Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 2
Publication Date: April, 25, 2022
DOI:10.14738/tmlai.102.11939.
Galliera, R., & Bagui, S. (2022). An Introduction to Data Encryption and Future Trends in Lightweight Cryptography and Securing IoT Environments. Transactions on Machine Learning and Artificial Intelligence, 10(2). 14-26.

An Introduction to Data Encryption and Future Trends in Lightweight Cryptography and Securing IoT Environments

Raffaele Galliera
Department of Computer Science, University of West Florida, USA

Sikha Bagui
Department of Computer Science, University of West Florida, USA

ABSTRACT

This paper presents an overview of the basic concepts of cryptography and encryption. The work aims at presenting the main concepts and concerns of encryption on a high-level of abstraction, allowing non-domain expert readers to navigate through these topics. Less traditional arguments are also shown, from the relevance of Key Management Services with its usage in Envelope Encryption, to Zero Knowledge proofs and their innovative applications. The crucial importance of securing communications between IoT devices and widely used algorithms to do so, are also discussed.

Keywords: Data Encryption, Cryptography, Zero Knowledge, Network Security, Key Management Service, Interactive Proofs, Non-Interactive Proofs

INTRODUCTION

By the day the number of devices connected to a network is increasing, and with that the data flowing to the internet is building exponentially. It has been estimated that by 2025, more than 75 billion devices will be connected to the Internet [1]. This implies a greater responsibility for both the academic and industrial research world. Numerous attacks during the last decade have targeted Internet of Things (IoT) networks, demonstrating an extremely effective exploitation of these devices for performing massively scaled DDoS attacks. One of the most notorious examples is the Mirai Botnet [2], which in October 2016, brought down huge portions of internet services, from streaming to banking, exploiting IoT device vulnerabilities present in digital cameras and DVR players. Computers in a network would get infected by a malware, i.e. Mirai, and computers would continually search the internet for vulnerable IoT devices and then use known default usernames and passwords to log in, infecting them. Very recently, a new Botnet inspired by Mirai has been reported compromising over 300,000 IoT devices [3]. Like the Mirai story, we find many other even scarier episodes of dangerous IoT device vulnerabilities. For example, in 2017, CNN reported that some cardiac devices could be hacked, putting at risk the life of patients [4]. Hence, since IoT devices are so widely used in so many fields in our everyday lives, their vulnerabilities lead to the necessity of developing stronger authentication methodologies and solid encryption systems in order to keep access to the data and the integrity of the information as safe as possible.

The rest of this paper is organized as follows. Section 2 discusses basics of Cryptography, including the importance of data encryption and what it means to keep information safe and secure. Symmetric and asymmetric data encryption, the two main types of encryption illustrating the basics of well-known algorithms, Advanced Encryption System (AES) [5] and Rivest-Shamir-Adleman (RSA) [6] are also discussed. In this section, a summarization of the latest National Institute of Standards and Technology (NIST) report [7] of the in-progress standardization process of Lightweight Cryptography for resource-constrained devices, is also presented. This section also introduces some of the concepts of Key Management Services to protect secrets and encryption keys. Section 3 brings in the topic of Zero-Knowledge, a non-traditional way of performing proof of knowledge, which can be applied, for example, to authentication. This sections also reviews the crucial role of present day IoT devices and the importance of establishing secure communications. Finally, section 4 presents the conclusions.

THE BASICS OF CRYPTOGRAPHY

Cryptography, from the Ancient Greek *Kryptos* meaning "hidden, secret" and *Graphein* meaning "writing", in the modern context of applications in information technology, is still the ancient art of securing communications. Though once physically and analogically performed, cryptography is now digitally performed by signals in networks.

The reasons for cryptography can be summarized as the need for five important items [8]:

- Privacy/confidentiality: Maintaining privacy or confidentiality means that no one other than who the message is intended for should have access to the message without the proper permissions.
- Authentication: Authentication is having a process of checking or verifying the user's identity before getting access.
- Integrity: To maintain the integrity of a message, we have to be sure that the message is not altered in any way while going from the sender to the receiver.
- Non-repudiation: This refers to having a way of making sure that the sender really sent the message.
- Key exchange: This refers to developing a protocol by which crypto keys are shared between the sender and receiver.

The following subsections will introduce the two main categories of encryption based on key exchange, Symmetric and Asymmetric key encryption.

