PROLISEAN: A New Security Protocol for Programmable Matter

Edy Hourany, Bachir Habib, Camille Fountaine, Abdallah Makhoul, Benoit Piranda, Julien Bourgeois

PROLISEAN: A New Security Protocol for Programmable Matter

EDY HOURANY and BACHIR HABIB, Holy Spirit University of Kaslik
CAMILLE FOUNTAINE, École Spéciale Militaire de Saint-Cyr Coëtquidan
ABDALLAH MAKHOUL, BENOIT PIRANDA, and JULIEN BOURGEOIS, Univ. Bourgogne Franche-Comté, FEMTO-ST Institute, CNRS

The vision for programmable matter is to create a material which can be reprogrammed to have different shapes and to change its physical properties on demand. They are autonomous systems composed of a huge number of independent connected elements called particles. The connections to one another form the overall shape of the system. These particles are capable of interacting with each other, and take decisions based on their environment. Beyond sensing, processing and communication capabilities, programmable matter includes actuation and motion capabilities. It could be deployed in different domains and will constitute an intelligent component of the IoT. A lot of applications can derive from this technology, such as medical or industrial applications. However, just like any other technology, security is a huge concern. Given its distributed architecture and its processing limitations, programmable matter cannot handle the traditional security protocols and encryption algorithms. This paper proposes a new security protocol optimized and dedicated for IoT programmable matter. This protocol is based on light weight cryptography and uses the same encryption protocol as a hashing function while keeping the distributed architecture in mind. The analysis and simulation results show the efficiency of the proposed method and that a supercomputer will need about 5.93×10^{25} years to decrypt the message.

CCS Concepts: • **Security and privacy**; • **Computer systems organization** → **Distributed architectures**;

Additional Key Words and Phrases: Modular Robots, Programmable Matter, IOT, Amoebots, Security Algorithms, Security Protocol, Lightweight Cryptography, Distributed Computing

ACM Reference Format:
Edy Hourany, Bachir Habib, Camille Fountaine, Abdallah Makhoul, Benoit Piranda, and Julien Bourgeois. 2020. PROLISEAN: A New Security Protocol for Programmable Matter. 1, 1 (January 2020), 29 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recent technological advances resulted in an increasing level of interest in programmable matter. Programmable matter is a vast domain of research at the border between robotics and computer sciences, mixing large ensembles of micro-robots and distributed programming. The idea is to find a way to make a matter that can change its physical properties at will, whether its shape, density or conductivity [2]. The potential of such a technology is tremendous. The most common idea is to use a countless number of robots working together within a structure called modular robot. A typical modular robot is expected to consist of a large number of robots called modules or particles which can be

Authors' addresses: Edy Hourany, edyhourany@gmail.com; Bachir Habib, bachirhabib@usek.edu.lb, Holy Spirit University of Kaslik, P.O. Box 446, Kaslik, Jounieh - Lebanon; Camille Fountaine, cam.fontaine49@gmail.com, École Spéciale Militaire de Saint-Cyr Coëtquidan, P.O. Box 56380, Guer, France; Abdallah Makhoul, abdallah.makhoul@univ-fcomte.fr; Benoit Piranda, benoit.piranda@univ-fcomte.fr; Julien Bourgeois, julien.bourgeois@univ-fcomte.fr, Univ. Bourgogne Franche-Comté, FEMTO-ST Institute, CNRS, 1 cours Leprince-Ringuet, Montbéliard, France.

reprogrammed to have different shapes and to change its physical properties on demand. Usually, these particles have limited power, storage, communication, and processing capabilities. Modular robots have many fields of application, including Space shuttles, medical appliances, and search and rescue, etc. [1]. They play an important role in saving human lives by dismantling into smaller configuration and search for survivals; or changing shapes into forming bridges or stairs for people to reach safer areas. In the near future, this new concept will play an indispensable role in everyday life and will encompass almost all areas. Mobile telephony, smart homes, vehicles, medicine, and embedded systems are the top concerns of researchers. These modules use different types of sensors and actuators to extract and analyze information. Billions of objects will be connected. Through this connection, this material will become an intelligent system composed of other systems.

Modular robots can be used in unattended even hostile environments, thus leaving them vulnerable to passive and active attacks. For instance, crashing a space shuttle if a module was hacked and start sending falsified coordinates, or a nano robot that is injected in the human body and starts killing healthy cells. Therefore, the security in such scenarios is a necessity especially in the context of modular robots where modules have resource-constraint, and are susceptible to physical capture. To the best of our knowledge, security in modular robots is not yet tackled by researchers. Many challenges are encountered when discussing such matter. This kind of robots have no central control and have limited memory and processing resources. Under such circumstances, traditional security protocols cannot be applied. Therefore, lightweight security protocols must be proposed and studied. Such protocols were proposed in the context of wireless sensor networks [5, 15, 29], ad-hoc networks [23] and Internet of Things (IoT) [30, 33]. Although, modular robots are considered to be an application of IoT, they have some specific characteristics. For instance, they can not have a unique identifier or IP addresses. Moreover, they are not always connected to the internet and don't have a central control. Therefore, existing security protocols for IoT can not be applied for programmable matter.

The main contribution of this paper is to study and propose the first security protocol that can overcome the challenges mentioned above while keeping the system secure. It is based on the creation of a hashing function using the encryption algorithm and a block code already embedded in the nanobot to reduce the memory and processing usage. This protocol will be composed of two main parts: authentication and encryption. In the authentication part, the algorithm will be checking whether the connected module is authentic. In other words, check if this module is an eligible module and that it has the right to connect to the system. Once all modules are authentic, the second step will be encrypting the communication among the modules, thus making it hard for a third party that might be listening to the communication to understand the behaviors and the decisions of the system. The efficiency of the proposed protocol was proven via an analysis study and validated experimentally via simulations realized by VisibleSim[25], a simulator dedicated to modular robots. The obtained simulation results show the effectiveness of the proposed approach which ensures the security of a modular robot system.

The remainder of this paper is organized as follows. Section 2 of this paper describes the background behind this study by reviewing modular robotics and light weight cryptography. Section 3 is the main section of this paper. It will be discussing the proposed protocol with four different versions. And a full risk study through EBIOS method for all the versions proposed. Section 4 will be analysing the proposed protocol based on the CIA criteria while taking into consideration the hardware and logical limitations. Section 5 will be analyzing the HIGHT encryption algorithm by trying nine different attacks. And finally Section 6 is dedicated to the conclusion and future work.

2 BACKGROUND DESCRIPTION

2.1 Modular robotics and programmable matter

The term of programmable matter appeared a quarter century ago and was defined as a matter that is able to modify its physical properties like shape, color or density autonomously or controlled by a user. The most common solution known is to use a huge number of small robots comparable to the cells that compose our body. This comparison gave its name to this solution: Amoebots [13, 21, 32]. Modular robotics are robots designed with parts that can be reconfigured to accept different functions[6]. They are autonomous systems with variable morphology. These systems are composed of independent connected elements called modules or particles, whose connections to one another form the overall shape of the system [2]. Classifying modular robotic systems is quite a hard task because of the number of projects already existing and the diversity of their task, shape, size or control system. Some modules are controlled by external computers. Others are led by a control module or they are completely autonomous. Nevertheless, all modules are equally equipped physically and logically. Depending on the purpose of its creator, modular robotic systems can assemble its own structure, operate some changes in order to repair defective parts or to modify the shape of the structure. This is the main advantage of this technology, while classical robots are only able to fit with one application or objective, modular robots could possibly operate a wide range of missions [1]. Despite these advantages, this development suffers from a major security problem. To the best of our knowledge, no security protocols were used in programmable matter. Given the limitation in modular robotics regarding the bandwidth, the multiple modules, its distributed architecture, its memory limitation, and the lack of physical protection, the traditional key management approach is not reliable in the way of its distribution between the objects.

To reach such objective, a lot of challenges are faced. Such as the connections between the particles, the data transfer among the modules, the communications inside the structure or the ability to localize its own position in the space. Today, manufacturing amoebots is impossible but the technologies of micro-fabrication and cellar engineering have made such advances that the probability of discovering how to build it rises every day. That is why research centers over the world are looking for theoretical models to face the maximum of issues concerning programmable matter [12]. This work is based on the general definition of amoebot model provided in the second part of the document [12] that state programmable matter as a uniform set of simple computational units. In the general amoebot model, all the particles are connected to at least another one and needs to connect with several particles to be able to move inside the structure. None of the particles is free to flow inside the structure.

In order to proceed in developing a new protocol, first the characteristics of the modules, in agreement with what is more likely to be built in the next decade in term of microscopic modular robots, are defined as follows:

Definition 1 (Anonymity). *Because of the size of the modules, also known as particles, and the quantity that makes the structure, each nanobot is anonymous.*

Definition 2 (Identical modules). *All the modules have the same design, the same hardware and the same algorithms.*

Definition 3 (Fixed memory). *The standard memory of 16KB, prevents the usage of heavy algorithms thus creating new algorithms to solve traditional problems.*

Definition 4 (Fixed connection of a module with its neighbors). *A Blinky Block module always have at least one neighbor in the structure and maximum of 4 neighbors in 2D and 6 in 3D.*

Definition 5 (Joint internal clock). *Every module sets its internal clock exactly at the same instruction. One of the main consequences concerns the creation of random numbers. In fact, as random numbers are created with an algorithm that uses the internal clock as a seed, all the modules will give the same random number at the same time. Since all modules are identical, even when changing the seed, the randomness collision will always exist.*

Definition 6 (Peer to peer communication by contact). *A Blinky Block can only communicate with its neighbors, it is not able to contact another module except the one connected to its port.*

2.2 Lightweight cryptography

In the wake of the massive arrival of small connected devices on the market with small capabilities but sensitive information to communicate, IoT became an easy target for cyber-attacks. Therefore, various algorithms and protocols have been developed over the past years to secure those communications. Regarding the low capacities of the devices compared to a laptop or a smartphone and the need of authentication, the use of cryptography appeared as a good solution. Since security algorithms -in this particular case – have to use less memory and have a low complexity to reduce their power consumption, a new kind of ciphers merged: the lightweight cryptography.

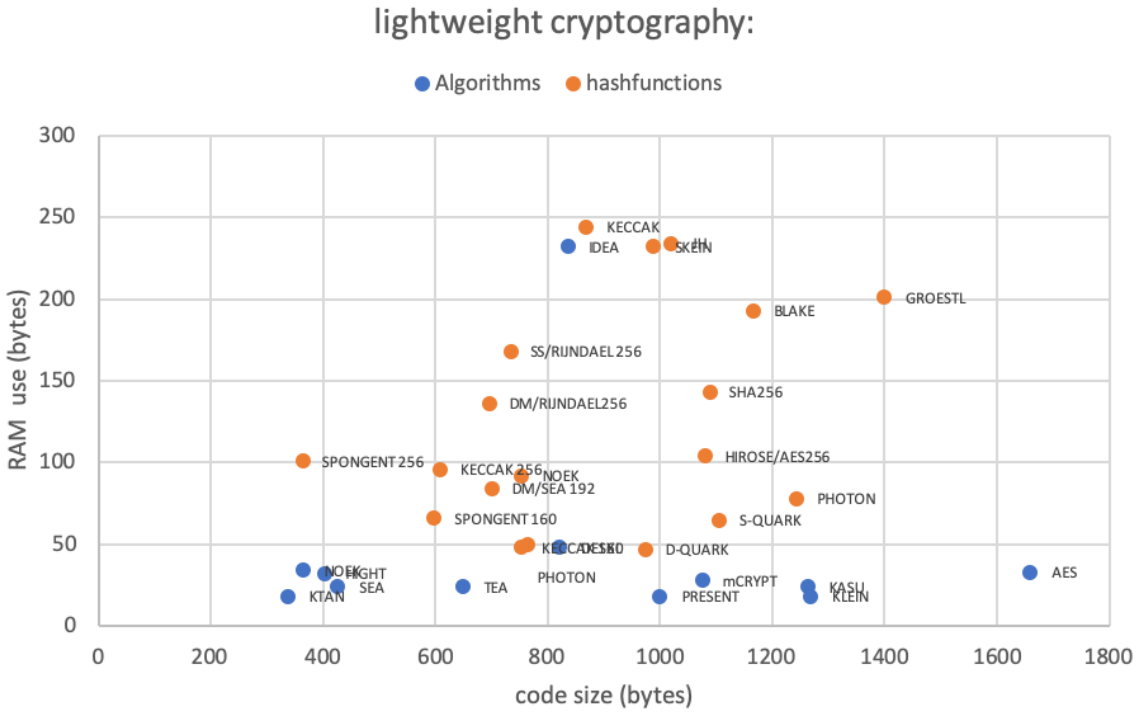


Fig. 1. Results of ECRYPT II ordered by code size and RAM use

Sharing the key is the major issue of this type since its acknowledgment by a malicious party compromises the system. Symmetric key algorithms allow confidentiality of data but can not guarantee authentication, this has to be done before the communication begins [7, 16, 19, 31]. Asymmetric cipher needs two keys: a private and a public one.

