Universitat Politècnica de Catalunya

DATA PROTECTION

LIGHTWEIGHT CRYPTOGRAPHY

Authors:
Oriol Lalaguna Royo
Sofia Tsagiopoulou

Course 2022-2023

Contents

1 Introduction

Today there is an explosion of Internet of Thing (IoT) devices in use across every industry, transforming our daily life and the way we do business. The more we use these devices, the more they need to comply with principles, such as Security by Design and Privacy by Design. However, these devices have extremely limited computational power and storage capacity. Given that the most popular cryptosystems need large calculations and storage to provide some security assurances, there is a rising need to apply various security measures in contexts with extremely limited compute capabilities. Lightweight Cryptography can be the solution here. While employing the least amount of processing and storage resources possible, lightweight cryptography tries to provide a respectable level of security. Nowadays, we see lightweight cryptography in the product roadmap of various IoT devices, and the number of applications is rising quickly.

Small computer device deployments, such as those of RFID tags, industrial controllers, sensor nodes, and smart cards, are becoming more widespread. Device connectivity and the amount of data shared over the air are growing exponentially.

Information integrity and confidentiality are maintained via cryptographic techniques. Small gadgets have replaced desktop computers, raising a variety of new security and privacy issues. IoT does not support traditional cryptographic techniques well, due to the limited power, storage and computational resources. Therefore, applying traditional standards to small devices is difficult.

The increase of use of IoT devices is growing the attack surface. Data collection from devices might be a target of cyberattacks in IoT systems that use data in the real world. As a result, encryption-based countermeasures are becoming increasingly important. Lightweight cryptography is a type of encryption with a tiny footprint and/or minimal computational complexity. Its goal is to broaden the uses of cryptography on restricted devices, and it is now undergoing worldwide standardization and guidelines compilation. Authenticated encryption that achieves both secrecy and integrity has received significant attention, and a

Needs more explanation as you talk about products not neccesarily known by the reader -> CAESAR technology challenge has been established. TWINE is a lightweight block cipher created by NEC, and OTR is an authenticated encryption technique that passed the CAESAR second-round selection [1].

While AES and SHA function well together within computer systems, they suffer in an Internet of things (IoT) or embedded environment because they use too much computing power, physical space, and battery capacity. A wide variety of lightweight cryptographic primitives have been suggested and utilized over resource-limited devices in the previous decade. Both national (NIST) and international (ISO/IEC) organizations detail a variety of lightweight cryptographic approaches that might be beneficial in IoT and RFID devices. The device spectrum is defined as follows:

- Conventional cryptography. Servers and Desktops; Tablets and smart phones.
- Lightweight cryptography. Embedded Systems; RFID and Sensor Networks.

2 Lightweight Cryptography

Lightweight Cryptography is an encryption method that features a small footprint and/or low computational complexity. It is aimed at expanding the applications of cryptography to constrained devices and its related international standardization and guidelines compilation are currently underway.

The primary limitations we face for lightweight cryptography typically have to do with power consumption, gate equivalents (GEs), and timing. When using passive RFID technology, the power supply is not connected to a battery; instead, the chip must be powered by radio wave energy. Thus, the power consumption associated with any cryptography functions on an RFID device is likely to be severely constrained, along with the timing constraints and the number of gates employed. Even if an RFID device has a battery attached to it (active RFID), it may be challenging to replenish the battery. Therefore, it is frequently necessary to reduce power consumption.

Recently, the IoT has seen the development of lightweight symmetric cryptography. The latter includes block/streaming ciphers like PRESENT, SPONGENT, and Quark, as well as hash functions and MACs like Marvin and Quark. Post-quantum cryptography lattices and codes, as well as number-theoretic cryptography, such as ECC, PBC, etc., are examples of asymmetric cryptography that can be utilized for the Internet of Things [4]. Due to the fact that the majority of IoT devices operate in multitask mode, software speed is essential for lightweight cryptography. Existing lightweight solutions like Chaskey, FLY, LEA, SPARX, etc. exhibit favorable evaluation findings. Considerations should be made for the encryption types, block size, key size, applicable attacks, etc. in the IoT scenario.

In cryptanalysis, the size of the key, the block, and the tag are typically taken into account for lightweight cryptography (especially for multi-key assaults, precomputational power, brute-force attacks, etc.) The ability to operate on devices with low resources is critical.

