1

Design Trends in Lightweight Ciphers

Meltem SÖNMEZ TURAN [1]
[1]*National Institute of Standards and Technology, Gaithersburg, Maryland*

General-purpose cryptographic algorithms are usually optimized for desktop and server environments, where there are no significant constraints on available resources. The security margins of these algorithms are often more than enough for long-term security (Aumasson 2019), even in cases where the attacker is assumed to have more capabilities and possible actions compared to real-world applications. Algorithms with high security margins (*e.g.*, 50% - 60%) are usually difficult or even impossible to implement in resource-constrained devices (*e.g.*, RFID tags, industrial controllers, sensor nodes) with acceptable performance. The goal of *lightweight cryptography* is to provide cryptographic primitives, schemes and protocols that are optimized for resource-constrained devices having a wide array of performance attributes while also having adequate security margin against known attacks.

In this chapter, we provide an overview of the standardization efforts, desired features, and design trends used in lightweight cryptography [1].

1. Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

1.1. Lightweight Cryptography Standardization Efforts

Over the last decade, a number of standardization efforts and public competitions were organized focusing on lightweight cryptography. The International Organization for Standards (ISO) and the International Electrotechnical Commission (IEC) have published (*i*) seven-part ISO/IEC 29192 *Lightweight Cryptography*, and (*ii*) ISO/IEC 29167, *Automatic Identification and Data Capture Techniques*, approving a number of lightweight algorithms. CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) included a specific use case dedicated to lightweight cryptography. In 2017, the Cryptography Research and Evaluation Committees of Japan published an extensive report that compared the suitability of lightweight designs for various target applications (CRYPTREC 2017). The eSTREAM stream cipher competition, organized by the European Network of Excellence for Cryptology, included a hardware profile for applications with highly restricted resources (Cid *et al*. 2012).

In 2018, the National Institute of Standards and Technology (NIST) officially announced its lightweight cryptography standardization process, soliciting cryptographic algorithms that are suitable for use in constrained environments where the performance of the current NIST cryptography standards, namely Advanced Encryption Standard-Galois/Counter Mode (AES-GCM) for authenticated encryption and Secure Hash Algorithm-2 (SHA-2) for hashing, is not acceptable. In 2019, the process received 57 authenticated encryption with associated data (AEAD) schemes with optional hashing functionality, and 32 of these schemes moved forward for the second round of the process (Turan *et al*. 2019). In 2021, NIST announced ten finalists, namely ASCON, Elephant, GIFT-COFB (COmbined FeedBack), Grain128-AEAD, ISAP, Photon-Beetle, Romulus, Sparkle, TinyJambu, and Xoodyak, for the last round of the process. These finalists provide a wide range of solutions for applications that need high performance and security in constrained environments.

1.2. Desired Features

The main engineering challenge in lightweight cryptography is to find an optimal balance between security, performance, and cost that is specific to target applications and the devices. Next, we highlight some of the desired features of the lightweight algorithms for different use cases.

Security. Similar to general-purpose cryptographic algorithms, security is the most important criterion in lightweight designs. These designs are expected to withstand all known cryptographic attacks, however different threat models might apply due to additional protections that a protocol can provide. For example, in applications that guarantee a proper generation of keys (*e.g.*, using key derivation functions), security against related-key attacks might not be a serious concern. Similarly, in applications

where nonce values are generated randomly, the algorithms may not need to have additional measures for nonce-misuse resistance. Additionally, the constraints of the devices might limit the number of known plaintext/ciphertext available to the attackers, and smaller state sizes and internal counters may be suitable.

Hardware performance. The hardware performance of the algorithms is one of the most relevant features in lightweight cryptography, and the target metrics for hardware implementations can be listed as (*i*) the physical *area* needed for a circuit to implement the algorithm, (*ii*) *latency*, *i.e.*, the amount of time it takes to obtain the circuit's output, (*iii*) *throughput*, *i.e.*, the amount of input data processed per time unit, and (*iv*) the amount of *power* and *energy* needed to use the circuit. In applications such as supply-chain management or anti-counterfeiting, radio-frequency identification (RFID) tags with a small amount of memory are commonly used to identify and track products. Hardware-oriented algorithms with a low memory footprint are preferred for such applications. In automotive industry applications, or applications using fast and secure payments, the latency of the hardware implementations is more relevant. Additionally, in applications using battery-powered devices (*e.g.*, embedded medical devices, or environmental sensors), the algorithms with low energy requirements are more suitable.

Software performance. The software performance of the algorithms in 8-, 16- and 32-bit microcontrollers is also important for lightweight designs. The target metrics for software implementations can be listed as (*i*) execution time, (*ii*) memory requirements (RAM/ROM), (*iii*) the size of the compiled code, and (*iv*) the *throughput*, *i.e.*, the amount of input processed during each clock cycle. In smart home applications using appliances with low-end processing units, software-oriented algorithms that consume a small amount of memory are desired.

Performance in high-end servers. In typical Internet of Things (IoT) applications, a large number of small devices get connected to a central high-end server, and then each device periodically sends short encrypted messages to the server, and the server decrypts these messages in real-time. Although lightweight ciphers are optimized for constrained devices, their performance in high-end systems is also important.

Side-channel resistance. For applications where a potential attacker has physical control of the constrained devices and has the ability to measure side channels such as power consumption or electromagnetic radiation, algorithms that can provide inherent side-channel resistance or lend themselves to efficient countermeasures such as masking may be desirable.