Each device has to know all the public keys of the network, this becomes a structural issue for nano-modular robots which theoretically includes billions of nano-modules. Therefore, this paper will be focusing on symmetric algorithms [8]. Figure 1 shows the results of different reports on lightweight cryptography algorithms and among all, the report of the European Network of Excellence in Cryptology [14] that bench-marked 11 different algorithms. This research focused on this type of encryption since all the algorithms were tested on a very low resources processor the ATtiny45 by the corporation ATMEL (Advanced Technology for Memory and Logic). It is an 8-bit processor made by ATMEL, with a limited number of instructions, a 256 Bytes internal SRAM, and 4 KB of flash memory. 19 hash-functions designed to be lightweight protocols are compared. Those are interesting in this study since there is no need for a key to use hashed data even if the decryption is too complicated for the nano-modular robots. The diagram in Figure 1, sorts all the applications by code size and RAM usage (in bytes) to restrain the list to the one that could fit in the nano-robots without impacting their memory capacity.

In this paper we proposed a new security protocol for programmable matter based on modular robots with low energy and computation resources. Similar studies were proposed for different technologies such as RFID, wireless sensor networks (WSN) and Internet of things (IoT). For instance, in [3], the authors describe a counter measure to prevent DoS/DDoS attack. The authors suggested the addition of an extra filter to determine a possible DoS attack and inform the routers to shut it down. In [11], the authors suggested first an installation of firewalls and updating software regularly from reliable resources. On another note, the paper recommended the usage of SSL certificate and efficient encryption algorithms. Using event detection techniques in wireless sensor networks, the authors in [18] developed an algorithm to detect malicious data injections. Malicious code injection attack is one where an attacker can determine the configuration of a system and create falsified measures allowing himself to bypass the security protocol.

Although the proposed techniques ensure good results in preventing several attacks, they can not be applied directly in the context of programmable matter. Hence, programmable matter have specific characteristics making them different from existing technologies. For instance, the communication between modules are done just between adjacent modules, the topology is very dynamic as the modular robot usually changes its shape, there is no base station to control the network, etc. Based on the study published in [20], we proposed in Table 1 a comparison between different technologies where lightweight cryptography protocols have been used (e.g. Radio-frequency identification (RFID), wireless sensor networsk (WSN), Internet of Things (IoT)) while adding the programmable matter (PM).

	Mobility Support	DB required	Power requirement	Popular Encryption Algorithms	Possible Security Techniques
RFID	Yes: modules are in form of tags and are attached to objects	Yes	Usually passive but sometimes Battery-powered	DES, AES, SecureRF, DESL	Optimistic trivial RFID authentication protocol (O-TRAP)
WSN	they are usually placed in fixed areas	Yes	Battery-powered	DES, 3DES, DES-X, blowFish, TEA, XTEA, AES, HIGHT	Route optimization algorithm, active trust,Q-s composite, TinySec, SPINS, LSec, LISA, LISP
IoT	Modules can be mobile or fixed	Yes	Battery-powered	ECC, Diffie-Hellman, COSE	REST, HTTP, JWT, CWT, WebSockets, TLS, DTLS
PM	Modules can be mobile or fixed	No	Battery-powered	HIGHT	PROLISEAN

Table 1. Comparison among different technologies.

3 PROLISEAN: OUR PROPOSED PROTOCOL

3.1 Risk study through EBIOS method

To begin with, it was necessary to identify potential threats to programmable matter based on modular robots. In this section, the elements will be summarized using the EBIOS method as shown in Figure 2. This method, created in 1995 and updated in 2010, is a memento for studying security risks for designing and engineering projects in information technology. The risk study of this method must be preceded by 3 steps: context study, study of threat scenarios and study of feared events.

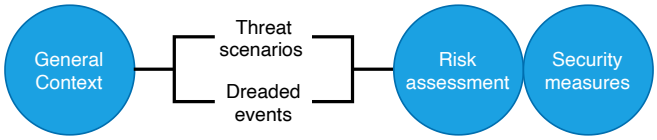


Fig. 2. EBIOS Process

3.1.1 Dreaded events. The main dreaded events can be summarized as follows:

- Loss of spatial landmarks: the spatial location by the entire structure is an important element in coordinating the individual actions of nanobots.
- Desynchronization of robots actions: although robots work individually, their action is useless if it is not coordinated with others.
- Data leak: internal communications within the structure are read by people or algorithms that should not be allowed to know the intentions of the structure.
- Distribution of messages from the outside: the modules receive instructions from a third party.

3.1.2 Threat scenarios. Robots communicate through physical connectors by contact with their direct neighbours. They can only send a message to one of their neighbours or to the entire structure through their neighbours. As described previously, the modules are homogeneous and autonomous.

The threat of remote interference in robot communications can therefore be ruled out. The main threats remaining are:

- The mixture of two different structures: causing misunderstandings between the modules because they would be linked with others with different behaviors, as shown in Figure 3. Or worse, if this second structure is controlled by a malicious person, it could transmit our internal information to its user, interfering with the execution of the mission.
- A spy in the structure: A hacker builds the same modules as those in our structure but equipped with an antenna and an interface, as shown in Figure 4. It could then connect, integrate into it and communicate with it to either take control or destabilize it. For example, it could claim to be the leader of the structure while another leader has been appointed.
- An intruder: An identical robot is sent to the structure by a malicious user, as shown in Figure 5. Its hardware is exactly the same, only few lines of its source code are different. Once connected, this intruder spreads malicious information to the whole structure to destabilize it. It could, for example, claim that it is the actual “leader” whereas another already exists or transmits wrong coordinates to its neighbours.

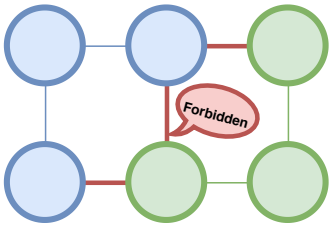


Fig. 3. Connection between two different structures

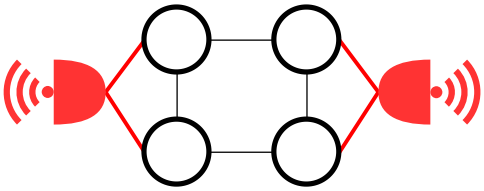


Fig. 4. External nanobots spying the structure

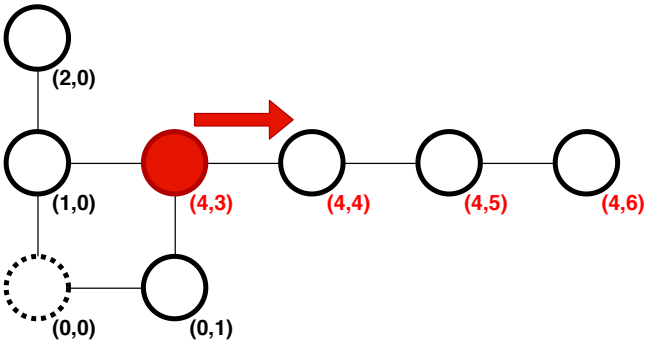


Fig. 5. An intruder transmits wrong coordinates to its neighbours

3.1.3 Risk assessment. Such a risk study is carried out in two parts: the analysis of the combined risk of scenarios and feared events with or without the security measures already in place. After considering the 3 elements of the threat scenarios, the risks are defined and classified by category and probability.

- Hacking the structure with no physical help: it has been demonstrated in previous sections that it is currently impossible to communicate with robots from a computer when the experiment is in progress. In this case, if a computer manages to get into a robot, it will not be able to take control of the entire structure. The probability is therefore negligible, and the risk is low. However, in another experimental context, where the structure would communicate directly with a computer, this risk would become one of the main concerns.
- Mixing two different structures of modules: these are two blocks of programmable matter moving nearby and composed of similar robots. It is then very likely that they will mix if they come into contact. The nanobots that compose them would stick together without understanding each other and the work of the two structures would be interrupted. The gravity of this case is not significant.

- Intentionally send a group of modules to insert into the structure: this risk is almost similar to the previous one, except that if the malicious structure has been programmed to come into contact with another, then its impact on its initial task will be more serious because it will be more incapacitating.
- Inserting a new module equipped with a communication interface: this risk, although unlikely because it is technically very complicated, would have critical consequences on the functioning of the programmable matter because it would allow another user to communicate with it and possibly divert it from its initial task or even destroy it.
- Inserting an identical module with wrong spatial coordinates in its code: this is more likely to happen than the previous one because it is easier to implement. However, the consequences would be a little less serious because the result would be at worst a deformed structure (topology) preventing the system to perform its task or an outgrowth on the final result.

Table 2 presents all considered risks, classified by category and probability.

		Risk assessment			
Category	Critical		Inserting a nanobot equipped with a communication interface		
	Significant		Intentionally send a group of nanobot to insert into the structure	Inserting an identical nanobot with wrong spatial coordinates in its code	
	Limited				Mixing two different structures of nanobots
	Negligible	Hacking the structure with no physical help			
		Negligible	Meaningful	High	Maximal
		Probability			

Table 2. EBIOS risk assessment

Despite the low hardware capacities, it is better to create simple rules applied by the modules that would seal the structure to any exterior threat. Each module works independently in the structure, it neither considers its orientation nor its place in the structure. Therefore, each module is concerned by securing the itself and the whole system. Moreover, block ciphers use less capacity than hash functions but needs to share its key to communicate data. Thus, a key is needed as well as a way to communicate it and a secured mean of authentication. Therefore, the system would be locked, no intruder could take part of the structure and nothing could interfere into the communications.

Indeed, some systems may need minimum security which does not affect the nanobots capacity, while others need a major security protocol even if it implies slowing down the robots and consumes more energy.

Thus, the work describes four different versions that can be used separately and based on ISO/CEI 27001 that respects the main requirements in cybersecurity: Confidentiality – Integrity – Availability.

3.2 Proposed algorithm: PROLISEAN

PROLISEAN stands for Protocol of Lightweight Security Embedded in an Architecture of Nanobots. It is a protocol for nano modular robots designed to offer the smallest footprint possible while providing a strong level of security against hacking.

Four versions of the protocol are proposed in this paper starting with the least to the most secured. The first version represents a simple authentication process. Protocols are more and more secure as we move to the next versions. In other words, the version that is less secure requires less processing power than the others. Therefore, the presentation of four versions was mandatory, leaving the choice to the end-user based on his system.

3.2.1 Version 1: Simple authentication.

Initial concept. When two nanobots connect, none of them knows if the other one belongs to its own structure. In order to authenticate, each robot will provide its source code to the other to compare it with its own. Since each robot from the same structure have the same source code, the comparison of the code is an efficient way to authenticate without adding any unnecessary lines in their program. However, if the Unknown-bot was malicious, this operation would turn into a data leak. Therefore, hashed source code is shared. Since a hash function gives the same result for the same data block, a comparison can occur. In this context, collision risk probability must be taken into consideration while pointing out a light weight hash function.