Several lightweight cryptographic protocols, methods, and primitives have been standardised as ISO/ICE 29121 throughout the last decade. For secure communication, lightweight primitives have been integrated into established protocols such as IPSec and TLS. In addition, a variety of embedded cryptography libraries such as wolfSSL, CyaSSL, sharkSSL, RELIC, and others have been developed [2].

3 Applications

The lightweight cryptography became popular due to the implementations on limited resources devices. Symmetric and asymmetric cipher implementations, that target on this kind of devices, are classified into software and hardware implementations.

The two aforementioned classifications are divided based on the following characteristics.

For hardware applications:

- Physical space: a circuit that implements the primitive requires this.
- Throughput: the volume of data processed per unit of time.
- Latency: the amount of time it takes to get the circuit's output after the input is set.
- Power consumption: how much energy is required to operate the circuit.

For software applications:

- RAM: the amount of memory utilized for each algorithm evaluation.
- Code size: the length of the executed, compiled source code.

- Throughput: the volume of data processed per clock cycle.

When selecting the proper security algorithm to be implemented in resource-constrained devices, these characteristics must be respected [3].

Target use cases for this cryptography include hardware encrypted data storage devices, low-cost and low-consumption sensor data transmission, RAIN RFID tags for anti-counterfeiting solutions, Internet of Things (IoT) devices, wearable technology, and low power wireless sensor networks.

But the applications where it is used the most are shown in the graph below.

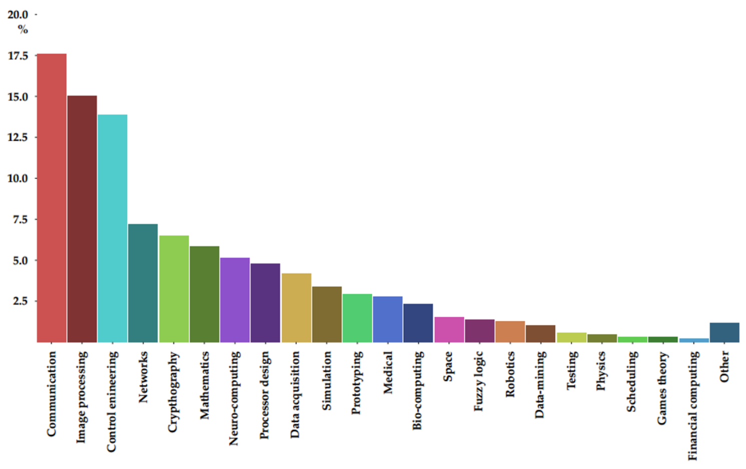


Figure 1: Lightweight cryptography applications common use

Communication, Image processing and Control engineering are the applications that make the most use of this technology.

4 IoT and Lightweight Cryptography Security

The biggest security risk posed by IoT systems, compared to traditional IT systems, is the possibility of cyberattacks when using these devices to collect data from the real world. For instance, by gathering data from a large number of sensors put in production equipment, interpreting it, and conducting autonomous control in real time, IoT can be help a plant to dramatically improve productivity and maintainability. If sensor data were to be manipulated during this process, inaccurate analysis results would be generated and incorrect control would follow. This would cause significant harm. Even if there isn't an issue right now, it's important to think about the impact of any potential dangers down the road.

When encryption is used on sensor equipment, data security for confidentiality and integrity is implemented. This can be a powerful defense against threats. Secure encryption can be used even on low-resource devices thanks to the function of lightweight cryptography.

On the data link layer of communication systems, like the cellphones, encryption is already used as normal practice. Encryption in the application layer is still effective in this situation to ensure security independent of the communication system and to provide end-to-end data protection from the device to the server. Then, encryption must be deployed on unused resources and at the processor processing the application; as a result, it should ideally be as light as feasible [4].

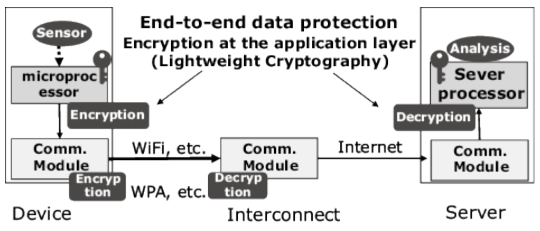


Figure 2: Lightweight cryptography applications example

5 Challenges

5.1 Challenges in IoT

Encryption methods were created for traditional computing devices (personal computers) because they need a lot of computational resources and power. Thus, they cannot be employed directly in IoT systems. However, an effort is made to establish a secure connection on IoT systems with low resources by using lightweight cryptographic algorithms.