Handling short messages. Most general-purpose cryptography algorithms are optimized for handling long messages to achieve high throughput. However, for applications in automotive and industrial controllers, many small messages are processed. For such applications, algorithms with small computational overhead that are optimized to handle short messages are preferred. Additionally, for applications that may update keys frequently, algorithms that use simple key scheduling algorithms are favored.

Multiple functionalities. All-in-one algorithms that use the same underlying primitive to provide multiple functionalities (*e.g.*, authenticated encryption, hashing or message authentication code (MAC)) provide benefits in hardware area requirements when the application requires more than one functionality.

Simplicity. The security of simple designs is easier to understand through cryptanalysis. Since the security margins of lightweight algorithms are tighter compared to general-purpose algorithms, it is very important to have a good understanding of the designs, which can be achieved using simple components, and round structures.

Flexibility. Designs that can provide flexibility in terms of the selection of internal parameters, such as word sizes, number of rounds, or capacity and rate selection for permutation-based designs, can provide the best tradeoff for applications that use specific message sizes. Additionally, designs that provide implementation flexibility are also preferred.

1.3. Design Approaches in Lightweight Cryptography

The initial attempts to design lightweight ciphers included choices that adversely affected the security of the candidates, such as using smaller key sizes (*e.g.*, 80 bits), having very tight security margins (*e.g.* around 10 %) or using smaller block sizes (*e.g.*, 64 bits, rather than 128) that significantly reduced the amount of data that can be processed securely. Similarly, some designers preferred using over-simplified key schedules that resulted in large weak-key classes. As the body of scientific knowledge on lightweight cryptography increased, new robust lightweight designs that do not compromise security have emerged. Note that the security requirements of the NIST standardization process for lightweight AEAD and hashing schemes are aligned with the generic security requirements expected for general-purpose cryptography algorithms.

The second-round candidates of the NIST standardization process used block ciphers, permutations, tweakable block ciphers, and stream ciphers as underlying primitives to construct AEAD schemes and hash functions (see Table 1.1). Using block ciphers and tweakable block ciphers is usually preferred to achieve a small hardware area by the candidates that offer AEAD only functionality. Tweakable block ciphers tend to use small tweaks (*e.g.*, less than 8 bits) to efficiently achieve domain separation for different types of input data (associated data or message) or differentiating between empty and non-empty inputs or partial blocks.

Permutation-based designs following the sponge construction and its variants are preferred by candidates that aim to achieve better performance in both constrained hardware and software, providing AEAD and hashing functionalities. Sponge designs are attractive for lightweight applications, since they do not need a separate key schedule algorithm. Additionally, the sponge construction and its variants also support using

Candidates providing AEAD only functionality	
Permutation	Elephant, ISAP, Oribatida, SPIX, SpoC, WAGE
Block cipher	COMET, GIFT-COFB, HyENA, mixFeed, Pyjamask, SAEAES, SUNDAE-GIFT, TinyJambu
Tweakable Block Cipher	ForkAE, ESTATE, LOTUS-AEAD & LOCUS-AEAD, Romulus, Spook
Stream cipher	Grain-128AEAD
Candidates providing AEAD and hashing functionalities	
Permutation	ACE, ASCON, DryGASCON, Gimli, KNOT, ORANGE, PHOTON-Beetle, SPARKLE, Subterranean 2.0, Xoodyak
Block cipher	Saturnin
Tweakable Block Cipher	SKINNY-AEAD & SKINNY-HASH

Table 1.1. *Underlying primitives of the second-round candidates*

different underlying permutations, which provides flexibility to adjust the number of rounds during processing different blocks (*e.g.*, smaller number of rounds to process associated data blocks). The flexibility to adjust the rate and capacity is also useful for making additional optimization when the message size is fixed.

Compared to general-purpose block ciphers, simpler nonlinear and linear layers are preferred in lightweight block cipher designs. For nonlinear layers, 4-bit or 5-bit S-boxes (rather than 8-bit ones) are commonly used to achieve significant area savings. For example, the smallest AES S-box implementation using standard gates requires 113 gates (32 `AND`, 77 `XOR`, 4 `XNOR`), whereas the S-box used for ASCON can be implemented using only 22 gates (5 `AND`, 11 `XOR`, 6 `NOT`). Low-degree S-boxes that can be implemented with a small number of `AND` gates are also easier to mask with a small overhead for side-channel protection. Simple linear layers constructed with bit permutations are more beneficial than complex linear layers in hardware-oriented designs. However, designs with very simple rounds need to iterate over a large number of rounds to achieve security.

It is still challenging to design robust ciphers with provable security properties that would be suitable for a wide range of real-world applications. However, partially due to the standardization efforts, there have been many improvements in the understanding of lightweight designs over the last decade.

1.4. Bibliography

Aumasson, J.-P. (2019), 'Too much crypto', Cryptology ePrint Archive, Report 2019/1492. https://eprint.iacr.org/2019/1492.

Cid, C., Robshaw, M., Babbage, S., Borghoff, J., Velichkov, V. (2012), 'The eSTREAM Portfolio in 2012', Technical Report, Cryptographic Technology Guidelines. https://www.ecrypt.eu.org/stream/.

CRYPTREC (2017), 'CRYPTREC Cryptographic Technology Guideline (Lightweight Cryptography)', Technical Report, Cryptographic Technology Guidelines. https://www.cryptrec.go.jp/en/tech_guidelines.html.

Turan, M. S., McKay, K., Ç. Çalık, Chang, D., Bassham, L. (2019), 'NISTIR 8268 Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process'. https://csrc.nist.gov/publications/detail/nistir/8268/final.