128/160 /224/256	88/136/176 /240/272	8/8/16/16 /16	$2^{80}/2^{120}$ $/2^{144}/2^{208}$ $/2^{240}$	$2^{40}/2^{64}$ $/2^{80}/2^{112}$ $/2^{128}$	$2^{40}/2^{64}$ $/2^{80}/2^{112}$ $/2^{128}$	Linear distinguishers [2] on 23 rounds of the SPONGENT permutation

Table 2.7 Lightweight hash functions (implementation properties)

Name	Ref	Techno. (μm)	No. of GEs	Throughput (Kbps @ 100kHz)
ARMADILLO	[49]	0.18	(2923/4353/5406/6554/8653) vs. (4030/6025/7492/8999/11,914)	(27/250/250/25/25) vs. (109/1000/100/100/100)
DM-PRESENT	[102]	0.18	(1600/1886) vs. (2213/2530)	(14.62/22.9) vs. (242.42/387.88)
H-PRESENT	[102]	0.18	2330 vs. 4253	11.45 vs. 200
GLUON	[77]	–	2071/2799.3/4724	12.12/32/58.18
Lesamnta-LW	[281]	0.09	8240	–
LHash	[582]	0.18	817/817/1028	2.40/2.40/(1.81, 0.91)
PHOTON	[251]	0.18	(865/1122/1396/1736/2177) vs. (1168/1708/2117/2786/4362)	(2.82/1.61/2.7/1.86/3.21) vs. (15.15/10.26/20/15.69/ 20.51)
QUARK	[33]	0.18	(1379/1702/ 2296) vs. (2392/2819/4640)	(1.47/2.27/3.13) vs. (11.76/18.18/50)
sLiSCP	[20]	0.065	2271/3019/3019	29.62/44.44/22.22
SPN-Hash	[144]	0.18	(2777 / 4625) vs. (4600 / 8500)	(36.1 / 35.8) vs. (55.7 / 111.3)
SPONGENT	[100]	0.13	(738 / 1060 / 1329 / 1728 / 1950) vs. (1127 / 1687 / 2190 / 2903 / 3281)	(0.81 / 0.34 / 0.4 / 0.22 / 0.17) vs. (17.78 / 11.43 / 17.78 / 13.33 / 11.43)

schemes can be constructed from block ciphers (e.g., CBC-MAC (part of the ISO/IEC 9797-1:1999 standard) or OCB-MAC [504]), from cryptographic hash functions (e.g., HMAC (RFC 2104)), etc. Three lightweight security architectures have been proposed for wireless sensor networks: TinySec [316], MiniSec [382] and SenSec[370]. TinySec and MiniSec recommend CBC-MAC and the patented OCB-MAC, while SenSec recommends XCBC-MAC, for which there is an existential forgery attack [238], and all suggest the use of 32-bit tags. 32-bit security is not enough—the recommended size is at least 64 bits.

Design choices for lightweight MACs include shorter tag sizes, simpler key schedules, small hardware and/or software implementations, better performance on very short messages, no use of nonces, and generation from lightweight block ciphers and hash functions. Some lightweight MACs are listed in Table 2.8, and the best known attacks against these MACs are provided in Table 2.9.

2.2.5 Authenticated Encryption Schemes

Authenticated encryption (AE) schemes combine the functions of ciphers and MACs in one primitive, so they provide confidentiality, integrity, and authentication of a given message. Besides the plaintext and the secret key, they usually accept variable length Associated Data (AEAD schemes), a public nonce, and an optional secret nonce. AD is a part of a message that should be authenticated, but not encrypted.

Lightweight authenticated encryption schemes are presented in Table 2.10, and the best known attacks against these schemes are provided in Table 2.11. Sablier and SCREAM/iSCREAM are considered insecure. The hardware implementation is given with encryption/authentication and decryption/verification functionalities.