The main difficulties in applying traditional cryptography on IoT devices are:

- Limited memory (registers, RAM, ROM): IoT devices often have little processing power, little memory and storage space on flash or RAM, and restricted network protocol support in order to maximize battery life. IoT device manufacturers and software developers have a big hurdle when trying to implement intricate security protocols with a 64KB to 640KB memory storage.

- Lower power sources and capacity: Power is a key component of IoT devices, in which the battery won't be able to fully charge. Thus, the capacity is limited and the risk of network failure is increased. A major obstacle in the development of IoT devices and their communication protocols is energy efficiency. Energy supplies are therefore crucial, particularly in sensor devices that are battery-powered.

- Small physical area to implement the assembly: Since the majority of IoT devices are small, the embedded technology must also be compact. The quantity of silicon wafer space needed for memory processing must also be maintained to a minimum because it raises expenses.

- Real-time response: Real time decision making is essential to many IoT functionalities. IoT devices have the ability to collect data from sensors and devices, process it, and then send it to a server or a third party for analysis. Real-time results is a necessity for organizations.

5.2 Challenges that Lightweight Cryptography overcomes

When designing cryptographic algorithms, there is a trade-off between performance and the resources needed for a particular security level. Performance can be described in terms of throughput, latency, and power and energy consumption. Gate area, gate equivalents, or slices are typically used to condense the amount of resources needed for a hardware implementation. This can be seen in software through register, RAM, and ROM consumption.

Lightweight cryptography targets a very broad range of resource-constrained devices, including IoT end nodes and RFID tags. These devices can be implemented in both hardware and software and use a number of communication methods. The size, latency, or throughput, and energy consumption of the traditional cryptographic algorithms make their implementation, in environments with restricted resources, particularly challenging. On devices with limited resources, the lightweight cryptography decreases implementation costs and energy consumption. The goal of lightweight cryptography is to offer security solutions that can operate on devices with limited resources by consuming less memory, less computational power, and less energy [5]. Compared to traditional cryptography, lightweight cryptography is anticipated to be quicker and easier.

As a result of the nature of many restricted devices, power and energy consumption are significant measurements. Power may be crucial for equipment that harnesses ambient energy. An illustration would be an RFID chip that powers its internal circuit via the electromagnetic field sent out by a reader. In battery-operated systems with a set amount of stored energy, energy consumption (i.e., power consumption during a specific time period) is also crucial. On top of that, some devices' batteries could be hard to recharge or replace after being used, if not impossible. It should be noted that a number of variables, including the threshold voltage, the clock frequency, and the implementation technology, affect power consumption.

Furthermore, latency is particularly important. If we think of some some real-time applications, such as the automotive sector, there is a necessity for extremely quick response times for parts like steering, airbags, or brakes. Latency is the amount of time between the initial request for an operation and the creation of the output. The delay between the initial request for the encryption of a plaintext and the response that returns the appropriate ciphertext, for instance, is the latency of an encryption operation.

Finally, the code size, memory consumption (RAM), and energy consumption are key performance indicators in hardware implementation of the lightweight cryptographic primitives. The precise type of circuit (like a clock), memory, storing of the internal states, and key states should all be taken into account while evaluating lightweight cryptography. On top of that, in software implementation case, the implementation size and RAM consumption and the throughput (bytes per cycle) are preferable metrics for the lightweight applications [6]

6 Symmetric Lightweight Cryptography

The motivation of lightweight cryptography is to use less memory, computing resources and power supply to provide security solutions that can work over resource-limited devices. The lightweight cryptography is expected to be simpler and faster compared to conventional cryptography.