Analysis. This method allows to authenticate by comparison of the code with a few means and without providing the code to a malicious robot. Once the authentication is done, the flow of data is not protected but for a low security level. In fact, only identical robot can authenticate and this might be enough to consider for some systems.

This version has low security dialog that does not guarantee the integrity of the mean of connection. However, it remains important as it allows to make a difference between the robots that belong or not to the structure. It was designed to be time efficient for authentication by providing a first level of confidentiality.

3.2.2 Version 2: Authentication and Ciphering.

Phase 1: Authentication. The first phase of this version is extended from the previous. Moreover, the first part of the source code is hashed and used for authentication comparing and the second part is saved in the robot's memory because of its key role in the next phase. The latter will be called "identity-block" in the following parts. Figure 6 presents the first phase of version 2.

Phase 2: Encrypted communications. Once the nanobots are connected, the block cipher algorithm takes over the hash function to carry the communications out. As the block cipher is a symmetric algorithm it needs a common key in each nanobot. Therefore, the "identity block" previously created is used as a session key to encrypt the data. In this study, the robots can be connected to many other robots at the same time and the use of the same key for all the communications allows to save memory space and the energy that would be necessary to create and use one new key for each data transfer with a new robot. Figure 7 presents the second phase of this version.

Analysis. Ciphering of the communications provided confidentiality for the protocol, even if the session key is known by every robot across the structure. The hashing of the code in two parts prevents a leak of credentials during the authentication.

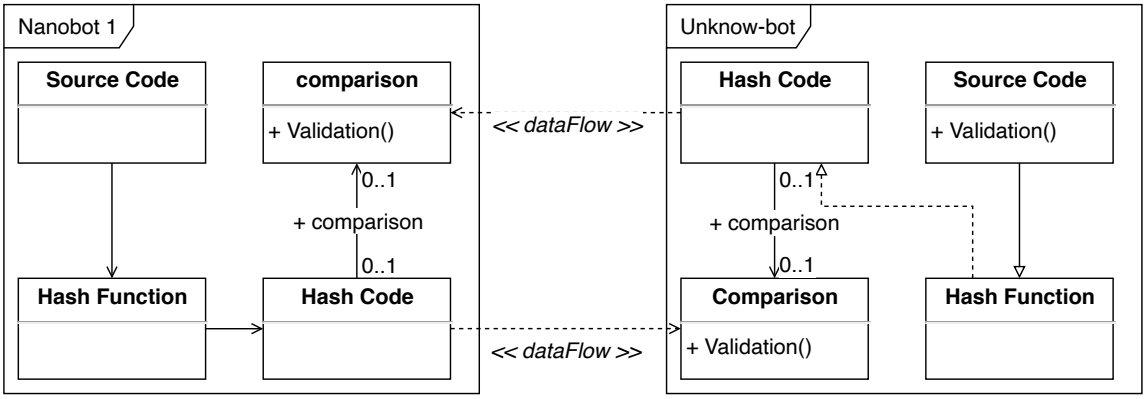


Fig. 6. Functional scheme of the first phase of version 2

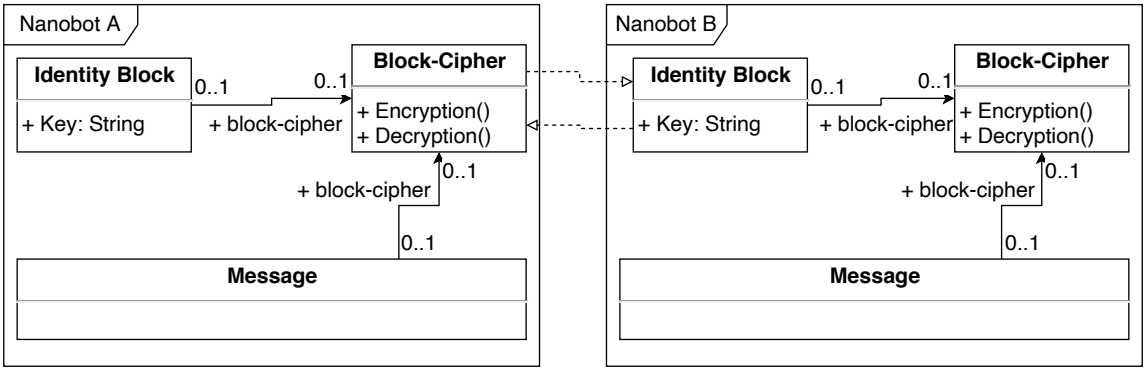


Fig. 7. Encrypted communications between 2 nanobots

Assuming that the session key is the hashing of the whole code, it would also be the data used to authenticate. A malicious nanobot could possibly copy this hashed block and authenticate with another robot of the structure and then decrypt the ciphered data. In this case, even if something copies the first credential to authenticate with another nanobot, it won't be able to understand the communications inside the robot swarm.

This protocol uses higher capacity than phase 1 and will not fit to use with very low resources. However, the use of lightweight algorithms combined with this protocol can provide an efficient solution.

To improve security, the code does not have to be cut into exact parts and the separation between the first part used for authentication and the second one used for encryption, can be kept secret by the user. As an example, for a 1,000 lines code, line 1 to 384 will be hashed to authenticate and line 385 to 1,000 to encrypt.

3.2.3 Version 3: PROLISEAN protocol.

Context. Hashing the whole code and comparing seem to provide enough security to connect two robots. However, communicating the same hashed code between thousands of robots that compound the structure is risky. A malicious robot could get this identity block and give it to another robot of the structure and thus get access.

For better explanation, the point of view of a single nanobot will be considered, unlike the two previous ones. The reader will then be able to see a robot belonging to the structure and an unknown robot trying to communicate or sometimes a more global view of an ensemble of nanobots.

Phase1: Refined authentication phase. There is no need to hash the whole code to provide high security hashed blocks and few lines are enough. In this context, random numbers cannot be used as all modules have the same seed and will generate the same random number.

Therefore, the proposed solution is based on using chaotic arithmetic sequence that would define which lines to hash. The chaotic function chosen in this paper was based on the logistic sequence presented in [24].

$$U_n = \mu U_{n-1}(1 - U_{n-1}) \tag{1}$$

where $3.57 < \mu < 4$ and $U_0 \in]0, 1[$

The integer n in this solution is the number of authentications the robot had before the current authentication, thus the robot can save the current U_n and will not have to calculate it for each authentication. This sequence, is known to be chaotic with those parameters. Which means that it won't tend to fix values or centroid when $n \longrightarrow \infty$. And as per [24], μ should be greater than 3.57 as the chaos will be set at this point. No oscillation is yet visible, in the output of the function, and slight variations in the initial population lead to radically different results. Moreover, as long as $\mu < 4$ the result will be included in $[0, 1]$, it will be multiplied by the size of the code to know which line to hash and send to the other robot to authenticate. Due to the hardware specifications of the nano-sized robots the variable n can not have a huge size, this issue is resolved by applying a modulo N in the iteration count where N is determined at the conception of the modular robot according to its own hardware limitations.

The security is not only provided using a chaotic determination of the lines but also by the choice of the parameters μ, U_0, N that can be changed before the initialization of the structure by the user itself.

To execute this phase, both nanobots must execute all the actions in the meantime as presented by Figure 8 and the sequence diagram in Figure 9

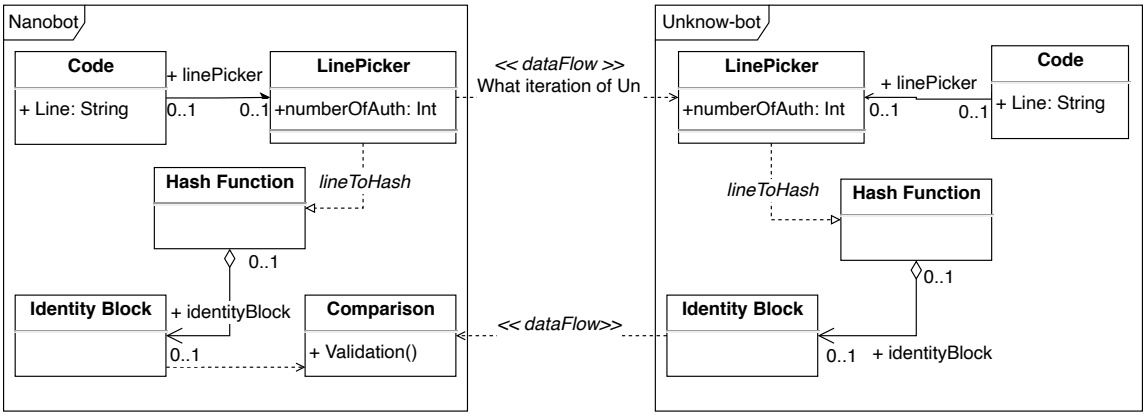


Fig. 8. Phase 1 refined

In order to allow robots to know which one must send the authentication request when they connect, it has been determined that only the one in motion sends an authentication request to the other. This was set up because a malicious

robot could just stick to another robot, wait for the identity block to be sent to it and then send it back to the same robot resulting to the hack of the authentication phase. Thus, it is necessary that the robot stationary at the time of contact is silent and waits to receive the authentication request from the one that just connected to it, to counter this flaw.

In case both robots were in motion at the time of contact, both robots would send a request. If one of them was malicious then it would perform the same scenario as before, waiting for the identity block to be sent and send it back instantly. The procedure to counter this second loophole is as follows:

- When sending an authentication request to one of its ports, if the nanobot also receives a request then the connection via that port will be processed last.
- If connections have been made with other neighbours, then forget the disputed port.
- If no other connection, send back an authentication request.

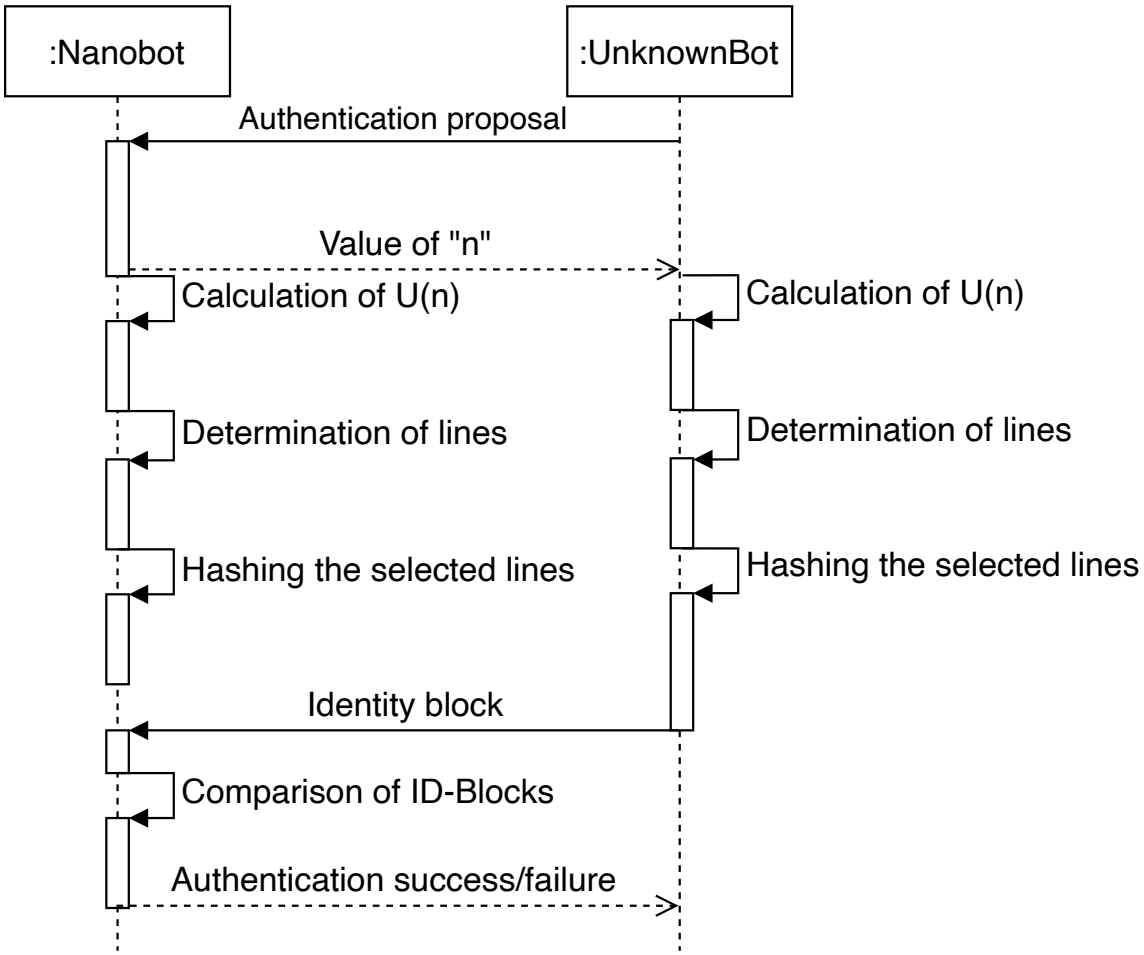


Fig. 9. Sequence diagram of the authentication phase in V3

If the robot is malicious, it will have to send a request again, get back to the initial authentication case and will be a refused authentication. Figure 10 illustrates the procedure to be adopted in the above-mentioned case. The blue robot, which knows the protocol, will first make connections with its other motionless neighbours and then, acknowledging that it is connected, waits for the red robot to send a second request. Finally, if the red robot belongs to the structure and is not connected to any other nanobot then it can make its connection with it like in Figure 10a, and if it does not belong to it then it will be blocked according to the classic authentication phase, Figure 10b.