2.3 Illustrative Issues in Security Evaluation of Certain Encryption Schemes

As a consequence of the simplicity which makes them lightweight, the security evaluation of lightweight encryption schemes arises as an issue of top importance. However, constraints on chapter space limit our discussion of the security evaluation. Consequently, this section shows only a number of illustrative issues relevant for the cryptanalysis of lightweight encryption techniques. In the first part, a generic approach for security evaluation is discussed, and in the second an advanced dedicated approach is pointed out.

Table 2.8 Lightweight MACs (characteristics)

Name	Ref	Type	Key size (bits)	Block size (bits)	Tag size (bits)	No. of rounds	Techno. (μm)	No. of GEs
Chaskey	[428]	Permutation-based MAC	128	128	≥64	8 (12)		3334.33 GE [356] estimated
LightMAC	[384]	New parallelizable mode with BC and two keys	$2 \times 80/128$	64/128	64/128	Depends of used BC	–	–
SipHash-2-4	[32]	ARX-based keyed hash function	128	256	64	$2 + 4$ 4 fin. rounds	–	–
TuLP	[238]	PRESENT BC in ALRED construction	80/160	64/128	64	14	0.18	2252/2764

Table 2.9 Lightweight MACs (best known attacks)

		Best known attack: data / time complexity
Chaskey	[428]	Differential-linear attack [369] 2^{48}/ 2^{67} on 7 rounds
LightMAC	[384]	–
SipHash -2-4	[32]	–
TuLP	[238]	–

2.3.1 *Reconsidering TMD Tradeoff Attacks for Lightweight Stream Cipher Designs*

We can simply divide the tradeoff attacks against ciphers into two groups, key recovery attacks and internal state recovery attacks. The first tradeoff attack against symmetric ciphers was introduced by Hellman [268] to illustrate that the key length of DES was indeed too short. Hellman prepared several tables containing DES keys. In general, the tradeoff curve is $TM^2 = N^2$ where T is the time complexity and M is the memory complexity. N is the cardinality of the key space. Here, the data complexity $D = 1$ since only one chosen plaintext is used to define a one way function which produces the (reduction of the) ciphertext of the chosen plaintext for a given key. Then, the tables are prepared during the precomputation phase. In practice, one generally considers the point $T = M = N^{2/3}$ on the curve since the overall complexity also becomes $N^{2/3}$. The precomputation phase costs roughly $O(N)$ encryptions. This is a generic attack which is applicable to any block cipher. Therefore, we can say that the security level diminishes to $2k/3$-bit security during the online phase of the Hellman tradeoff attack where k is the key length of a block cipher. However, one must pay a cost equivalent to exhaustive search to prepare the tables during the precomputation phase.

Stream ciphers also suffer from the same affliction by tradeoff attacks in that their keys can be recovered with an effort of $2^{2k/3}$ for each of them during the online phase. Stream ciphers consist of two parts. The initialization part uses an IV and a key to produce a seed value S_0. Then, S_0 is used to produce the keystream sequence through a keystream generator. While a state update function updates the internal states S_i, an output function produces the keystream bits (or words) z_i. It is possible to define a one way function from the key to the first k bits of the keystream sequence by choosing an IV value and fixing it. This is similar to the case of tradeoff attacks on block ciphers with a chosen plaintext. However, the attack may only be mounted on a decryption mechanism since it may not be possible to choose the IV during the encryption. Then, by preparing the Hellman tables, one can recover a key in $2^{2k/3}$ encryptions using $2^{2k/3}$ memory. The precomputation is 2^k. This is similar to the Hellman attack. Therefore, stream ciphers are prone to tradeoff attacks as with block ciphers in the key recovery case.

The other category of tradeoff attacks is aimed at recovering internal states of stream ciphers, rather than keys. Babbage [47] and Golić [236], independently, introduced another type of tradeoff curve $DM = N$ to recover an internal state.