While creating a lightweight version of cryptographic algorithms, particular focus is placed on symmetric cryptographic systems, such as symmetric block, stream, and message authentication codes. On the other hand, using public key cryptography systems in lightweight cryptographic systems is not seen as appropriate, because their implementation directly relates to computing resource and power. In any case, a lightweight variant of the cryptographic algorithm does not mean a weak cryptographic algorithm. Instead, lighweight cryptography aims to create cryptographic algorithms that achieve the best possible balance between performance, security, and resource usage. To achieve that lightweight algorithms are designed to use smaller internal states, short blocks, and key sizes. In more detail 3,

- Lightweight Block Ciphers :
  1. Small block size: reduces the length of the plaintexts to be encrypted. For example, 64 or 80 bits, rather than 128.
  2. Smaller key sizes: use small key sizes (less than 96 bits) for efficiency

3. Simple rounds: making the components and operations that are used in lightweight block ciphers simpler.
4. Simpler key schedule: reducing the memory, latency and the power consumption of implementations This may enable attacks using related keys, weak keys, known keys or even chosen keys. When this is the case, it is necessary to ensure that all keys are generated independently using a secure key derivation function

- Lightweight Hash Functions : when collision resistance is not a requirement, we use lightweight hash function which uses smaller inputs and smaller outputs.
- Lightweight Message Authentication Codes : A message and a secret key are combined to create a message authentication code (MAC). MAC creates a tag that is used to confirm the message's authenticity. For common applications, tag sizes of at least 64 bits are advised.
- Lightweight Stream Ciphers

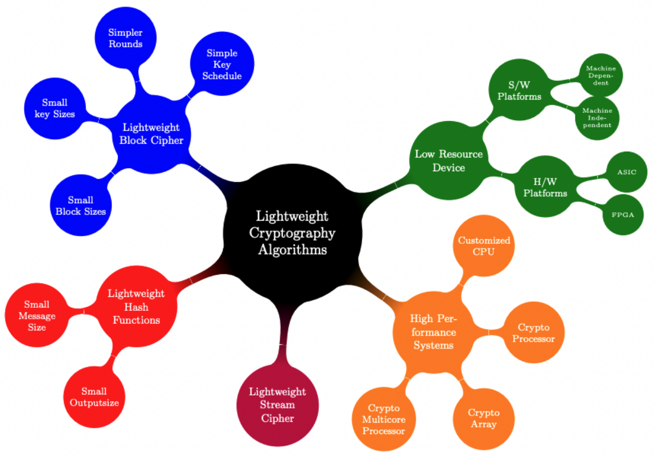


Figure 3: Lightweight Symmetric Cryptography [7]

6.1 Lightweight Cryptographic Algorithms

There are plenty of Lightweight cryptographic algorithms. The design of these algorithms focuses on specific characteristics. The main features to consider are [8]:

1. Security strength: Efficient security. The algorithm should be not easy to break. More specifically, the security strength should be at least 112 bits.
2. Flexibility: An algorithm should be able to be implemented effectively on a variety of platforms. Lightweight algorithms should enable implementations using fewer resources than other algorithms.
3. Low overhead for multiple functions: Different primitives, such as a hash function and block cipher, can also share logic. Therefore, they reduce the resources needed to implement multiple algorithms in the same device.

4. Ciphertext expansion: The smaller the encrypted message, the better. Long ciphertexts consume more power to transmit and storage.

5. Resistance to attacks: Implementations have a variety of ways to leak sensitive data, especially data related to the key or plaintext. Side channel attacks exploit implementation-specific characteristics, such as timing, power usage, and electromagnetic emissions, during the execution of cryptographic operations [9]. In this way, attackers can gather this sensitive data. Furthermore, another technique to compromise data is by introducing faults in the computation. Fault attacks are able to recover this sensitive information. On top of that, key-realated attacks is a key issue. These attacks allow an attacker to gain information about a key by performing operations using multiple unknown keys that have a specific relation. This is a problem mostly in protocols that do not generate their keys independently and randomly [10].

Some examples of symmetric Lightweight agorithms are: Grain, Trivium [11], Mickey [12], Chaskey [13], TuLP, PHOTON, Quark, SPONGENT [14], RC5, TEA, PRESENT [15]. This research project focuses on PRESENT, SPONGENT, Chaskey to give more details abouth their design.

6.1.1 PRESENT

Present serves as an illustration of an SP-network and has 31 rounds. Each of the 31 rounds consists of an xor operation to introduce a round key Ki, where K32 is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a single 4-bit S-box S which is applied 16 times in parallel in each round. The supported key lengths are 80 and 128 bits, and the block length is 64 bits. It is recommended to use 80-bit keys for low power devices. This way, the algorithm meets the security measurements providing more than enough security for the low-security applications commonly required in tag-based deployments. The algorithm is described below 4.