All these conditions had to be set up to prevent a nanobot from being separated from the structure due to security rules. This is the case in the following illustration, if they both made the authentication request then any robot could connect. However, if communications were cut off after a double request then the robot with no connection would be isolated from the others.

Phase 2: Ciphered communication with multiple keys. The second phase considers that the encryption key is unique per couple of nanobots. In this configuration, each couple of nanobots confirmed that the identity block was the same for both and therefore it is used as a session key to encrypt data between both. A robot will have to save up to four or six session keys at the time and delete it to save memory space once the communication is over.

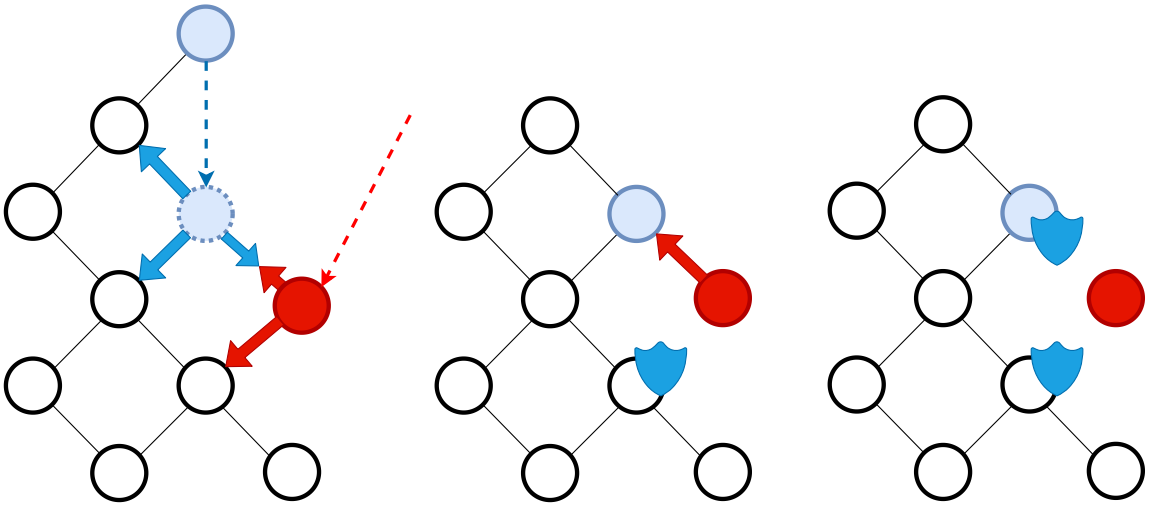
One of the conditions for using symmetric encryption is that the public key is known only by a small controlled number of entities. Ideally, if only the two entities in communication have this key, it is called a session key. In this case it is created by the two nanobots, compared and used, as one would use a session key in a traditional way of such an algorithm.

In this scenario, it is highly unlikely that three nanobots would have the same encryption key to perform two communications; i.e $ID_{A-B} = ID_{B-C}$; In order to avoid any unpleasant surprises, it is recommended in this protocol to delete the encryption keys when the communication is over and to set a sufficiently large n, where n is the number of authentications the robot had before the current authentication. Indeed, one of the situations where this could happen would be a communication in A and B that would last long enough for B to make N other communications when a C robot requests authentication and produces the same key.

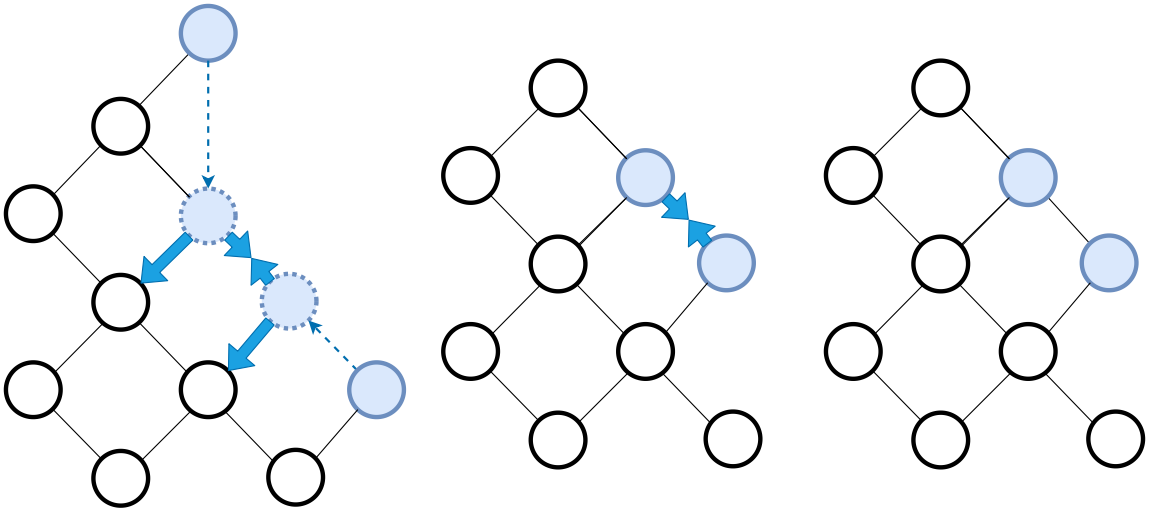
Figure 11 presents the functional scheme of this phase.

Analysis. This stage of the protocol is heavier than the previous versions but safe. The way it uses the chaotic sequence, the logistic sequence, offers three configurable parameters. Once combined with the use of a hash function and a block cipher, the parameters offer a very strong security system. Moreover, if the unknown-bots were trying to penetrate the structure, the only data they will receive is the value of n that is permanently changing with the authentications on the other hubs of the nanobots and the refusal of the authentication. Thus, no data leaks from the structure to this intruder. Like the others, it was designed to be light for low resources swarm of robots by the use of a simple arithmetic sequence and its own lines of code as keys. Figure 12, summarizes every possible interaction a robot and its neighbours could have by following this protocol.

Since the nanobots will have to handle up to six communications at the same time with multiple authentications and disconnections if the structure is in movement, the protocol may cause latency or consumes more time than if there were no security at all. However, this will not be such an issue since the miniaturisation of technologies is going deeper every year and nano scaled processors will soon be able to run this protocol. One can also notice that if all those protocols and algorithms are embedded during the conception of the nano-modular robots, the rest of the code can be modified to fit with this way of communicating and reduce the latency.



(a) When a friendly robot and a malicious robot are in movement at the same time, the friendly robot finished authentication with the modules that are were not in movement. Then tries to authenticate the malicious robot, since the malicious robots does not know the protocol it will not be authenticated.



(b) When two friendly robot are in movement at the same time, each module finishes the authentication with the modules that were not in movement then authenticates the moving module.

Fig. 10. Conflict resolution in three steps against a malicious robot (a) and with a friendly robot (b). Blue modules represent a friendly robot in movement. Red modules represent malicious modules. The desired new place of a module is depicted by dotted module and a dashed arrow. When messages are authenticated, they are represented by thick blue arrows or thick red arrows if otherwise. The blue Shields represent a secured interface.

3.2.4 *Version 4: Improved PROLISEAN protocol.*

Context. In the previous version, the fixed module is the one that is asking the moving particle for authentication. The problem occurs when two moving modules get connected at the same time. Then they will both be waiting for the

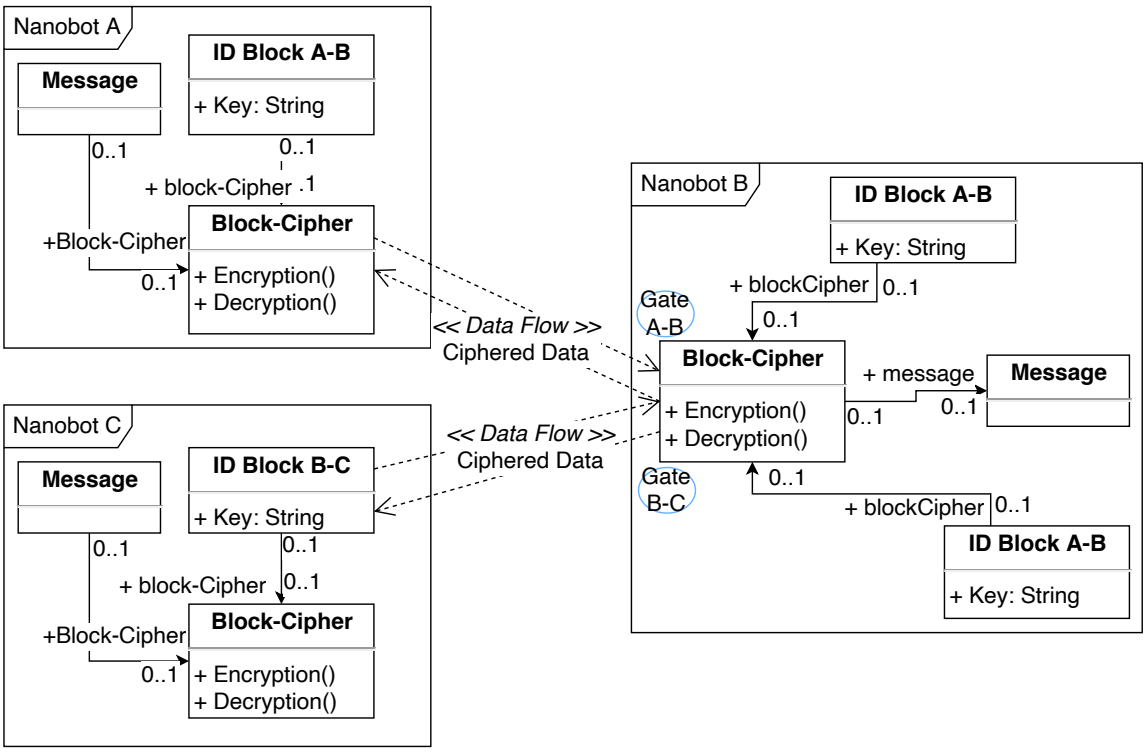


Fig. 11. Functional scheme of the ciphered communications during the second phase

other module to authenticate, and this will lead to an infinite wait time. Thus, a logical disconnection between the two subjects. This section of the paper will be proposing a different approach based on the leader election process. At the end of the leader election, the leader is a root of a spanning tree. Using the depth first approach, the leader can send an identification message that will give different ID's for each module. The identification generated will be referred to as ID [4].