Table 2.10 Lightweight authenticated encryption schemes (characteristics)

Name	Ref	Type	Key (bits)	Nonce (bits)	IS (bits)	Block/rate (bits)	Tag (bits)	Techno. (µm)	No. of GEs
ACORN v3[a]	[581]	SC (LFSR)	128	128	293	1	128	–	–
ALE	[103]	SC (AES, LEX leak)	128	128	256	128	128	0.065	2700
APE	[22]	Sponge (different hash f.) e.g., PHOTON-196	160[d]	36n[d] (opt.)	196[d]	36[d]	160[d]	0.18	1634[d]
ASC-1	[300]	SC (AES, LEX leak CFB-like mode)	256	56	512	128	128	0.065	4964 [103]
Ascon[a]	[186]	Sponge (SPN)	96/128	96/128	320	64 (128 for Ascon-128a)	96/128	0.009	2570 (7970) Ascon-128 [245] (SCA-protected)
C-QUARK	[36]	Sponge (LFSR, NFSR)	256	64	384	64	≥ 64	0.09	4000
FIDES	[87]	Sponge (AES-like, 16 rounds)	80/96	80/96	160/192	10/12	80/96	0.09	793−2876/ 1001−4792
Hummingbird-2	[200]	Hybrid (SPN)	128	64	128	16	128	0.13	2159-3220
Helix	[215]	SC (ARX)	256	128	160	32	128	–	–
Joltik[b]	[304]	tweakable BC (Joltik-BC)	64/80/96/ 128+64/48/96/64 tweak	32/24/48/32	64	64	64	–	2100/2100/ 2600/2600 (estimated)
KETJE[a]	[82]	Sponge (Keccak-f)	$k \leq 182$/ $k \leq 382$	182-$\lvert k \rvert$/ 382-$\lvert k \rvert$	200/400	16/32	64	–	–
LAC[c]	[596]	SC (LBlock-s, LEX leak)	64	80	144	48	64	–	1300 (estimated)

(continued)

Table 2.10 (continued)

Name	Ref	Type	Key (bits)	Nonce (bits)	IS (bits)	Block/rate (bits)	Tag (bits)	Techno. (μm)	No. of GEs
NORX32 v.3[a]	[35]	Sponge (LRX, 4/6 rounds)	128	64	512	320	128	0.018	62,000
NORX8 /NORX16	[34]	Sponge (LRX, 4/6 rounds)	80/96	32	128/256	40/128	80/196	–	1368/2880 (estimated)
Sablier[c]	[594]	SC (LFSR)	80	80	208	16	32	–	1925 (estimated)
SCREAM[b] /iSCREAM	[246]	tweakable BC (SPN+LS-designs)	128+ 128 tweak	8−120	128	128	128	–	–
sLiSCP	[20]	Sponge (Type-2 GFS+Simeck)	80/112/128	80/80/128	192/192/256	32/32/64	80/112/128	0.065	2289/2289/3039
TriviA-v2 /uTriviA	[132]	SC (Trivium-like)	128	128	384	64	128	0.065	21,521 / 16,748

[a]Indicates CAESAR Round 3 candidate
[b]Indicates CAESAR Round 2 candidate
[c]Indicates CAESAR Round 1 candidate
[d]In this case APE is used with PHOTON-196

Table 2.11 Lightweight authenticated encryption schemes (best known attacks)

Name	Ref	Best known attack: data complexity/memory/time complexity
ACORN v3	[581]	–
ALE	[103]	Forgery attack [324]: $2^{40}/-/2^{110}$
APE	[22]	–
ASC-1	[300]	–
Ascon	[186]	Key-recovery attack [371]: $2^{103.9}$ time on 7 out of 12 rounds ASCON-128
C-QUARK	[36]	–
FIDES	[87]	State-recovery/forgery attacks [184]: 1KP/$(2^{15}, 2^{18})$/$(2^{75}, 2^{90})$
Hummingbird-2	[200]	Related key-recovery attack [525]: 24 pairs of related keys/$-/2^{40}$
Helix	[215]	Key-recovery attack [432]: 2^{17} CP/$-/2^{88}$
Joltik	[304]	–
KETJE	[82]	–
LAC	[596]	Differential forgery attack [368] with probability $2^{-61.52}$
NORX32 v.3	[35]	–
NORX8/NORX16	[34]	–
Sablier	[594]	**Practical** state/key recovery attack [213]: $-/-/2^{44}$
SCREAM/iSCREAM	[246]	**Practical** forgery attack [530] with 2 queries
sLiSCP	[20]	–
TriviA-v2/uTriviA	[132]	–