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,$K_{32}$)
```

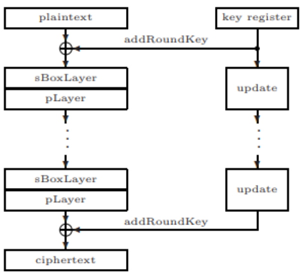


Figure 4: PRESENT.

6.1.2 SPONGENT

Spongent is a lightweight hash function algorithm with hash sizes of 88, 128, 160, 224, and 256 bits. It is based on a sponge construction instantiated with a present-type permutation, following the hermetic sponge strategy. Given a finite number of input bits, it produces an n-bit hash value. The sponge construction proceeds in three phases 5:

- Initialization phase: Padding with the value 1 followed by the required zeros.

- Absorbing phase: The blocks of the input message are xored with the first bits of the state.

- Squeezing phase: The first r bits of the state are returned as output.

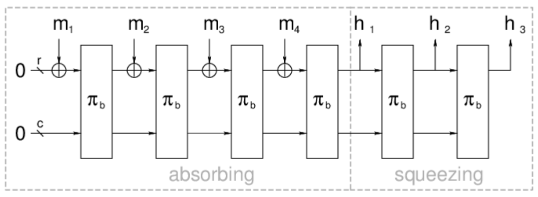


Figure 5: SPONGENT PHASES.

6.1.3 Chaskey

Chaskey is a Message Authentication Code (MAC) technique, effective for 32-bit microcontrollers. It is designed for applications that need 128-bit security but are unable to implement typical MAC algorithms due to strict constraints for speed, energy use, or code size. Chaskey is a permutation-based MAC algorithm that uses the Addition-Rotation-XOR (ARX) design methodology. We establish the security of an underlying Even-Mansour block cipher as the foundation for our proof that Chaskey is secure in the conventional model. Chaskey is built to operate efficiently on a variety of 32-bit microcontrollers. Chaskey implementation has been tested on ARM Cortex-M3/M4 and ARM Cortex-M0. It runs at a speed of 7.0 cycles per byte, as opposed to 89.4 cycles per byte for AES-128-CMAC. Furthermore, Chaskey shows 16.9 cycles per byte and 136.5 cycles per byte for AES-128-CMAC, respectively.

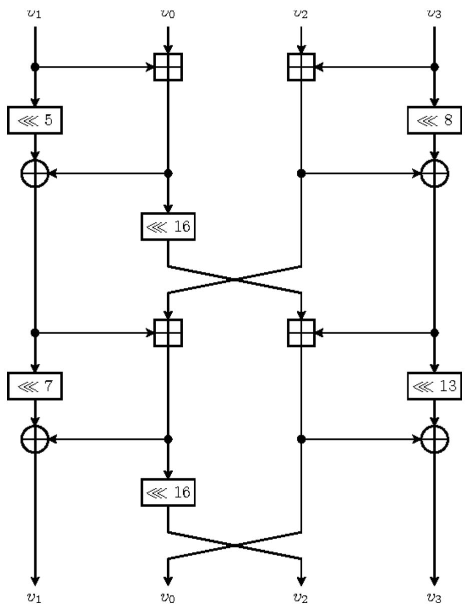


Figure 6: Chaskey.

7 Asymmetric Lightweight Cryptography

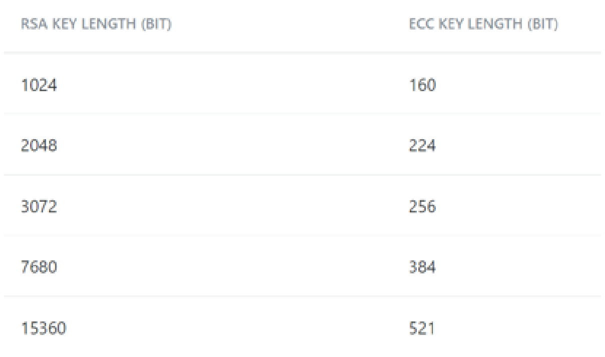
As IoT applications are becoming more popular, there is n increasing need for more security and privacy measurements. Thus, many researches nowadays try to implement Asymmetric Lightweight Cryptography [16], [17], [18].

In general, Asymmetric cryptography is more power costly as its algorithms of the key generation, encryption, and decryption are more complex. On top of that, more memory storage is required.

However, Asymmetric cryptography has been used in encryption, signatures, digital envelopes and key establishment to provide confidentiality, integrity, authentication, nonrepudiation, availability and access control services.

7.1 Elliptic Curve Cryptography

The most popular Lightweight Asymmetric Algorithm is the Elliptic Curve. Elliptic Curve cryptography (ECC) is a powerful approach to cryptography and an alternative method from the well-known RSA. It is used for public key encryption by utilizing the mathematics behind elliptic curves to generate security between key pairs. ECC can provide the same level of security as RSA with a much smaller key size 7. Fact tha makes ECC be more suitable for low resources devices, than RSA.



RSA KEY LENGTH (BIT)	ECC KEY LENGTH (BIT)
1024	160
2048	224