Symmetric and Asymmetric key encryption

The concept of symmetric encryption algorithms, for example AES [5], is probably the easier of the two to understand. Basically, a shared cryptographic key is used to both encrypt and decrypt the desired information. This means that, in order to not put information at risk, it is crucial for users to keep the shared key safe and secure. One of the weak points of this approach is with respect to transportation and distribution. If the transportation is not secure and the key is intercepted, fraudulent actors might take over the key with potentially catastrophic consequences, depending on the information. Asymmetric encryption algorithms, however, use two different keys known as the Public key and the Private key. This methodology can be used in a variety of different applications, from securing a communication to digital signature. Depending on the scenario, one of the two keys involved is used to encrypt the data while the other is used to decrypt it. While the two keys are mathematically linked to one another and

URL: http://dx.doi.org/10.14738/tmlai.102.11939

the Public key is distributed to the public, the private key cannot be inferred from the public key and is kept secret by the single entity involved in the communication channel. This kind of encryption also involves the presence of a third-party entity, the *Certification Authority*. This entity has the purpose of issuing certificates, linking a Public key to its owner and certifying the validity of this association to whoever requests it.

Let's suppose that Alice wants to send an encrypted message to Bob. Alice will encrypt the message using Bob's Public key, so that hopefully nobody but Bob will be able to decrypt the message using the Private key, even if the communication channel is not guaranteed to be secure [see Figure 2a].

Another application for asymmetric keys is a digital signature, where the owner "signs" a document with the Private key, and everybody else can verify the signature by decrypting it using the Public key. This process is also shown in Figure 2b. This methodology is widely used where detecting forgery is especially important, for example in the financial sector or contract management software.

Generally, symmetric key encryption also corresponds to many advantages with respect to the asymmetric counterpart in terms of efficiency. And, in order to overcome carrier channel security vulnerability by the design of encryption methodology, symmetric encryption is often coupled with asymmetric encryption algorithms for the exchange of keys. This way, Alice and Bob can securely exchange the symmetric key before starting their communication. Bob, who starts the communication, would encrypt the symmetric key with Alice's public key, so that when the encrypted package arrives, she can decrypt it with her private key, and finally start the symmetrically encrypted communication without the virtual risk of disclosing the key to fraudulent third-party entities.

Block Cipher and Stream Cipher

There are two separate methods to perform symmetric encryption, *block ciphers* and s*tream ciphers*. As explained in [9], during the first, the plain messages are divided into fixed-size blocks, usually 64 and 128 bits, and converted into ciphertext. During the second, the information is encrypted bit-by-bit (or byte-by-byte), converting each individual quantity to a ciphertext by performing XOR operations between pseudo-random generated bits from the key and the information bits. In most parts of the modern symmetric algorithm, like AES [8], the current standard for symmetric encryption, are block ciphers. Despite their speed of transformation, stream ciphers are prone to insertions and modifications due to malicious interceptors, who can break the algorithm and insert spurious text in the stream while still appearing as an authentic transmission. As shown by [10], for cryptosystems which use the same key repeatedly, block ciphers are usually considered stronger than stream ciphers.

AES - Advanced Encryption System

The AES algorithm was developed in 1998 by Joan Daemen and Vincent Rijment [5]. AES was born as a result of a need for NIST [7], as an alternative for Data Encryption Systems (DES), which became vulnerable to attacks causing increased computational performance. There are three different versions of the algorithm for different key lengths, AES-128, AES-192 and AES-256 [5]. The algorithm encrypts and decrypts data in blocks of 128 bits using a cryptographic key with lengths 128, 192 or 256, depending on the algorithm version. The data is then divided

in 4 basic operational blocks, treated as an array of bytes and forming a 4x4 matrix, known as the state [5]. During the multiple rounds, cipher transformations are performed, such as Bytes substitution, Rows shifting, Columns mixing and Round key addition. This algorithm is briefly shown in Figure 1.