Authentication phase. At the time of connection, both modules will exchange the following information: the ID given by the leader, their clock and a hashed part of their code. Module A will be receiving ID_B, $CLOCK_B$ and $H_B(Code_B)$. On the other hand, module B will be receiving ID_A, $CLOCK_A$ and $H_A(Code_A)$. When module A receives ID_B and $CLOCK_B$, it will know what part of the code was hashed. Using its hashing function, it will hash the designated part of the code and will obtain $H_A(Code_B)$,then compare it with the received $H_B(Code_B)$. If both hashing results are the same, then the modules are authentic and they will trigger the next phase of the protocol that is the encryption.

Two main conditions might occur in such a scenario. The first one is when the end user wants to intentionally merge two different structures to become one system, and the other case is when an intruder tries to connect to the swarm. When the first case occurs, there is a high risk that both modules can have the same ID, this is when the $CLOCK$ value is taken into consideration. This value, with the ID, will be used to define what line of code is used to retrieve the hashed value.

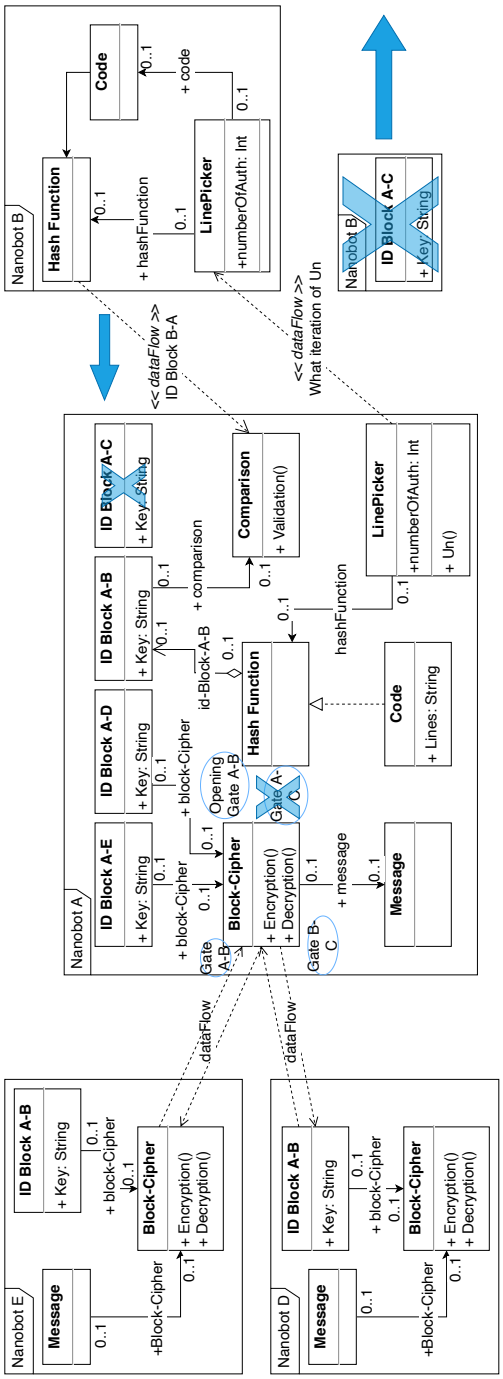


Fig. 12. Neighbours interactions following the version three of the protocol

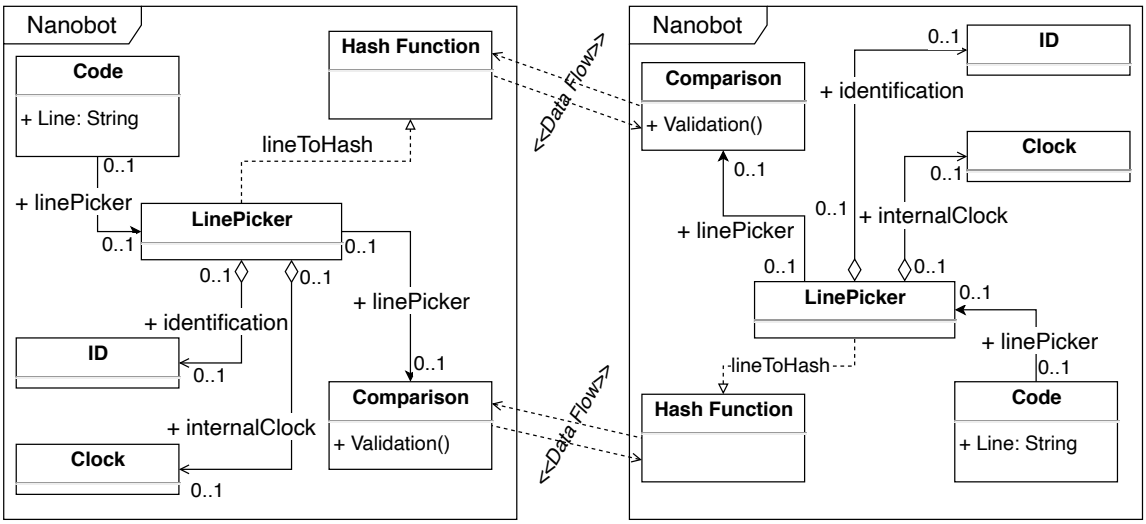


Fig. 13. Phase 1 of version 4

If an intruder module connects and tries to reply back with the same hashed value and the same *ID* and *CLOCK*, the other module will definitely reject the reply. This is related to the fact that modules of the same system cannot have the same *ID* due to the depth first method used by the leader to provide identifications for the modules. Furthermore, modules from different systems can not have the same *CLOCK*. Figure 13 shows in depth the modules, the classes and the function used to achieve phase 1 of version 4.

To execute this phase properly, both nanobots must execute all the actions in the meantime as presented by the sequence diagram in Figure 15.

As described in [26], before a module accomplishes its movement, its current neighbour will be sending the new neighbour a message informing it that there is a new module in movement that is coming to connect. A different approach of the authentication can then be introduced when particles are moving. The diagram of Figure 14 shows and explains how authentication happens when a modules is in movement.

Encrypted Communication. The encryption method used in version three can be used here since it is safe and secure.

Analysis. The use of both chaotic function and identifications generated by the leader helps making the system even more secure. Such approach requires more computational power than the previously described versions. The person creating the system is the only one who is aware of the computational power and its modules. Therefore, the end user is left with the decision of what version to choose.

3.3 Analysis

3.3.1 Limits of the method. The goal is to create an authentication method that has the least possible impact on the functioning of the whole system. It was necessary to find a method using the minimum of resources. Most of the limitations of the proposed protocol come from the environment in which the experiment takes place. Nothing has yet clearly defined what motivates robots to move in a certain way. This knowledge might be useful to design a lighter protocol.

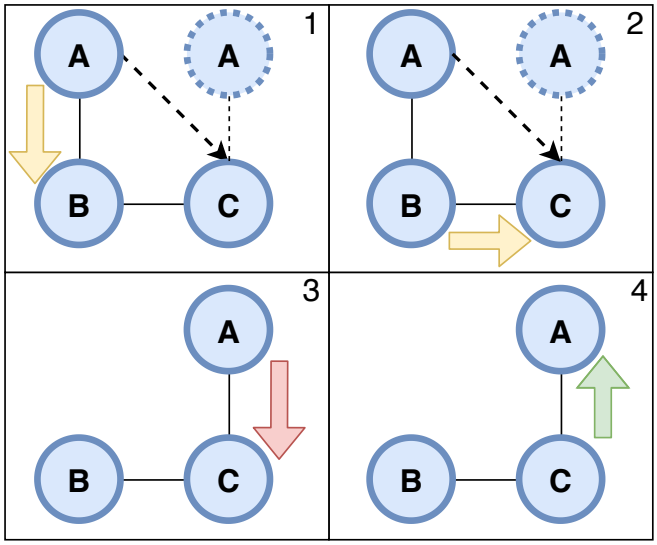


Fig. 14. Authentication approach when modules are moving and connecting to new neighbours: 1) When A decides to move and connect to C, it will inform B about its movement and sends its ID with the message; 2) B, by its turn, will inform C that A is coming to connect to it and forwards A's ID as well; 3) Whenever A accomplish its movement, it will input its ID into the chaotic function and sends the result obtained to C; 4) C will input the obtained ID from B, process it in the chaotic function and compares the results. If both results are matching, then the connection authenticated; The new position of module A is depicted by the dotted module. The dashed arrow represents the movement that will be made by A. Yellow arrows means a normal communication and message exchange. Red arrow means that the module is asking for authentication and green arrow depicts the approval of the authentication

From a more conceptual point of view, energy supply is not yet defined either. Some researchers would like to be able to supply energy via the static electricity that surrounds the material. However, others are looking to create nanoscopic batteries. This would make it possible to define the individual energy consumption of each robot and know whether the application of such a protocol does consume too much energy at the macroscopic level. Not to mention the fact that the only thing known about the hardware capabilities of the modules is that they are very limited. In order to be able to properly dimension this protocol, it will be necessary to know the memory card size, the CPU speed, the transmission rate between robots, and many other metrics.

3.3.2 Security assessment according to CIA criteria. ISO/CEI 27001 is a norm that defines security standards for IT field. Among all its content, 6 criteria can be found to assess and classify security algorithms and protocols. This upcoming section focuses on the following four: Confidentiality, Integrity, Authentication and Availability. Each one will be defined and discuss through the 4 versions of PROLISEAN. The remaining criteria are:

- Traceability, which is inexistent in this protocol because of the low hardware capacities.
- Attributability, where each entity in this protocol is responsible of one action. In this protocol, it is the case in each version. Thus, it does not allow an assessment.

Integrity. The use of the hashing technique based on considering some lines of code provides integrity to the system. Moreover, the use of chaotic hashing function will prevent adversary robot to use the same hash code. If the hashed result

compared at the receiver side is not the same, the robot will know that the message was modified during transmission. This provides integrity of messages, and the robot will ask for the message to be resent.

Confidentiality. Version 1 provides a first level of confidentiality by filtering the connections to the swarm of robots. Moreover, Version 2 enhances the confidentiality by using ciphered communications that can be only decrypted by the key-owners. Then, in the third version, a potential intruder is only able to read data from the robot that have been hacked. In the last version, it is quite impossible for an intruder to communicate with a module.

Authentication. Authentication was the main point during the development of PROLISEAN. It is the ability to guarantee the identity of the users, here it is done by comparing the source codes – or some lines – of the robots since it is supposed to be the same across the structure.

