BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 3
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0024

Application of Amicable Numbers in Cryptographic Data Protection Methods

Tomasz Krokosz

Department of Information Technology, Poznań University of Economics and Business, Poland
E-mail: tomasz.krokosz@ue.poznan.pl

Abstract: *Cryptographic data protection algorithms require specific resources for proper implementation and security. Certain environments, particularly those involving Internet of Things devices, cannot provide the necessary set of measures, including processor speed, memory size, and communication bandwidth, with an acceptable response time and a limitation on battery power consumption. Unfortunately, effective and known data protection mechanisms, such as public key infrastructure, have no potential application in this case. An alternative is to develop other solutions that will ensure an appropriate level of security and will be dedicated to devices with minimal computing and communication resources. This paper proposes a new method to access the resources or to verify access rights using so-called amicable numbers as a base for the authentication mechanism. The mathematical properties of amicable numbers can form the basis for developing new algorithms that are highly needed in the IoT world and relate to fundamental data security and privacy protection issues.*

Keywords: *Amicable numbers, Data protection, Data security, Privacy, Encryption, Internet of Things.*

1. Introduction

The name “Internet of Things”, in fact, appeared in the 1980s but was formally defined by Kevin Ashton [1] several years later. The concept was first publicly introduced in 1999 as a digital network formed by interconnected objects, simultaneously connected to the outside world. Since the presentation of the official binding term, the idea of the Internet of Things (IoT) [2] has been dynamically developed and expanded. Over the years, the vigorous commitment of researchers has contributed to the creation of a new branch of the development of information science and new technologies. Private citizens, corporate employees, and consumers are now witnessing technology’s enormous impact on their daily lives [3]. The subject of the Internet of Things can be either a small sensor, for example, responsible for measuring the level of smog in the air, or objects of much larger dimensions (a larger casing does not mean that more efficient components have been installed), e.g., “smart” light bulbs, cleaning robots, bathroom scales, air purifiers, televisions, or recently autonomous vehicles and robots. In addition to the aforementioned modest set of applications, extended with all wearables accessories, including training bands or watches, the technology offensive includes, among others, the food industry, energy, waste management, medical devices, etc. [4]. The use of the Internet of Things in the industry is the basis of the fourth industrial revolution. Intelligent IoT devices and accessories are used to increase flexibility, automate tasks as well and optimize processes. As a result of their use, productivity is improved while reducing costs, guaranteeing more effective and efficient work (e.g., as a result of earlier detection of potential failures). The most frequently implemented communication channels within IoT devices are WiFi and Bluetooth, as well as specialized technologies such as Zigbee and zWave. Data flow enabled in this way determines the use of devices to learn, analyze, and adjust customers' needs, and, consequently, better match the advertising message, services, or offered products.

The flow of information between IoT devices is intended to contribute ultimately to cooperation in the service of humans. For example, cameras and motion sensors are used for monitoring. When an undesirable, unexpected activity is detected (e.g., initiated by a burglar or a wild animal), appropriate notifications are sent to the proper people, i.e., law enforcement agencies or security personnel. In this case, another solution could aim to activate the alarm and lock the doors, thus preventing the intruder from escaping – or, on the contrary, to activate the siren to scare the animal away. Note that the “intelligent” reaction of the system to the detection of humans and animals should be different, which cannot be achieved by using only simple automation. Note also that in the case of open data transmission protocols, the system can be easily deceived, e.g., by substituting a fake sensor or blocking the camera. Therefore, proper data protection and verification of system elements that exchange such data becomes necessary.

As a rule, advanced cryptographic measures and mechanisms are used for protection, including public/private key infrastructure [5, 6], symmetric encryption, stream encryption [7], digital signature [8], certificates, and elliptical curves [9]. Their functionality ensures confidentiality, integrity, accountability, anonymity, and availability. According to the Kerckhoffs ([see 10]) principle, their security should rely on keeping the key secret, not hiding the secret of an algorithm.

Unfortunately, although popular and generally available data protection measures are effective in the classic sense, they are not applicable in the Internet of Things environment. This is due to the technical limitations of the devices used. The limit applies not only to the processor and memory but also to communication methods. Some of the devices work only in the transmitter mode, so the data exchange operation is impossible in this case. Performing encryption takes a certain amount of time, but the time required to do such calculations on a small device will be unacceptable because of limited resources. In addition, the performance of such a cryptographic transformation will negatively affect the battery life, effectively reducing its operation time and forcing the use of additional mechanisms, e.g., power supply, charging, and heat dissipation. Therefore, alternative solutions concerning the Internet of Things and secure data transfer should be sought. While guaranteeing security and confidentiality, these solutions will not be so demanding and computationally complex that they cannot be successfully used in small devices and IoT networks. Moreover, the fact that the devices belong to the same network does not necessarily mean that they may communicate with any other node in the network. If it is necessary to apply such a limitation, additional measures to guarantee it will require additional resources – computingg/memory/power.