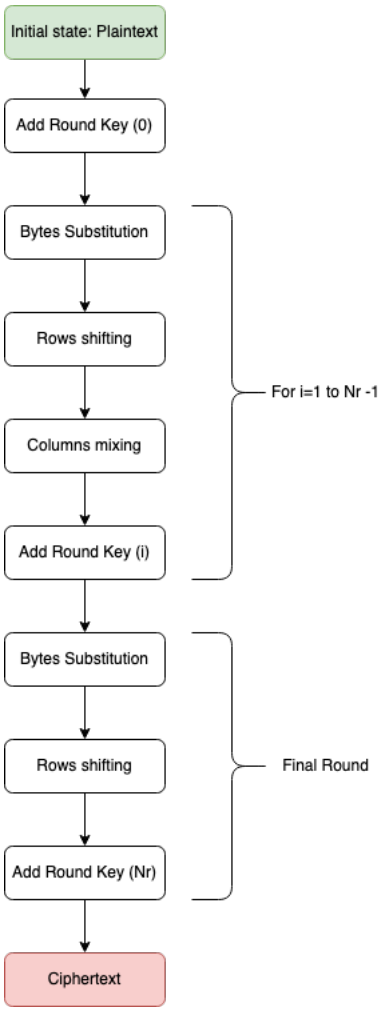


Figure 1: AES Algorithm

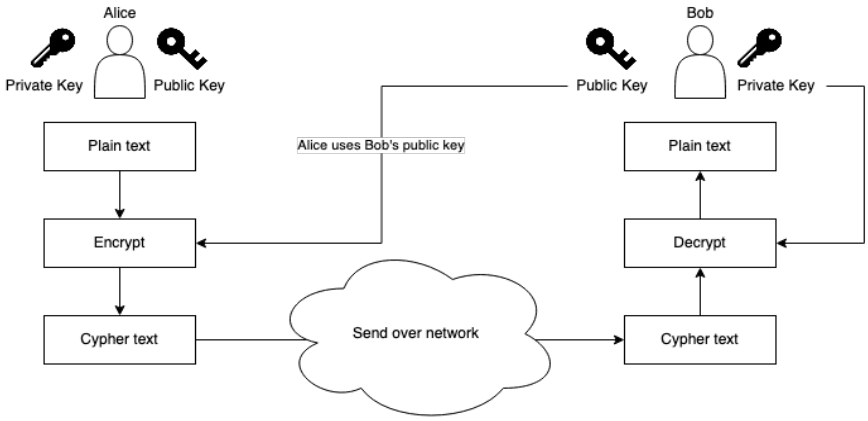


Figure 2a: RSA - A message encryption using asymmetric encryption

URL: http://dx.doi.org/10.14738/tmlai.102.11939

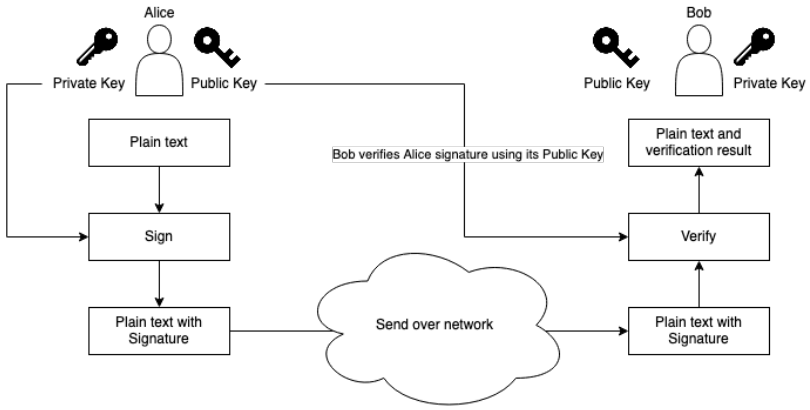


Figure 2b. A digital signature verification process between Alice and Bob

RSA - Rivest-Shamir-Adleman

RSA is a cryptosystem named after its founders Rivest, Shamir and Adleman[22], and it is one of the most known Public key algorithms for encryption and digital signature. The main concept is as explained in section for the asymmetric encryption algorithms, and it uses number theory. It uses two prime numbers to generate the public and private keys, sized 1024 to 4096 bits. The process of generating these prime numbers can be summarized with the following pseudocode algorithm:

Algorithm 1: RSA Algorithm - Key generation

The public key, which allows us to encrypt the message, is given by the exponent e and the modulus n. The decryption key is given by d.

#two large prime numbers are selected p, q = randomized-primes-selection():
# p and q are multiplied, creating n, used as the modulus for both the private and public keys n = p×q
# the Carmichael's totient (l.c.m) function is computed for n, defined as the least common multiple of two numbers, and is obtained by computing first the greatest common divisor (g.c.d) between two numbers
a = p - 1 b = q - 1
$\lambda(n) = l.c.m(a,b) = \frac{a \times b}{g.c.d(a,b)}$
A value e is chosen such to be co-prime with $\lambda(n)$
Find d, such that
d × *mod* $\lambda(n)$ = 1

Once *n, p, q* and *d* are obtained, the encryption and decryption process follows simple number theory operations, as demonstrated in Algorithm 2.