As shown previously, versions 1 and 2 are very light over this fact and allow a very fast authentication. But the result is a low degree of security in the authentication phase. The second version presents higher security level since the

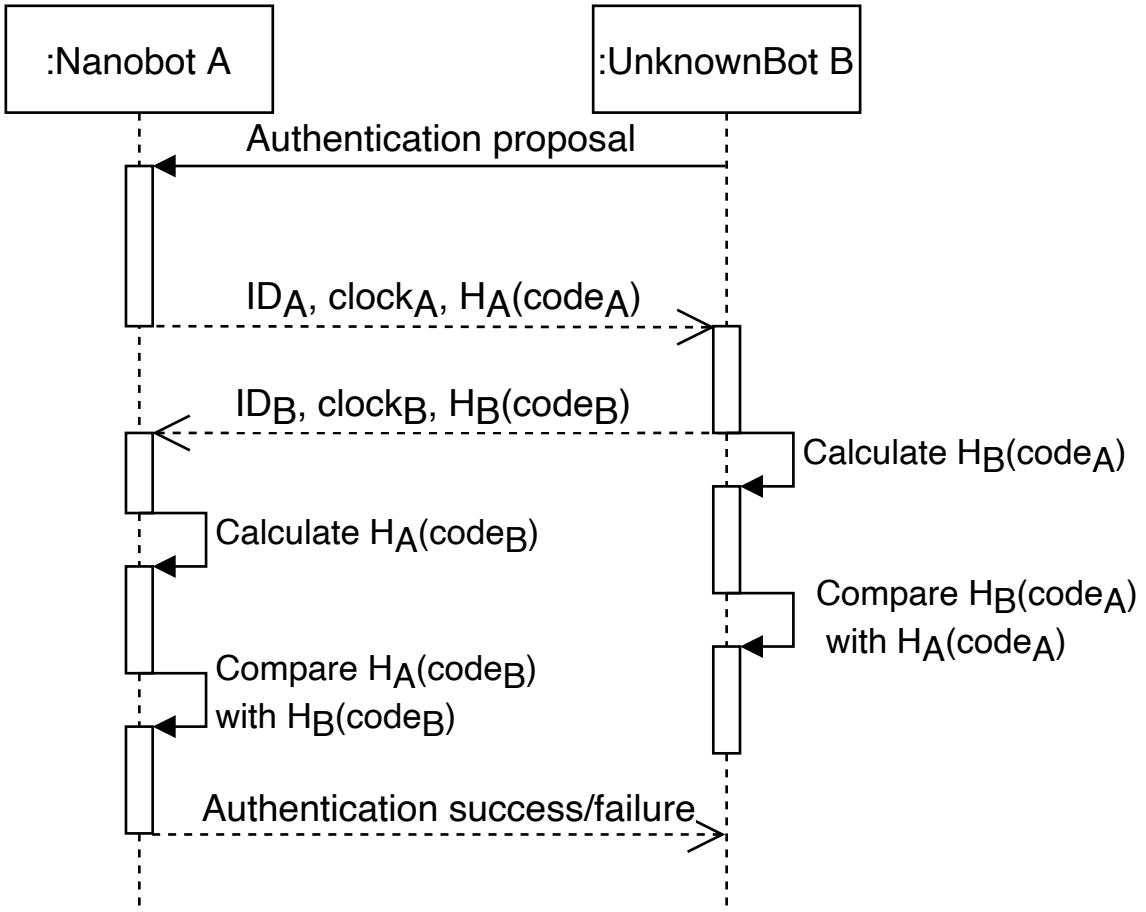


Fig. 15. Sequence diagram of the authentication phase in V4

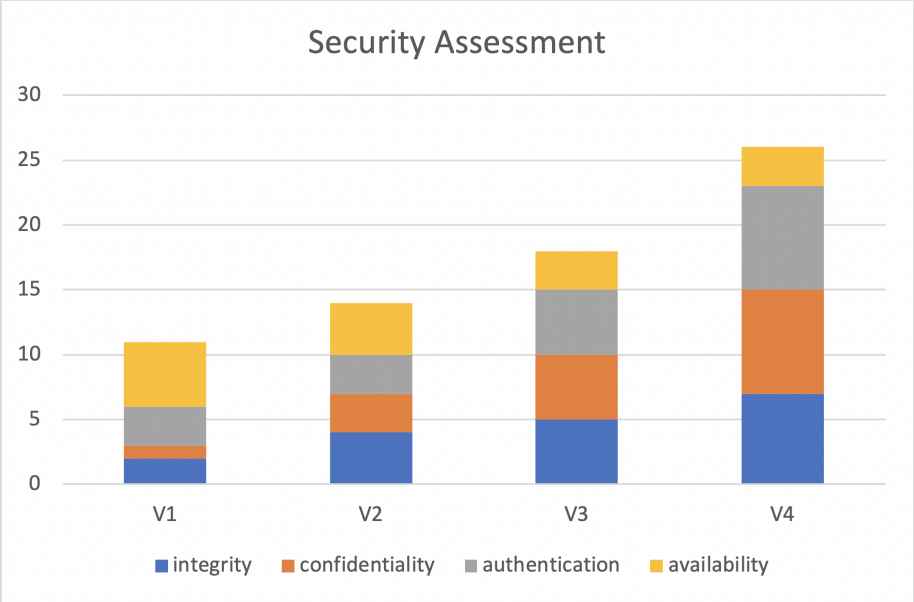


Fig. 16. Self-assessment of the 4 proposed protocols through CEI criteria

part of the code used to authenticate can vary according to the initial parameters. The third version is heavy. However, regarding authentication, the identity block to show is built for each connection.

And finally, the forth version presents the highest level of security. The usage of identifications generated by the depth first algorithm and its combination with the clock makes authentication more efficient.

Availability. Availability is the guarantee that all the services and resources remains available. The point here is that the more secure the version is, the more latency can be expected. And the assessment is inverted compared to the previous points. Moreover, in the third version, some connections may not happen between two friendly robots in the case of double authentication request. However, version four fixes this issue with the usage of the identifications.

Graph 16 summarizes the CIA criteria for the different versions of the proposed protocol.

4 ALGORITHM ANALYSIS AND PROPOSED SOLUTIONS

4.1 The block cipher

4.1.1 An efficient block cipher. Lightweight cryptography is a cryptographic algorithm or protocol tailored for implementation in constrained environments including RFID tags, sensors, contact less smart cards, health-care devices and so on. Lightweight properties are described based on target platforms. In hardware implementations, chip size and/or energy consumption are the important measures to evaluate the lightweight properties. In software implementations, the smaller code and/or RAM size are preferable for the lightweight applications [7].

Considering the previous results, the focus is on the 5 lightest algorithms:

- TEA
- KTAN

- SEA
- HIGHT
- NOEK

Ideally, this latter must have the lowest RAM, energy and memory consumption but also be fast to encrypt and decrypt. One thing that is noticed during this part of the study is that the lower the size is, the higher the energy consumption will be. Indeed, an algorithm with few lines of code written in memory will repeat its functions more to encrypt with a high security level than a big algorithm with all features in its code. So, the most efficient one should be chosen.

Collision risk. There is a probability that from two different data blocks the same hash result is obtained, this is called the risk of collision, it is expressed by hash functions designers as a probability. According to modern cyber security standards, this probability is very low but not zero. For example, for SHA3 it is estimated that a collision can occur at most every $2^{\frac{n}{2}}$ hashes, with n the output size of the blocks in bits.

Here, all the versions of this protocol rely on the result of a hash function for the authentication between two nanobots. There is, therefore, a risk that a communication may be authorized while the two parties do not belong to the same structure and consequently a risk of intrusion in the event of an attack by someone malicious. Moreover, this probability is not applied to a single robot but to all the robots in the structure and since their theoretical size is in the order of nano meter, even a structure with a volume of 1 cm^3 could contain thousands of billions of nanobots. This risk is therefore not to be underestimated.

However, several points prove the use of a hash function for authentication remains a possible solution:

- With a hashing block size of 128 bits the risk is 1 in 18 billions and with 256-bit blocks it is 1 in 3.4×10^{38} for a collision to happen.
- The only version of this protocol that relies on the hash function has been created to prevent two distinct structures from associating. And occasionally, to prevent deliberate intrusion by a malicious entity but not controlled by an external user who may understand the authentication mechanism. Thus, for the objective that was set, the hash function fulfils, at least in theory, its role. If, during a collision between two structures, a tiny fraction of the robots, in the order of 1 out of billions mix together, the structures will be able to continue to perform their initial tasks without difficulty.
- The protocols use block encryption. So the session key is only known by nanobots belonging to the same structure. Even if an intruder manages to connect, it will be impossible for him to interpret the communications because it must have the encryption key. In some models, it will also be necessary for this malicious nanobot to be able to connect to at least one second robot in the structure to intercept the data flow through it. Otherwise, it will be considered as one end and will only receive part of the information. Which drastically reduces the probability of interception of the data.

Memory use. This study focused on symmetric encryption to use the minimum number of keys possible. Using an asymmetric encryption appeared to be a more secured solution but it included the use of a directory containing millions of public keys. A solution to this issue could have been to create groups of robots with a common public key. The bigger the groups, the lower the security. That is why the lines of the robot's code are used as session key in the stages 2,3 and 4. Especially in the stage 3 and 4 in which keys are created in a chaotic way that allows both robots to communicate with a unique session key that is delete at the end of the communication.

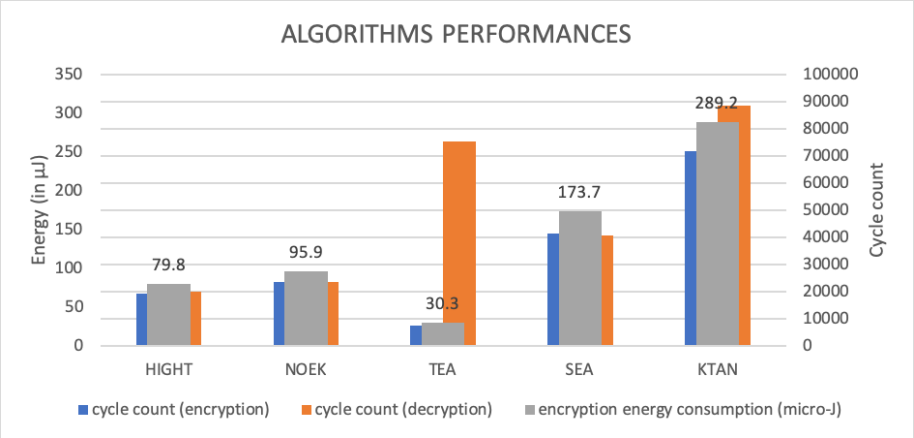


Fig. 17. Crossed analysis of ECRYPT II figures

Using the code of the robot in the encryption process also allow to save memory space. Indeed, there is no need to add lines of code such as registers for the encryption keys. Hash function provide blocks with a maximum size of 256 bits that is saved in the robots memory, multiplied by 6 communication ports it gives 1,536 bits of keys or 192 Bytes.

The diagram in Figure 17 is based on the figures of ECRYPT II (European Network of Excellence in Cryptology II) [14] and compares the cycle count for encryption and decryption of the 5 algorithms and the energy consumption for encryption. All those experimental figures were obtained by encrypting a maximum sized block of data for each algorithm.

- KTAN and SEA: The result was quite predictable regarding the size of both implementations. The number of cycles is twice higher than HIGHT or NOEK based implementations. For example, the AES based lightweight algorithm tested in ECRYPT II uses 18 time less cycles with the biggest code size of this report.
- TEA: As the energy consumption seems to be proportional with the number of cycles, the energy needed to decrypt the TEA-based algorithm will be one of the highest among the others.
- HIGHT and NOEK: Both seem to be a good compromise regarding all the parameters previously introduced. Since HIGHT algorithm has less complexity, therefore, this algorithm will be the focus of this paper.

4.1.2 HIGHT lightweight encryption algorithm. HIGHT for High Security and light weight is a block cipher designed for low resources devices with 64-bit block length and 128-bit key length for 8-bit processors. It was created to secure RFID tags that are used in contactless credit cards. Those devices are very interesting in this study since it presents the same requirement as nano-modular robots which are low hardware resources and low energy consumption.

It uses a Feistel-like 32-round iterative structure 18 that can either be hardware implemented or software implemented [17]. According to the authors, this algorithm is more efficient in hardware implemented version since Feistel structure uses XOR logical gates and additions. HIGHT mostly uses mod 2^8 additions, XORs and circular shifts of bits. It is also faster than AES-128 once embedded in a 8-bit processor and lighter in the same time by the use of only one 128-bit register in the key schedule algorithm for both encryption and decryption since it contains the 2 master keys values necessary for those operations.