3072	256
7680	384
15360	521

Figure 7: RSA key length and security level VS ECC key length and security level.

Apart from the benefit of the key size. Breaking a 228-bit RSA key would take less energy than what is needed to boil a teaspoon of water. Alternatively, breaking a 228-bit ECC key would require more energy than it would take to boil all the water on earth. Let's assume two points on the curve are P and Q, such that kP=Q, where k is a scalar. If by chance an intruder gains access to the values of P and Q, then it is not easy for the intruder to compute the value of k since computing the value of k is impractical from the curve 8. Which is the Elliptic Curve Discrete Logarithm Problem.

The equation of elliptic curve on which all the mathematical operations are defined is: y2=x3+ ax + b where. An elliptic curve represents a set of pair of (x,y) satisfying the equation defined where value of 'a' and 'b' gives a different elliptic curve. All point pairs (x,y) and point 'O' at infinity lies on the elliptic curve.

In elliptic curve cryptography, private key generated by choosing a random number 'r' from the range 1, . . . , n-1 (here n defines the order). The public key is calculated as P=rG (where G is the generator point of the subgroup).

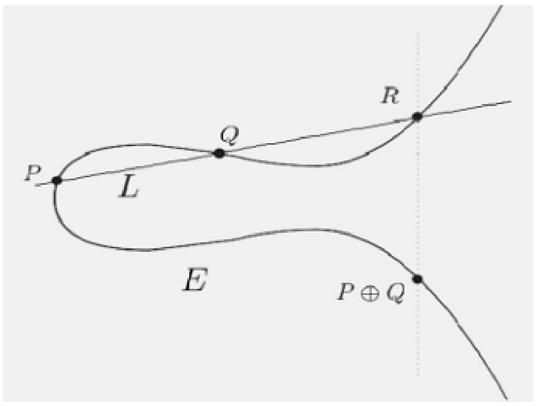


Figure 8: ECC.

All public key cryptosystems have some underlying mathematical operations. For instance, RSA has exponentiation. Elliptic Curve has the point multiplication, repeated addition of two points given as $KP = P + P + P + P...$. ECC supports other operations too. The operations of ECC are hierarchy depending on the operation field 9. The group operations on the elliptic curve and the scalar multiplication influences the number of clock cycles required for encryption [19].

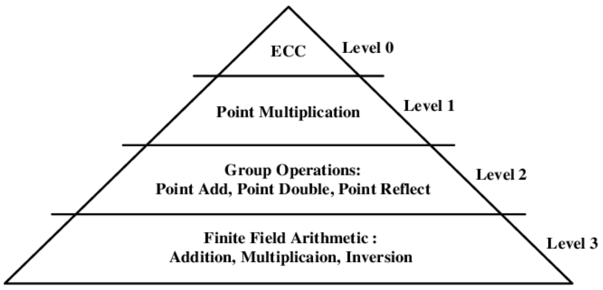

ECC
Level 0
Point Multiplication
Level 1
Group Operations:
Point Add, Point Double, Point Reflect
Level 2
Finite Field Arithmetic :
Addition, Multiplicaion, Inversion
Level 3

Figure 9: ECC operations.

7.2 ECDH and ECDS

The two main applications of ECC are in digital signing and in key exchange. Actually, elliptic curves simply give an unlimited supply of additive abelian groups in which the discrete logarithm issue is extremely difficult, so that any cryptographic protocol based on exploiting a group may be utilized with elliptic curves. Two well known cryptographic methods based on groups are Elliptic Curve Diffie Hellman Key Exchange (ECDH) and digital signature generation Elliptic Curve Digital Signature (ECDS) Algorithm [20]. These two algorithms work as the images below describe 10 ,11.

7.3 Applications of ECC

Most devices require a dedicated cryptographic processor, such as a hardware security module or a smartcard, to execute cryptographic functions properly. In the case of a smartcard, these are already rather small. Small enough to be inserted as a microchip in a credit card.

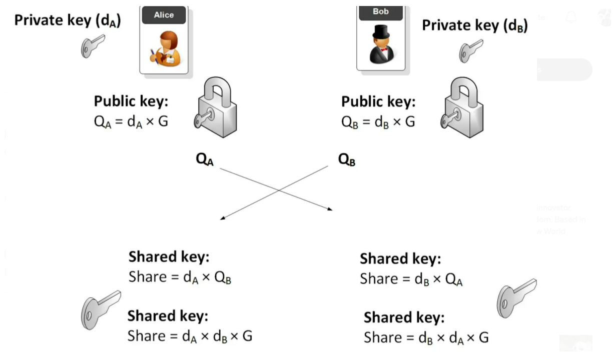

Private key (d_A)
Alice
Bob
Private key (d_B)
Public key: $Q_A = d_A \times G$
Public key: $Q_B = d_B \times G$