Algorithm 2: RSA Algorithm - Sender

# The sender computes typically a 256-unsigned integer representation of the message
message = get-message() # obtain public key tuple (n, e) n, e = get-public-key-from-ca ()
# compute ciphertext using the public key $c = (message^e) mod\ n$
send message(c)

Algorithm 3: RSA Algorithm - Receiver

The receiver simply revers the messages by using its private key d $message = (c^d) mod\ n$

The security of this process relies on a simple assumption that the factoring problem of large prime numbers is difficult. In fact, if a fraudulent actor is able to factor the modulus *n* into its prime factors *p* and *q*, they could use the public key *e* to derive the private key *d*. As of today, a "classical computer" would take a very long time to break such an algorithm. Instead, it is considered easier when quantum computing comes into play.

Quantum computing, which is the science of exploiting properties of quantum states, like superposition and entanglement to perform computation, has made incredible progress during the last three decades. Quantum algorithms are able to exploit these capabilities to resolve hard problems in polynomial time. As presented in [11], they significantly reduce the cost of factoring integer numbers by taking inspiration from different methodologies like the Shor's Algorithm [12], showing that it is possible to break a 2048-bit RSA key in less than 10 hours with the use of Quantum Computing. This raised the necessity of Quantum-safe asymmetric cryptosystems. The most known methodology is Quantum Key Distribution, or QKD, where, as explained in [13], by using properties of photons, it is possible to transmit information and detect if this information has been sniffed or copied in any point of the transmission.

Authenticated Encryption with Associated Data and Hashing

Authenticated Encryption with Associated Data (AEAD) techniques, as addressed by Rogaway in [14], is defined as the cryptographic problem of flowing a message which is in part privacy-protected, in part clear and all of it authenticated. Before defining AEAD, authenticated encryption methodologies were known already, but were lacking the possibility of binding an associated-data to ciphertext [14]. The implementation of AEAD techniques allows one to achieve confidentiality of a payload as well as integrity and authenticity of the information in a message. On a high-level of abstraction, the associated data gets intrinsically linked to ciphertext, creating a contextual situation.

Let us suppose that we have a database table with key-value tuples, where the value is an encrypted field. We can use the key field as associated data. Every time we decrypt the value we must present its key. This prevents malicious switches of values between the rows of the table. Another possible usage of AEAD is to secure the communication between two parties. Starting from Transport Layer Security (TLS) 1.3, defined in RFC 8446 [15] in August 2018, all cipher suites, hence the combination of algorithms that provide key exchange, authentication, encryption to Message Authentication Code (MAC), adopt AEAD techniques, preventing padding oracle vulnerabilities found in the previous versions. Hashing algorithms, instead, are able to compress data to a fixed size while benefiting from the properties of *irreversibility* and *uniqueness*. One of the ways to exploit these properties is the verification of data authenticity. For example, in digital signatures, data is often attached with a hashed version and encrypted with the sender's private key. Once the data is received, the recipient decrypts the document and calculates a new hash for it. If this matches the decrypted version of the attached hash, the recipient can infer that any modification has been performed since the signature. Hash and AEAD can be used together, for example, a password can be used both to generate an encryption key using hashing and associated data to a ciphertext.

Key Management Services

We have seen how cryptographic algorithms are intrinsically bound to cryptographic keys and how such keys must be safe and secure in order to keep data undisclosed to third-party entities.

URL: http://dx.doi.org/10.14738/tmlai.102.11939

The problem about where we should store the keys naturally arises. Questions like: "*Where should we store this key? Is it safe?"* have been heard, at least once, by every software developer which develops secure applications. Once the number of services and applications rises, the need for scalable solutions becomes indispensable in order to keep everything under control. The usage of Key Management Services (KMS) is a possible solution, which allows for the strengthening and securing of the encryption keys, while serving a secure infrastructure adding an additional level of abstraction to clients and developers. KMS are tools used to centralize the management of encryption keys. KMS provides different services like *key generation, key versioning* and *rotation,* and complex multiple layers of keys for encrypting data and encryption the key itself using mechanisms like *envelope encryption.* A comparative study of different Key Management Services is presented by [16], analyzing 58 software based key management tools, of which 32 are open source projects and 26 are closed source.

Envelope Encryption

Encrypting a key with another key is referred to as envelope encryption and is one of the services provided by most of KMSs. In this case we need two different keys. The *Data Encryption Key (DEK)* is used to encrypt the original or actual data and the *Key Encryption Key (KEK)* is used for wrapping the DEK. This provides extra protection to the KMS since the encryption version (KEK) is stored in the database. And of course, a different data key is generated for each record stored. When using envelope encryption, to decrypt the data, we have to first retrieve the wrapped DEK, use this key in the KMS to unwrap the encrypted DEK and the plain DEK to decrypt the encrypted data.