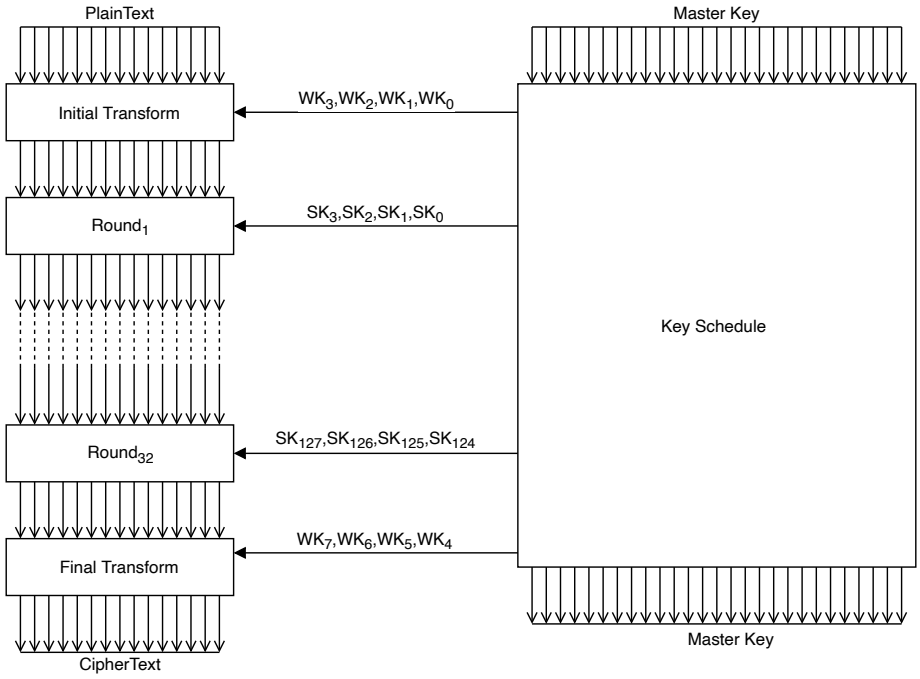


Fig. 18. Encryption process of HIGHT [31]

This algorithm has proven through different studies [14, 17] its high resistance against all kind of known attacks to be considered as resistant as classic symmetric block ciphers. To go into more detail, the initial transformation prepares the data to be processed in the rounds by concatenating the $WK_{3,2,1,0}$ whitening keys and the data to be encrypted in the form of a matrix whose coefficients are bit lists. Then the method called Round is applied 32 times to this matrix. A Round consists of 5 operations on the rows and columns of the matrix comparable in some way to the precise and known mixing of a Rubik's cube. These 5 operations include the following:

- An exchange of the coefficients of this matrix.
- Two XOR operations between the coefficients and the Subkeys $SK_{0,\ldots,128}$.
- Two bits additions modulo 2^8

Once the 32 laps have been completed, the resulting matrix is finally transformed by the Final Transform function, which undoes the last two operations of the 32^{nd} lap and then concatenates the entire matrix into a bit string that can be interpreted by the computers as a character string.

4.2 The Hash function

Considering Figure 1 the focus will be on the SPONGENT hash function family which are lightweight hash functions based on the sponge functions model. A sponge function is often used in hashing but not only, this term defines functions able to take as a parameter any size of data and provide a fixed size block in return.

It is advantageous to use such function in this study since the source code of the robot should be hashed which size is variable. Then use it later as a 128-bit key for one of the symmetric block cipher.

Those Hash functions are based on the working principle of another lightweight block cipher called PRESENT. The latter is tested in ECRYPT II report [14] but the results of the use of this algorithm to cipher communications was not efficient for the use of this study. Nevertheless, SPONGENT hash functions that are using its design principles and are the lightest presented in [14] in term of memory size and also RAM capacity use that is the reason why this report is focusing on it.

According to the creators [10], those hash functions were engineered from the very beginning to be embedded in small sensors networks such as RFID. It uses a smart management of the data generated during the hashing process by stocking the least bits possible during the hashing and serializing the implementation to the point that the logic part of the function is much smaller that the state part.

The function, like HIGHT, also uses logical gate to optimize the ratio size/RAM use and even if we could implement it to artificially use those gates. It would ruin the creators efforts to make it light.

The SPONGENT version would be more likely to be used is the 128-bit one since the 2 previously introduced block ciphers needs a 128 bits key to work. This version needs 1060 logical gates to work in its original design [10]. In this case, logical gates should also be used to make optimal use of this function. According to the various documents previously mentioned [7, 10, 14, 19], it would seem, so far, very complicated to make a hash function with a small size and a small processor footprint without using it.

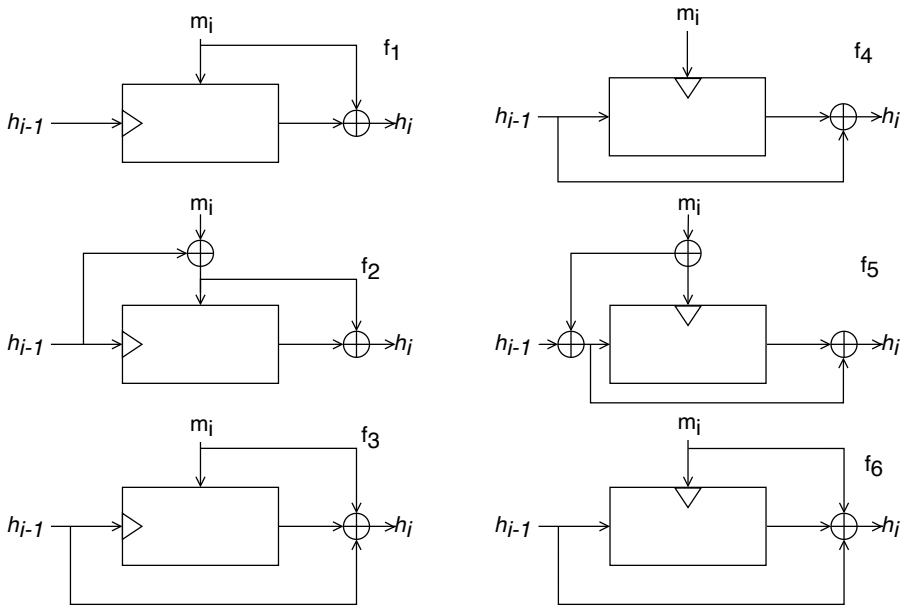


Fig. 19. Six different ways to turn a block cipher to a hash function [9]. M is the block of data to hash that has been divided into the parts noted mi ; hi is the result of a hash turn, h0 is a constant arbitrarily chosen and the central block is the encryption algorithm. The latter is used i times based on one of the models

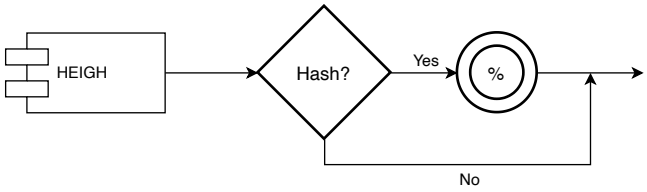


Fig. 20. Using the HEIGH algorithm for encryption and hashing simultaneously

4.3 Proposed Hashing function

To improve the the protocol, the encryption algorithm will be using both hashing function and encryption such as suggested by Preneel, Govaerts and Vandewall in [27]. The challenge was to find a way to use a block cipher that would be secured enough, designed to be embedded in low capacity devices and be able to run as a hash function within the addition of a few lines of code.

It is the final stage to create a safe, hardly breakable and lightweight system to secure nano-modular robots structures. All the security of the structure will fit into a small algorithm able to encrypt, decrypt and hash on demand and with high security standards.

This method was very interesting in the past. The first hash functions were created by modifying DES for instance, and then, with the increase of the threat in cybersecurity those functions could not handle the high security requirement. Finally, with the arrival of AES standard the interest for it pop-up back in the scientist landscape to the point that now, a lot of work about this topic is available. For now, only heavy algorithms have been turned into hash function successfully in terms of security standards respect and it would be a great challenge to create such versatile algorithm.

There are 64 ways to turn a block cipher into a hash function, Figure 19 introduces 6 examples, among them few can handle the international security standards. For a simple reason, ciphers are created to encrypt and decrypt but not hash functions. That is why, encryption keys are considered weak in the hashing standards.

Nevertheless, the idea is relatively simple in the design because it is necessary to find a way to use the encryption algorithm that makes it almost indecipherable but not due to chance. Indeed, for the same block of data, a hash function must return the same block.

As previously mentioned, this solution appears to be the most convincing among all because of the simplicity to implement, the resistance against known attacks and the low memory space required. Furthermore, [19, 28] are dealing with a Noekeon based hash function with the addition of a few code lines. It might be interesting to integrate block encryption and hash function from Noekeon. Indeed, according to [19], the Noekeon-based hash function is heavier than SPONGENT during its operation and also in the memory space. As most of the code lines of the latter would already be written in the source code of the robots with the block cipher, then it results that by using the NOEKEON-based hash function this only adds a few more lines in the code. The diagram in Figure 20 shows how an encryption algorithm can be used for hashing as well, thus reducing the memory and processing usage.

5 SIMULATION AND DIFFERENTIAL CRYPTANALYSIS

5.1 Simulations

A simulation was done on VisibleSim in order to implement and evaluate the protocol. VisibleSim is a modular robotic and programmable matter simulator based on C++. It enables developers to run event-based simulations. An event is a

Method	$P(E_k(m) = c)$	Total Rounds	Total Plain Text	Total Encryption	decrypt time 64b in year	decrypt time 32b in year	decrypt time 16b in year	decrypt time 8b in seconds
Differential	2^{-58}	11+	2^{248}	NA	$7.17 * 10^{49}$	$1.55 * 10^{31}$	$3.37 * 10^{12}$	20
Linear	2^{-54}	10	2^{228}	NA	$6.83 * 10^{43}$	$4.74 * 10^{26}$	$3.29 * 10^{09}$	0.7
Impossible Diff.	NA	8	$2^{93.6}$	$2^{109.2}$	$5.71 * 10^{32}$	$2.33 * 10^{18}$	$9.51 * 10^{3}$	0.001
Saturation	NA	16	2^{168}	2^{51}	$5.93 * 10^{25}$	$1.35 * 10^{13}$	3.06	$2.19 * 10^{-5}$
Boomerang	NA	13	2^{248}	NA	$7.17 * 10^{49}$	$1.55 * 10^{31}$	$3.37 * 10^{12}$	20

Table 3. Cryptanalysis results for the HIGHT algorithm

task executed by the module simulating one of its actuators. In this scenario, a task is a message send or received. Table 4 shows all type of messages that are being exchanged in the system.

Message Type	Message Description
AUTH_REQUEST_MSG	This message is sent by a module that is requesting authentication to connect
AUTH_GRANTED_MSG	Once this message is sent, it means that the module is authenticated to connect
AUTH_REJECTED_MSG	Once this message is sent, it means that the module is not authenticated to connect and the connection will be blocked

Table 4. Different message type

In this simulation, three modules are placed. One of the modules is the main robot in the initial system. The other two are placed in a way that one is authentic and the other one is not. Figure 21 shows in details the messages exchanged among the modules in order to approve or reject the connected robot. Furthermore, table 5 shows and explains the real values during the simulation. The columns Value1 and Value2 represents the values of the variables with index 1 and 2 respectively. Moreover, Figure 22 presents the structures simulated on VisibleSim.

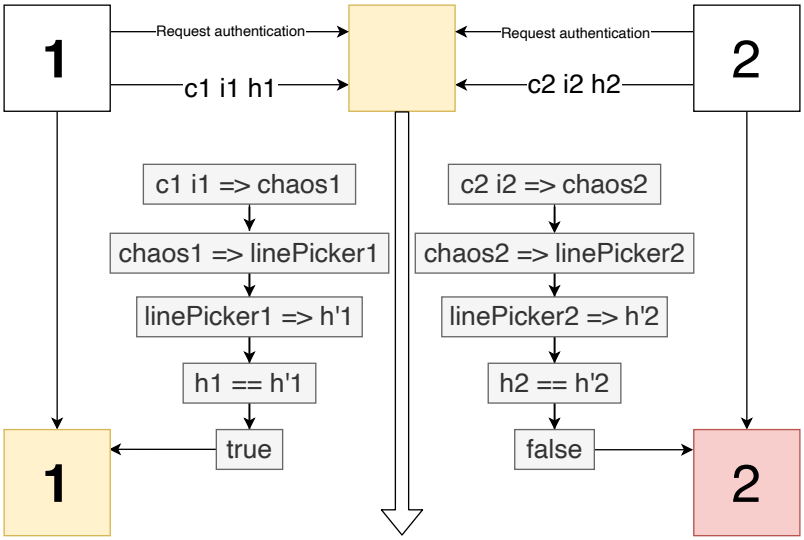
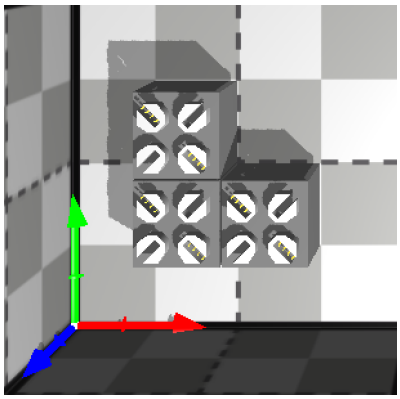


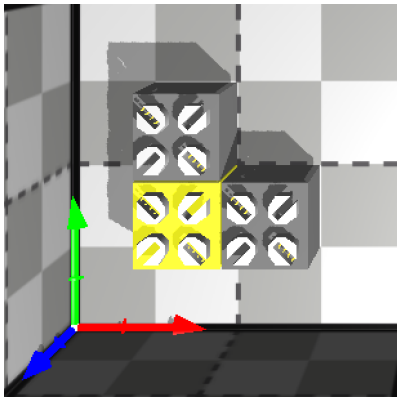
Fig. 21. Simulation graph. The initial module is colored in yellow. The bots that are trying to authenticate are in white and numbered 1 and 2. After the authentication phase, module 1 is authenticated, therefore, it is now colored in yellow. Module 2 in rejected, therefore it is colored in red.