One can pick out the point $D = M = N^{1/2}$ to get an overall complexity of $N^{1/2}$. Then, storing $\sqrt{N}$ internal states with their outputs (keystream parts with an appropriate length), one can recover a keystream used during encryption/decryption if it is loaded in the table. We need roughly $\sqrt{N}$ data to ensure a remarkable success rate. So, it is conventionally adopted that $\sqrt{N}$ should be larger than 2^k as a security criterion just to ensure that the internal state recovery attack through tradeoff is slower than the exhaustive search. This simply means that the internal state size should be at least twice as large as the key size. This extremely strict criterion has played a very crucial role in raising extra difficulties in designing lightweight stream ciphers.

Another highly effective tradeoff attack for internal state recovery is the Biryukov-Shamir attack [91]. This simply makes use of Hellman tables. But, instead of recovering just one specific internal state, it is enough to recover only one of D internal states. Then, preparing just one Hellman table is an optimum solution and the table can contain N/D states. So, the precomputation phase is around $O(N/D)$ and the tradeoff curve is $TM^2D^2 = N^2$ where D is bounded above by $\sqrt{T}$ since the number of internal states contained in just one table is limited to avoid merging of collisions. We can pick out the point on the curve where time and

memory are equal and maximize the data, namely $T = M = N^{1/2}$ and $D = N^{1/4}$. We need $N^{1/2}$ to be larger than 2^k if we want the online phase of the attack to be slower than an exhaustive search. This again simply implies that the internal state size should be at least twice as large as the key size.

The condition on the size of the internal states of stream ciphers makes designing ultralightweight stream ciphers too difficult. Indeed, there are several ultralightweight (say less than 1000 GE) block ciphers recently designed, such as PRESENT [101], LED [252], KTANTAN [126], Piccolo [526], and SIMON/SPECK [65], whereas there are almost no modern stream ciphers with hardware area cost less than 1000 GE.

The security margin for state recovery attacks through tradeoff techniques is k bits, whereas it is much less, $2k/3$ bits, for the key recovery attacks, although any information about the key is assumed to be more sensitive than any information about the internal states. One can produce any internal state once the key is recovered. However, recovery of an internal state may reveal only one session of the encryption/decryption with the corresponding IV. Hence, it seems that the more sensitive data are, contradictorily, protected less against tradeoff attacks!

The security level of tradeoff attacks to recover internal states should be the same as the security level of tradeoff attacks to recover keys, just to be fair. So, the online phase of a tradeoff attack should be at least $2^{2k/3}$ instead of 2^k. Similarly, the precomputation should be not faster than exhaustive search. In this case, $D = M = N^{1/2} \geq 2^{2k/3}$ for the Babbage-Golić attack. Then, N should be at least $2^{4k/3}$. The same bound is valid for Biryukov-Shamir attack since the smallest overall complexity is attained when $T = M = N^{1/2}$.

The precomputation phase of the Biryukov-Shamir attack is roughly N/D; which is simply $N^{3/4}$ when $D = N^{1/4}$. So, the precomputation phase is more than 2^k. This means that it is slower than an exhaustive search. On the other hand, the precomputation phase of the Babbage-Golić attack is M, and hence if the data is restricted to at most $2^{k/3}$ for each key we have $M \geq 2^k$ and hence the precomputation phase will be slower than an exhaustive search.

It seems it is enough to take the internal state size as at least $4k/3$, not at least $2k$, for security against tradeoff attacks. This simply implies that it is possible to design lightweight stream ciphers with much smaller internal states. However, it is an open question how to design stream ciphers with very small internal states. The security is generally based on the largeness of the states.