LIGHTWEIGHT CRYPTOGRAPHY AND ZERO KNOWLEDGE PROOF

Present day resource-constrained IoT devices often form complex interconnected networks, with a great amount of data flowing between nodes. But, although still constrained, many IoT devices today are able to bring to the edge, that is, on the network, many operations that once required third services on the cloud, consequently reducing latency and decentralizing the computation.

As the number of IoT devices increase, along with the burden of their responsibilities, the need of securing such environments is impelling researchers to find cryptographic algorithms and methodologies more suitable to secure resource-constrained devices. The following section presents a summary of the ongoing research led by NIST and the possibility of using *zero-knowledge proof systems* in the context of resource-constrained devices.

Lightweight Cryptography for resource-constrained devices

Techniques like AEAD and hashing can be applied in order to preserve the sensitivity of our communications, and while many frameworks and libraries allow us to easily take advantage of such security measures, this becomes less trivial to achieve for resource-constrained devices like the ones usually involved in IoT environments. In 2018, NIST [7] published submission requirements and evaluation criteria for the *LWC (Lightweight Cryptography)*, starting a standardization process that points to such issues. The call by NIST sought to determine AEAD and hashing schemes suitable for more resource constrained environments. The selection started in February 2019, with 57 candidates and in March 2021, ten finalists were selected for the final round of selection process, with a report published in the NISTIR 8369 technical report [7]. During the latest round of the technical report, NIST developers provided optimized

implementations of the candidates as well as software benchmarks. The scope of this standardization process was to provide protection schemes for such environments, as well as radio frequency identification tags and sensor networks. In March 2021, NIST announced 10 finalists of the standardization process. The criterion considered for the advancement to the final phase were:

- First criterion: Third-party analysis or the presentation of well-understood design principles and security proof that strengthens their security claims;
- Second criterion: Performance of the candidate solution in applications using constrained devices and that outperformed current NIST standards such as AES-GMC and SHA-256;
- Other criterions: Side-channel resistance, nonce misuse security, releasing unverified plaintext security, impact on state recovery and post-quantum security;

Among the ten finalist proposals, three presented an AEAD and Hashing solution and seven provided AEAD-only functionalities. All the AEAD with Hashing finalists used permutations while the ones providing AEAD-only functionality used either Permutation, Block Cipher, Tweakable Block Cipher or Stream Cipher. The ten finalists were:

- ASCON(AEAD + Hashing): Permutation
- Elephant(AEAD): Permutation
- GIFT-COFB(AEAD): Block cipher
- Grain-128AEAD(AEAD): Stream cipher
- ISAP(AEAD): Permutation
- PHOTON-Beetle(AEAD + Hashing): Permutation
- Romulus(AEAD): Tweakable Block Cipher
- SPARKLE(AEAD + Hashing): Permutation
- TinyJAMBU(AEAD): Block cipher
- Xoodyak(AEAD + Hashing): Permutation

The final selection of the proposal was based on different benchmarking initiatives, including software and hardware benchmarks. As for software benchmarks, an internal evaluation by NIST was performed, testing 327 AEAD and 116 hash implementations on two 8-bit microcontrollers and four 32-bit microcontrollers. Two additional benchmarks on microcontrollers were also performed, the first using frameworks developed by Renner et al [17], published in [18], while the second was performed with the usage of optimized implementations for timing performance measurements developed by Weatherley [19]. Additional results of different nature regarding software benchmarks are presented in [20] [21] [22] [23].

Along with the performance evaluations and the in-depth analysis of security claims, other factors also played a significant role in the selection. NIST also considered tweaking plans with possible structural changes proposed by the candidates. For example, the diversity of the candidates, where candidates based on the same primitive were compared to each other, and how the candidates would fit within its existing portfolio of cryptographic standards were considered. NIST estimates that the third round of evaluation and review will take about a year since the announcement of the finalists.

URL: http://dx.doi.org/10.14738/tmlai.102.11939

Interactive Zero-Knowledge Proofs

In 1989, [24] introduced the rigorously formalized notion of *zero-knowledge proof system.* A theorem-proving procedure, based on the concept of an *interactive* proof, where the "recipient" of the proof and the "prover" respectively, ask questions and formulate answers, in order to verify the correctness of the proof. This communication process allows a system to prove their knowledge without disclosing the actual information and preventing the recipient from knowing the actual information. The logic behind this methodology leverages probabilistic assessments. This means that the recipient of the proof is not going to have complete certainty about the knowledge of the prover, but instead, thanks to repeated interactions, the prover is going to provide un-linkable information and answers to the recipient questions, so that the latter can rely on an overwhelming probability that the prover effectively asserts the truth about its knowledge.