Variable	Definition	Value1	Value2
c	Clock	1598048073	1598048068
i	Iteration	17	20
h	Hashed value	6652819590941555539	17788932521966903233
chaos	Is the value generated by the chaotic function based on c and i	47	46
linePicker	Is the line picked based on the value of chaos	year	joke
h'	The hashed result by the authenticating bot	6652819590941555539	13758745852358344340

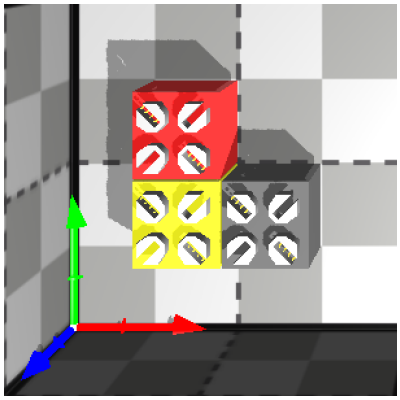
Table 5. Variables used in the simulation and their values



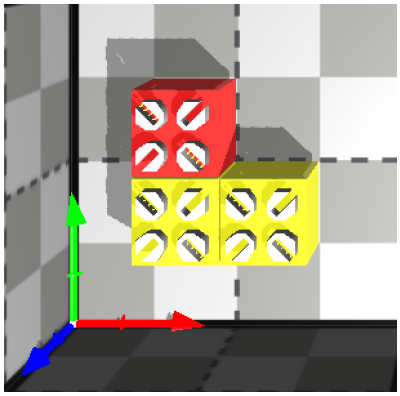
(a) This is the initial state of the system.



(b) The first bot in the system is colored in yellow



(c) This is the initial state of the system.



(d) Three spanning trees. The green and blue trees are identical.

Fig. 22. Comparison of two cases, a) and b) has the same blue and green trees but b) has yellow modules in addition. In the case b), yellow tree will dramatically change the results. The yellow tree will be loosing against the blue tree, and it will be then merged into one tree. So in this case having two identical trees is not considered as a collision.

5.2 Differential cryptanalysis

In [17], the authors provided the security analysis and statistical randomness tests of HIGHT against multiple types of attacks. Table 3 summarizes the analysis results. Those tests were made on an 8-bit key lengths to be able to reverse the encryption in a timely manner. Whereas the key length used in this scenario is a 64 bits key.

As previously discussed those testings were made on an 8-bit key lengths to be able to reverse the encryption in a timely manner. Whereas the key length used in this scenario is a 64 bits key. The fastest decryption process was the linear cryptanalysis. The number of plain texts needed to reverse the encryption on an 8 bit key was 2^{57}. When using the 64 bit key, the number of plain texts needed will be $(2^{57})^4 = 2^{228}$. As long as the modules are changing their key at least every 227 times they send a message, the encryption will remain safe. In this scenario, it is suggested to change the key used every 2 or 3 times the module sends a message.

The fastest supercomputer [22], called Summit, can make $200 * 10^{15}$ calculations per seconds. Using this value and the differential cryptanalysis method, the values in Table 3 are calculated. It is then obvious that 16bit key length for HIGHT encryption is more than enough to secure the encryption and make it impossible to decrypt it in a timely manner.

6 CONCLUSION

This paper presents the design and implementation of the first security protocol for programmable matters based on nano modular robots. It describes a solution that avoids intrusion, whether physically by a foreign nanobot or logically by an external hardware. The protocol is derived in four versions in order to adapt to the capacity of the matter and to the needs of a potential user. It describes and uses optimal lightweight cryptography and hashing algorithms from the perspective of space/time complexity and energy consumption. Moreover, a chaotic arithmetic sequence was applied to the source code of the robots, at each hashing time slot, to avoid Trojan attacks. Finally, the protocol is implemented on VisibleSim and evaluated. Results show that 16 bit key length for HIGHT encryption is enough to secure the system.

In our future work, we will study the complexity of the whole system. In this context, new techniques based on clustering the system must be developed. The leader election phase can be extended by introducing sub-leaders on different levels to have better communication complexity for large scale systems. This aspect presents many challenges from key distribution between the levels and the sub-leaders, to the security policy that must be applied in case of leaders breach. Finally, a real implementation must be considered that considers the leader election, the security protocol and the shape formation all together.

REFERENCES

[1] Hossein Ahmadzadeh and Ellips Masehian. 2015. Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization. *Artificial Intelligence* 223 (2015), 27–64.

[2] Hossein Ahmadzadeh, Ellips Masehian, and Masoud Asadpour. 2016. Modular Robotic Systems: Characteristics and Applications. *J. Intell. Robotics Syst.* 81, 3-4 (March 2016), 317–357. https://doi.org/10.1007/s10846-015-0237-8

[3] Ahmet Arış, Sema F Oktuğ, and Sıddıka Berna Örs Yalçın. 2015. Internet-of-things security: Denial of service attacks. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)*. IEEE, 903–906.

[4] Joseph Assaker, Abdallah Makhoul, Julien Bourgeois1, and Jacques Demerjian. 2020. A Unique Identifier Assignment Method for Distributed Modular Robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020*. IEEE.

[5] Jacques M Bahi, Christophe Guyeux, and Abdallah Makhoul. 2010. Secure data aggregation in wireless sensor networks: homomorphism versus watermarking approach. In *International Conference on Ad Hoc Networks*. Springer, 344–358.

[6] Jad Bassil, Mohamad Moussa, Abdallah Makhoul, Benoît Piranda, and Julien Bourgeois. 2020. Linear Distributed Clustering Algorithm for Modular Robots BasedProgrammable Matter. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020*. IEEE.

[7] Isha Bhardwaj, Ajay Kumar, and Manu Bansal. 2017. A review on lightweight cryptography algorithms for data security and authentication in IoTs. In *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 504–509.

[8] Alex Biryukov and Léo Paul Perrin. 2017. State of the art in lightweight symmetric cryptography.

[9] John Black, Phillip Rogaway, and Thomas Shrimpton. 2002. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Annual International Cryptology Conference*. Springer, 320–335.

[10] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. 2011. SPONGENT: A lightweight hash function. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 312–325.

[11] Zoran Cekerevac, Zdenek Dvorak, Ludmila Prigoda, and Petar Cekerevac. 2017. Internet of things and the man-in-the-middle attacks–security and economic risks. *MEST Journal* 5, 2 (2017), 15–25.

[12] Zahra Derakhshandeh. 2017. *Algorithmic Foundations of Self-Organizing Programmable Matter*. Ph.D. Dissertation. Arizona State University.

[13] Zahra Derakhshandeh, Robert Gmyr, Andréa W Richa, Christian Scheideler, and Thim Strothmann. 2016. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 289–299.

[14] ECRYPT Francois-Xavier Standaert, II and D SYM. 2010. ECRYPT II, Final Lightweight Cryptography Status Report. (2010).

[15] Christophe Guyeux, Abdallah Makhoul, and Jacques M. Bahi. 2014. A Security Framework for Wireless Sensor Networks: Theory and Practice. In *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, Sumitra Reddy (Ed.). 269–274.

[16] Neil Hanley and Maire ONeill. 2012. Hardware comparison of the ISO/IEC 29192-2 block ciphers. In *2012 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 57–62.

[17] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, et al. 2006. HIGHT: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 46–59.

[18] Vittorio P Illiano and Emil C Lupu. 2015. Detecting malicious data injections in event detection wireless sensor networks. *IEEE Transactions on Network and service management* 12, 3 (2015), 496–510.

[19] Masanobu Katagi, Shiho Moriai, et al. 2008. Lightweight cryptography for the internet of things. *Sony Corporation* (2008), 7–10.

[20] Vinod Kumar, Rakesh Kumar Jha, and Sanjeev Jain. 2020. NB-IoT Security: A Survey. *Wireless Personal Communications* 113, 4 (2020), 2661–2708. https://doi.org/10.1007/s11277-020-07346-7

[21] Giuseppe A. Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. 2017. Brief Announcement: Shape Formation by Programmable Particles. In *31st International Symposium on Distributed Computing (DISC 2017) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 91)*, Andréa W. Richa (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 48:1–48:3. https://doi.org/10.4230/LIPIcs.DISC.2017.48

[22] Morgan L McCorkle. 2018 (accessed November 3, 2020). *ORNL Launches Summit Supercomputer*. https://www.ornl.gov/news/ornl-launches-summit-supercomputer.

[23] Sunday Oyinlola Ogundoyin. 2020. An autonomous lightweight conditional privacy-preserving authentication scheme with provable security for vehicular ad-hoc networks. *International Journal of Computers and Applications* 42, 2 (2020), 196–211.

[24] Daniel Perrin. 2008. La suite logistique et le chaos. *Rapport technique, Dép. Math. d'Orsay., Univ. de Paris-Sud, France* 22 (2008).

[25] Benoit Piranda, S Fekete, A Richa, K Römer, and C Scheideler. 2016. Visiblesim: Your simulator for programmable matter. In *Algorithmic Foundations of Programmable Matter (Dagstuhl Seminar 16271)*. https://projects.femto-st.fr/projet-visiblesim/en.

[26] Benoît Piranda, Guillaume J. Laurent, Julien Bourgeois, Cédric Clévy, Sebastian Möbes, and Nadine Le Fort-Piat. 2013. A new concept of planar self-reconfigurable modular robot for conveying microparts. *Mechatronics* 23, 7 (Oct. 2013), 906–915. https://doi.org/10.1016/j.mechatronics.2013.08.009

[27] Bart Preneel, René Govaerts, and Joos Vandewalle. 1993. Hash functions based on block ciphers: A synthetic approach. In *Annual International Cryptology Conference*. Springer, 368–378.

[28] Phillip Rogaway and John Steinberger. 2008. Constructing cryptographic hash functions from fixed-key blockciphers. In *Annual International Cryptology Conference*. Springer, 433–450.

[29] Iván Santos-González, Alexandra Rivero-García, Mike Burmester, Jorge Munilla, and Pino Caballero-Gil. 2020. Secure lightweight password authenticated key exchange for heterogeneous wireless sensor networks. *Information Systems* 88 (2020), 101423.

[30] Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. 2020. Lightweight Security Algorithms for Resource-constrained IoT-based Sensor Nodes. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[31] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. 2007. The 128-bit blockcipher CLEFIA. In *International workshop on fast software encryption*. Springer, 181–195.

[32] Pierre Thalamy, Benoît Piranda, and Julien Bourgeois. 2019. A survey of autonomous self-reconfiguration methods for robot-based programmable matter. *Robotics and Autonomous Systems* 120 (2019), 103242.

[33] Mohammad Wazid, Ashok Kumar Das, Vivekananda Bhat, and Athanasios V Vasilakos. 2020. LAM-CIoT: Lightweight authentication mechanism in cloud-based IoT environment. *Journal of Network and Computer Applications* 150 (2020), 102496.