The solution depicted in the text is related to the properties of amicable numbers. This property is manifested in such a way that for a pair of numbers a and b, the summary of the proper divisors of a (i.e., less than the same) is equal to the number b, while the sum of all proper divisors of b corresponds to the number a. Using the aforementioned property, it is possible to successfully implement a new safety method, to grant access to the resource or to act as a confirmation of identity, as further described in the paper. Aware of the limitations of some IoT devices' computing power and communication methods, the text proposes a different, secure data protection method. While implementing the method, the entire spectrum of calculations and transmissions, which drastically reduce the devices' operation time, will not be required.

The remainder of the paper is as follows. The second chapter describes amicable numbers. Cryptographic patterns of using the presented solution are presented in the third chapter. Examples of usage scenarios with numbers and the function *f*(*x*) are provided in the fourth chapter. As part of the proposal, developed implementation was developed to validate the theoretical concepts and demonstrate their full applicability. The code was written in C and ran on the ESP32-C6 device. Details of the implementation and obtained results are presented in the penultimate chapter. The summary and conclusions are included in the last chapter.

2. Amicable numbers

Amicable numbers are a pair of natural numbers, the difference being that the sum of the divisors of each number is equal to the second number [11]. The first pair was designated by Pythagoras, who, when asked about a friend, replied that it is a ratio of the numbers 220 and 284. Values are the proper divisors of the first-mentioned number:

(1) $$D_{220} = \{1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110\}.$$

While computing the summary of all divisors, the obtained result will be equal to 284:

(2) $$1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284.$$

By doing the same for the second number and writing all its proper divisors, we get the values:

(3) $$D_{284} = \{1, 2, 4, 71, 142\}.$$

The summary of the divisors of 284 will be as follows:

(4) $$1 + 2 + 4 + 71 + 142 = 220.$$

This specificity, characterized by the fact that each number is the sum of the divisors of the other, excluding both of these numbers, is a property of amicable numbers. Although the relationship characterizing such a pair does not seem complicated, there has not been a single formula by which such pairs can be efficiently determined. Unlike other numbers known to the Pythagoreans, such as perfect numbers (the sum of the proper divisors of a number equals that number), these amicable numbers provided researchers with quite a challenge [12]. This race was started by the Arab mathematician al-Sabi Thabit Ibn Qurrah al-Harrani (called Sabit), who made the following theorem. If each of the three natural numbers:

(5) $$p = 3 \times 2^{n-1} - 1,\ q = 3 \times 2^{n} - 1,\ r = 9 \times 2^{2n} - 1,$$

is prime, then:

(6) $$A = 2^{n}pq,$$

(7) $$B = 2^{n}r.$$

They are amicable numbers.

It can be seen that for parameters n = 2, p = 5, q = 11, and r = 71, performing the calculations by substituting the appropriate values, we will get the first of the pair of amicable numbers, i.e., 220 and 284. Pattern will also be used for pairs n = 4, p = 23, q = 47, r = 1151 (A = 17,296, B = 18,416), and n = 7, p = 191, q = 383 and r = 73,727 (A = 9,363,584, B = 9,437,056). However, it is unknown whether Sabit dedicated a solution to the determination of amicable numbers for n > 2! (2 factorial). The Moroccan scholar Ibn al-Banny designated the second pair of amicable numbers (A = 17,296, B = 18,416), although until the end of the 1970 s, this merit was attributed to Fermat. Significant discoveries in the subject of amicable numbers took place at the beginning of the 17th century. Two independent researchers, Fermat (in 1636) and Descartes (in 1638), independently and without cooperation, rediscovered Sabito's theorem and put forward further conclusions and theorems. They used the arithmetic function $\sigma(n)$, which describes the sum of the natural divisors of n for every $n \in \mathbb{N}$. The most important two properties of this function are as follows:

(8) $$\sigma(ab) = \sigma(a)\sigma(b).$$

For any relatively prime numbers a and $b \in \mathbb{N}$ we should also mention another property:

(9) $$\sigma(pn) = \sigma(p^{n}) = 1 + p + \cdots + p^{n} = \frac{p^{n+1}-1}{p-1}.$$