Let us imagine a real-world situation where we need to implement a login for some kind of web server. In the classic scenario, the user creates its account and chooses the password needed to log into the service. Then, through a more or less complex process, the password is hashed on the server.

The evident problem in this process is that the server has this hashed version of the password and leverages on the fact that the server itself cannot be compromised, with the user "trusting" the login process to be clean and protected. The benefits of a Zero-Knowledge approach are immediate, for example, we would be able to perform authentication without the need for disclosing any secrets in the process.

A classic example, presented by Quisquater et al. in [25], explains in simple words the idea behind how zero knowledge works. The example is also illustrated in Figure 3 and is known as the *cave of Ali Baba*. Let Alice know a password and wants to prove to Bob her knowledge without disclosing the password itself. The password, in this example, is the secret word to open a magic door placed in a circular cave. Alice wants to show Bob that they know the secret word, but without revealing it:

- Alice goes inside the cave, and can choose to go whichever direction she wants, left or right;
- Alice chooses left, and Bob finally enters the cave. Bob does not know if Alice choose left or right;
- Bob asks Alice to come outside, and to appear coming from its right (remember that Alice went left before);
- Because Alice knows the secret word to open the door, she opened the door and walked all the cave appearing on Bob's right;
- *What if Alice didn't know the secret and was just lucky?* Choosing to take the right at the beginning instead of the left. The problem is that Alice might have luck the first time, choosing the path that Alice asked and so it didn't need to open the door. At this point it's very easy to fool Bob, because there's a 50% chance that Alice knows the secret. In order to reduce the chance of Bob being fooled by Alice, they repeat the procedure several times. Each time they repeat this process the chances of Bob cheating reduces. For example, if they do it 20 times in a row, Alice's chance of successfully anticipating all of Bob's requests would become vanishingly small - about one in a million.