Q_A
Q_B
Shared key: Share = $d_A \times Q_B$
Shared key: Share = $d_B \times Q_A$
Shared key: Share = $d_A \times d_B \times G$
Shared key: Share = $d_B \times d_A \times G$

Figure 10: ECDH

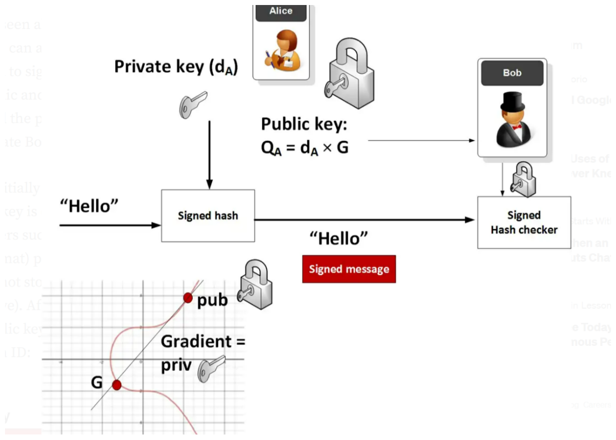


Figure 11: ECDS

Because ECC has such a low data demand, we can utilize smaller, faster, and cheaper cryptoprocessors, which greatly widens the range of devices with which it may be incorporated. As a result, ECC has become popular in IoT industry, for instance, issuing certificates to IoT devices. Furthermore, ECC improves device performance while decreasing power consumption, making it appropriate for a wide range of applications, including IoT and wireless sensor network (WSN) devices. To protect the secrecy and security of data and communications, ECC must be implemented rigorously.

For small devices like IoT devices that have low computational power and limited storage the ECC can be used in:

- Wireless communication devices.
- Smart cards.
- Storage that need to handle a lot of encrypted data. [21]

8 Conclusion

The Internet of Things (IoT) is now widely used, with many resources being limited and small devices being widely deployed and wirelessly talking with one another and the Internet at large. The lightweight cryptography is used to address this issue with respect to security requirements and properties of low resource availability. It is also an approach headed for smart security applications in low-power devices. Specifically, in IoT applications, provision of high-level security is challenging due to their inbuilt low-speed processors and low memory modules. Therefore, much lighter versions of cryptography are researched to suggest durable security solutions. This article's goal was to provide the best explanation possible of the traits, developments, and applications relevant to this subject.

References

[1] *Lightweight Cryptography Applicable to Various IoT Devices.* URL: https://www.nec.com/en/global/techrep/journal/g17/n01/170114.html.

[2] K. Minematsu. "Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions". In: *EUROCRYPT* (2014).

[3] *Hardware Implementation of Secure Lightweight Cryptographic Designs for IoT Applications.* URL: https://www.hindawi.com/journals/scn/2020/8860598/.

[4] T. Suzaki et al. "A Secure Cloud Storage using ECC-Based Homomorphic Encryption". In: *SAC* (2012).

[5] Salim Ganiev and Zarif Khudoykulov. "Lightweight Cryptography Algorithms for IoT Devices: Open issues and challenges". In: *2021 International Conference on Information Science and Communications Technologies (ICISCT).* 2021, pp. 01–04. DOI: 10.1109/ICISCT52966.2021.9670281.

[6] *Report on Lightweight Cryptography.* URL: https://csrc.nist.gov/CSRC/media/Publications/nistir/8114/draft/documents/nistir_8114_draft.pdf.

[7] Nilupulee A. Gunathilake, William J. Buchanan, and Rameez Asif. "Next Generation Lightweight Cryptography for Smart IoT Devices: : Implementation, Challenges and Applications". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019, pp. 707–710. DOI: `10.1109/WF-IoT.2019.8767250`.