2.3.2 *Guess-and-Determine Based Cryptanalysis Employing Dedicated TMD-TO*

This section presents an illustrative framework for cryptanalysis employing guess-and-determine and time-memory-data trade-off (TMD-TO) methods using the results of security evaluations of the lightweight stream ciphers Grain-v1, Grain-128 and LILI-128, reported in [415, 416], and [417], respectively.

2.3.2.1 Generic Approach

Certain stream ciphers can be attacked by employing the following approach: (1) Assuming the availability of a sufficiently long sample for recovering an internal state, we develop a dedicated TMD-TO attack which allows recovery of the internal state for a certain segment of the available sample. (2) The dedicated TMD-TO attack is developed over a subset of the internal states in which certain parts of the internal state are preset or algebraically recovered based on the considered keystream segment. Assume that the state size is ν and that certain bits (say β) of the internal state are fixed according to a specific pattern. Then, with this information, for the corresponding keystream segment, we try to obtain some more bits (say γ) of the internal state. The final goal is to recover the unknown bits of the internal state $\delta = \nu - \beta - \gamma$ by employing a suitable TMD-TO attack. Accordingly, the cryptanalysis is based on the following framework:

- preset certain bits of the internal state to a suitable pattern (the all-zeros pattern, for example);
- for a given m-bit prefix (usually an m-zeros prefix) of the keystream segment, algebraically recover up to m bits of the internal state assuming that the remaining internal state bits are known;
- recover the assumed bits of the internal state by employing the dedicated TMD-TO attack.

2.3.2.2 Summary of Cryptanalysis of Grain-v1 Employing Guess-and-Determine and Dedicated TMD-TO Approaches

The internal state of Grain-v1 consists of 160 bits corresponding to the employed nonlinear and linear feedback shift registers NFSR and LFSR, respectively. For a given parameter m, let $\Omega^{(m)}$ be a subset of all internal states where three m-length segments of all zeros exist which implies that the state generates m consecutive zero outputs. Let the vectors $\mathbf{b}^{(i)}$ and $\mathbf{s}^{(i)}$ be the states of the NFSR and LFSR, respectively, at the instant i, $\mathbf{s}^{(i)} = [s_i, s_{i+1}, \ldots, s_{i+79}]$ and $\mathbf{b}^{(i)} = [b_i, b_{i+1}, \ldots, b_{i+79}]$. Let $\mathbf{u}^{(i)}$ be the internal state of Grain-v1, and accordingly, $\mathbf{u}^{(i)} = [\mathbf{s}^{(i)}||\mathbf{b}^{(i)}] = [s_i, s_{i+1}, \ldots, s_{i+79}, b_i, b_{i+1}, \ldots, b_{i+79}]$. For a given parameter m, the set $\Omega^{(m)}$ is the set of internal state vectors defined as follows $\Omega^{(m)} = \{\mathbf{u}^{(i)} | s_{i+25-j} = 0, s_{i+64-j} = 0, b_{i+63-j} = 0 \ , \quad j = 0, 1, \ldots, m-1\}$. Consequently, the number of internal states belonging to $\Omega^{(m)}$ is upper-bounded by 2^{160-3m}.

The internal state recovery is based on the following: Whenever we observe an m-zeros prefix of a keystream segment, we suppose that the segment is generated by an internal state belonging to $\Omega^{(m)}$ and we employ a dedicated TMD-TO attack to check the hypothesis. The complexities of this cryptanalysis and a related one are illustrated in Table 2.12.

Table 2.12 An illustrative numerical comparison of two algorithms for cryptanalysis of Grain-v1

Approach	Time complexity of processing	Space complexity of pre-processing and processing	Time complexity of pre-processing	Required sample
Cryptanalysis reported in [416]	$\sim 2^{54}$	$\sim 2^{78}$	$\sim 2^{88}$	$\sim 2^{72}$
Cryptanalysis reported in [385]	$\sim 2^{53}$	$\sim 2^{78}$	$\sim 2^{78}$	$\sim 2^{82}$

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.