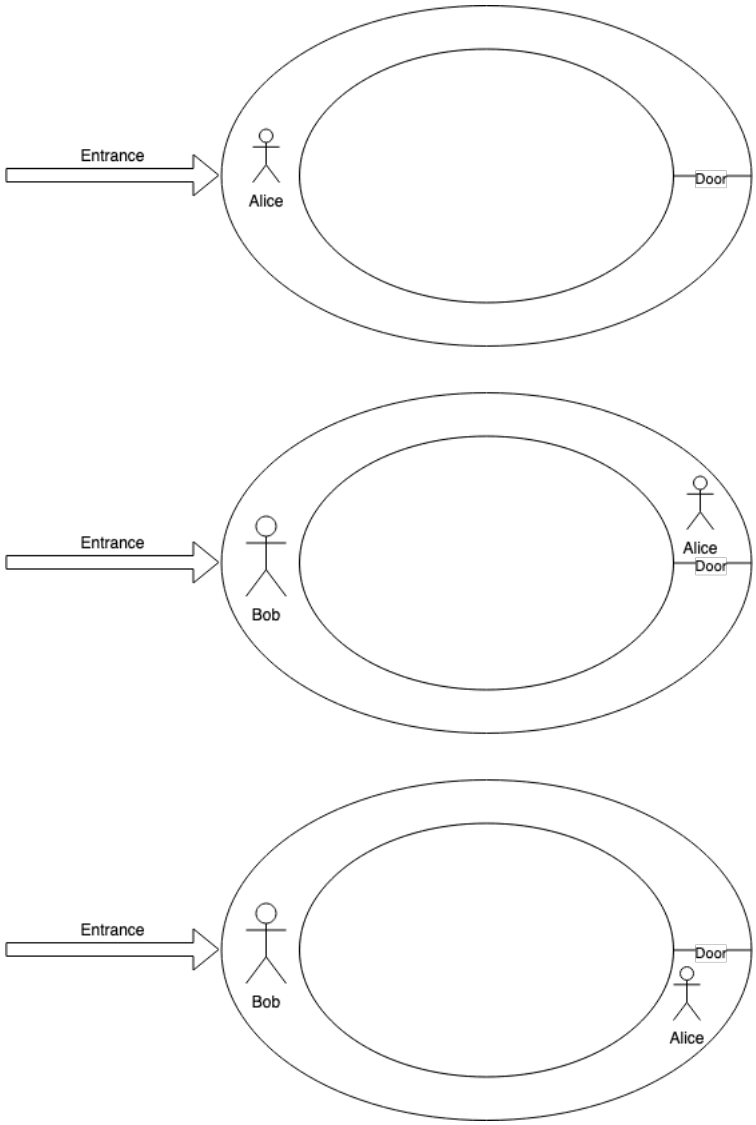


Figure 3: The cave Of Ali Baba - Interactive Zero-Knowledge proof example

Non-Interactive Zero-Knowledge Proofs

The previous section introduced Interactive Zero-knowledge proofs. This, as the name suggests, requires the prover and the verifier to interact. This methodology might be suitable for certain environments, but not for systems that require a single interaction and verification within milliseconds. In non-interactive Zero-Knowledge proofs, the role of the verifier is substituted by a hash function.

Zero Knowledge proofs for IoT networks

All the tasks that we have just listed imply the handling of sensitive data flowing through the network, from sensor data that can also include healthcare applications [26] to audio signals and video frames. And the more devices deployed on the network, the more "surface of attack" malicious actors have. That is why solid authentication methodologies and encryption algorithms must be adopted and optimized. Research has been focusing on this thematic and many of its aspects, like the detection of malicious attacks in Wireless Sensor Networks [27] and the possible vulnerabilities [28]. Recently, many authentication schemes based on zero

URL: http://dx.doi.org/10.14738/tmlai.102.11939

knowledge proofs have been proposed, while keeping in mind the resource constraints brought by such low-powered devices. [29] proposes TinyZKP, a Zero Knowledge Proof approach which reduces energy cost and time needed for an efficient authentication, while [30] uses a Non-interactive zero-knowledge approach, indicating a better suitability rather than more traditional interactive approaches.

CONCLUSION

We have reviewed some of the most important topics on cryptography, looking at both what are the historical standards and terminologies, as well as what are the future trends and challenges of this field. Many other arguments that are crucial in the field might be of interest for the user, like Envelope encryption, the various challenges of the radio spectrum when it comes to encryption and security or how Deep Learning techniques are being used to understand encrypted communications. It is clear that both the academic and industrial research worlds are pointing to developing new secure protocols, challenging the advent of Quantum general-purpose computing and strengthening networks involving IoT devices, with particular regards to authentication and computationally-efficient proofs of knowledge systems.

References

1. Lionel Sujay Vailshery. Number of IoT devices 2015-2025. Nov. 2016. url: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.
2. DDoS attack that disrupted internet was largest of its kind in history. Oct. 2016. url: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.
3. George Dascalu. New Botnet Dubbed Mirai Compromised Over 300,000 IoT Devices. July 2021. url: https://news.softpedia.com/news/new-botnet-dubbed-mirai-compromised-over-300-000-iot-devices-533441.shtml.
4. The FDA confirmed that St. Jude Medical's implantable cardiac devices have vulnerabilities that could allow a hacker to access a device. Once in. FDA confirms that St. Jude's cardiac devices can be hacked. url: https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/.
5. Dworkin, M. , Barker, E. , Nechvatal, J. , Foti, J. , Bassham, L. , Roback, E. and Dray, J. (2001), Advanced Encryption Standard (AES), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.FIPS.197 (Accessed December 18, 2021)
6. R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: Commun. ACM 21.2 (Feb. 1978), pp. 120–126. issn: 0001-0782. doi: 10.1145/359340.359342. url: https://doi.org/10.1145/359340.359342.
7. National Institute of Standards and Technology. Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process. Tech. rep. Computer Security Resource Center, July 2021. doi: https://doi.org/10.6028/NIST.IR.8369.
8. Gary Kessler. An Overview of Cryptography. 2021.
9. Jean-Philippe Aumasson. Serious Cryptography: A Practical Introduction to Modern Encryption. USA: No Starch Press, 2017. isbn: 1593278268.
10. Gustavus J. Simmons. "Symmetric and Asymmetric Encryption". In: ACM Comput. Surv. 11.4 (Dec. 1979), pp. 305–330. issn: 0360-0300. doi: 10.1145/356789.356793. url: https://doi.org/10.1145/356789.356793.
11. Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: Quantum 5 (Apr. 2021), p. 433. issn: 2521-327X. doi: 10.22331/q- 2021- 04- 15- 433. url: https://doi.org/10.22331/q-2021-04-15-433.

12. Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: SIAM Journal on Computing 26.5 (Oct. 1997), pp. 1484–1509. issn: 1095-7111. doi: 10.1137/s0097539795293172. url: http://dx.doi.org/10.1137/S0097539795293172.

13. Charles H. Bennett and Gilles Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: Theoretical Computer Science 560 (Dec. 2014), pp. 7–11. issn: 0304-3975. doi: 10.1016/j.tcs.2014.05.025. url: http://dx.doi.org/10.1016/j.tcs.2014.05.025

14. Phillip Rogaway. “Authenticated-Encryption with Associated-Data”. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. CCS ’02. Washington, DC, USA: Association for Computing Machinery, 2002, pp. 98–107. isbn: 1581136129. doi: 10.1145/586110.586125. url: https://doi.org/10.1145/586110.586125.

15. Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. Aug. 2018. doi: 10.17487/RFC8446. url: https://rfc-editor.org/rfc/rfc8446.txt.

16. Kuzminykh, I., Ghita, B., & Shiaeles, S. (2020). Comparative Analysis of Cryptographic Key Management Systems. Internet of Things, Smart Spaces, and Next Generation Networks and Systems, 80–94. doi:10.1007/978-3-030-65729-1_8

17. S. Renner, E. Pozzobon, and J. Mottok. Benchmarks for Lightweight Cryptography. url: https://lab.las3.de/gitlab/lwc/compare.

18. Renner S, Pozzobon E, Mottok J (2020) Current and Future Efforts in Benchmarking NIST LWC Ciphers, NIST Cryptography Workshop 2020. https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/current-and-future-efforts-in-benchmarking-lwc2020.pdf.

19. R. Weatherley. rweather/lightweight-crypto: NIST Lightweight Cryptography Primitives. url: https://github.com/rweather/lightweight-crypto.

20. Fabio Campos et al. Assembly or Optimized C for Lightweight Cryptography on RISC-V? Cryptology ePrint Archive, Report 2020/836. https://ia.cr/2020/836. 2020.

21. D. Bernstein and T. Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. url: https://bench.cr.yp.to/.

22. Goerken Nisanci et al. WILL THE FUTURE LIGHTWEIGHT STANDARD BE RISC-V FRIENDLY? url: https://csrc.nist.gov/CSRC/media/Presentations/will-the-future-lightweight-standard-be-risc-v-fri/images-media/session4-yalcin-will-future-lw-standard-be-risc-v-friendly.pdf.

23. Luan Cardoso dos Santos and Johann Großschäd. An Evaluation of the Multi-Platform Eÿciency of Lightweight Cryptographic Permutations. url: https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop- 2020/documents/papers/performance- evaluation- cryptographic- permutations- lwc2020.pdf.

24. S. Goldwasser, S. Micali, and C. Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: SIAM J. Comput. 18 (1989), pp. 186–208.

25. Quisquater JJ. et al. (1990) How to Explain Zero-Knowledge Protocols to Your Children. In: Brassard G. (eds) Advances in Cryptology — CRYPTO’ 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol 435. Springer, New York, NY. https://doi.org/10.1007/0-387-34805-0_60

26. Moshaddique Al Ameen, Jingwei Liu, and Kyungsup Kwak. “Security and privacy issues in wireless sensor networks for healthcare applications”. In: Journal of medical systems 36.1 (2012), pp. 93–101.

27. S.G. Hymlin Rose and T. Jayasree. “Detection of jamming attack using timestamp for WSN”. In: Ad Hoc Networks 91 (2019), p. 101874. issn: 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2019.101874. url: https://www.sciencedirect.com/science/article/pii/S157087051830667X.

28. Yuan Gao et al. “Mobile Network Security and Privacy in WSN”. In: Procedia Computer Science 129 (2018). 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION,INFORMATION AND KNOWLEDGE IN THE INTERNET OF THINGS, pp. 324–330. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.03.083. url: https://www.sciencedirect.com/science/article/pii/S187705091830320X.

29. Limin Ma, Yu Ge, and Yuesheng Zhu. “TinyZKP: A Lightweight Authentication Scheme Based on Zero-Knowledge Proof for Wireless Body Area Networks”. In: Wirel. Pers. Commun. 77.2 (July 2014), pp. 1077–1090. issn: 0929-6212.

30. Marcus Walshe et al. “Non-interactive zero knowledge proofs for the authentication of IoT devices in reduced connectivity environments”. In: Ad Hoc Networks 95 (2019), p. 101988. issn: 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2019.101988. url: https://www.sciencedirect.com/science/article/pii/S1570870519304895.8