[8] Vishal Thakor, Mohammad Abdur Razzaque, and Muhammad Khandaker. *Lightweight Cryptography for IoT: A State-of-the-Art*. June 2020.

[9] Maha Prabu and R. Shanmugalakshmi. "An Overview of Side Channel Attacks and Its Countermeasures using Elliptic Curve Cryptography". In: *International Journal on Computer Science and Engineering* 2 (Aug. 2010).

[10] Gilles Piret and Jean-Jacques Quisquater. "Related-Key and Slide Attacks: Analysis, Connections, and Improvements (Extended Abstract)". In: (July 2002).

[11] Christophe De Cannière and Bart Preneel. "Trivium". In: *New Stream Cipher Designs: The eSTREAM Finalists*. Ed. by Matthew Robshaw and Olivier Billet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 244–266. ISBN: 978-3-540-68351-3. DOI: `10.1007/978-3-540-68351-3_18`. URL: `https://doi.org/10.1007/978-3-540-68351-3_18`.

[12] Steve Babbage and Matthew Dodd. "The MICKEY Stream Ciphers". In: *New Stream Cipher Designs: The eSTREAM Finalists*. Ed. by Matthew Robshaw and Olivier Billet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 191–209. ISBN: 978-3-540-68351-3. DOI: `10.1007/978-3-540-68351-3_15`. URL: `https://doi.org/10.1007/978-3-540-68351-3_15`.

[13] Nicky Mouha et al. "Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers". In: *Selected Areas in Cryptography – SAC 2014*. Ed. by Antoine Joux and Amr Youssef. Cham: Springer International Publishing, 2014, pp. 306–323. ISBN: 978-3-319-13051-4.

[14] Andrey Bogdanov et al. "spongent: A Lightweight Hash Function". In: *Workshop on Cryptographic Hardware and Embedded Systems*. 2011.

[15] A. Bogdanov et al. "PRESENT: An Ultra-Lightweight Block Cipher". In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466. ISBN: 978-3-540-74735-2.

[16] Lejla Batina et al. "Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks". In: *Security and Privacy in Ad-Hoc and Sensor Networks*. Ed. by Levente Buttyán, Virgil D. Gligor, and Dirk Westhoff. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 6–17. ISBN: 978-3-540-69173-0.

[17] Ling-Yu Yeh et al. "An Energy-Efficient Dual-Field Elliptic Curve Cryptography Processor for Internet of Things Applications". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.9 (2020), pp. 1614–1618. DOI: `10.1109/TCSII.2020.3012448`.

[18] Parwinder Kaur Dhillon and Sheetal Kalra. "Elliptic curve cryptography for real time embedded systems in IoT networks". In: *2016 5th International Conference on Wireless Networks and Embedded Systems (WECON)*. 2016, pp. 1–6. DOI: `10.1109/WECON.2016.7993462`.

[19] Jalaja Swamy. "Licensed Under Creative Commons Attribution CC BY ASIC Architectures for Implementing ECC Arithmetic over Finite Fields". In: *International Journal of Science and Research (IJSR)* (June 2013), pp. 2319–7064.

[20] Mahboob A. “EFFICIENT HARDWARE AND SOFTWARE IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY”. PhD thesis. Jan. 2004.

[21] Daya Sagar Gupta and Gosta Pada Biswas. “A Secure Cloud Storage using ECC-Based Homomorphic Encryption”. In: *Int. J. Inf. Secur. Priv.* 11 (2017), pp. 54–62.