For any prime p, and $n \in \mathbb{N}$. According to what has been written, the sum of the proper divisors of a given number $a \in \mathbb{N}$ is equal to $\sigma(a) - a$. analytical description for a pair of amicable numbers a and b will be as follows:

(10) $$\sigma(a) - a = b, \text{and } \sigma(b) - b = a,$$

while the condition necessary to be satisfied by the pair of numbers a and b to be able to include them in the friendly set is as follows:

(11) $$\sigma(a) = a + b = \sigma(b).$$

Applying the markings as in the case of Sabito's theorem, we obtain

(12) $$\sigma(A) = (2^{n+1} - 1)(p + 1)(q + 1), \sigma(B) = (2^{n+1} - 1)(r + 1),$$

wherein

(13) $$(p + 1)(q + 1) = 9 \times 2^{2n-1} = r + 1,$$

meaning that

(14) $$A + B = 2^{n}(pq + r),$$

and

(15) $2^n(pq + r) = (2^{n+1} - 1)(r + 1) \leftrightarrow pq + r = 9 \times 2^{n-1}(2^{n+1} - 1)$,
which is proof of Sabito's theorem.

Further significant discoveries in the subject of amicable numbers came after the publication of the works of Leonhard Euler. Thanks to his work on amicable numbers, he has made tremendous progress, as he discovered 59 pairs of such numbers and is the author of five different methods for determining them. One of them is related to the Sabito theorem presented earlier and is as follows: Let $m, n \in \mathbb{N}$, $m < n$, and $g := 2^{n+m}+1$. If numbers

(16) $p = 2^m g - 1,\ q = 2^n g - 1,\ r = 2^{n+m} g^2 - 1$

are prime numbers, then numbers

(17) $A = 2^n pq$ and $B = 2^n r$,

belong to the group of amicable numbers.

There are forty pairs of amicable numbers in which both numbers are less than a million (max: A = 898,216, B = 980,984), and there are forty-two of those in which one of them is greater than a million (A = 998,104, B = 1,043,096). Not only the group of mathematicians to whom we owe studies on prime numbers, i.e., Mersenne, and the aforementioned Fermat and Descartes, dealt with the set of amicable numbers. It was also a topic raised by Euler, Legendre, Chebyshev, and the Polish mathematician Jan Brożek. By 2001, a million different pairs of such numbers were known. Over the next six years, eleven million more pairs were discovered, and as of now, we know over one billion two hundred million such pairs. It is not known whether there are infinitely many pairs of amicable numbers. The proof that the ratio of amicable numbers no greater than x tends to zero when x tends to infinity was provided by Paul Erdos. On the other hand, the proof that if pairs of amicable numbers were infinitely many, then the series composed of the reciprocals of all numbers is convergent was provided by Carl Pomerance.

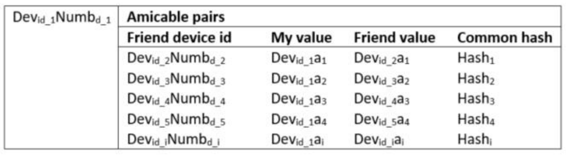
3. Exemplary cryptographic usage schemes

A group of computers and peripheral devices (printers, mass storage devices) that are connected to share resources and exchange data is called a computer network [13]. Networks allow not only communication between devices, sharing and sharing files, but also provide, among others, the functionality of sharing infrastructure or the ability to run remotely and query services. The basic set of functions offered is available (default) in any type of network, regardless of whether it is LAN, CAN, MAN, or WAN, which are different in terms of size, i.e., their range. While all types of networks have different requirements, they will also have different expectations and goals (e.g., home and corporate networks), and access to them is always protected. The condition for accessing the network is the need to know the password (or proof of such knowledge). This will confirm that the one with such knowledge (password) is who he claims to be; therefore, access should be granted. If the subject obtained such access, it means that the authentication process has been completed. Its declared identity was confirmed, so the verification engine failed to return an error message. For example, in WiFi networks for this purpose we use the PSK Algorithm – Pre Shared Key Algorithm [14, 15]. However, these traditional authentication methods may fail in the case of an IoT network. Firstly, the smallest devices are not protected at all – thus, it is hard to provide access grants specific to them for those who are already granted access to the network as a whole. Secondly, for bigger devices, the resources used to check the access rights may substantially reduce operational time and other resources (CPU, memory, communication bandwidth, etc.). Thus, the classical methods are only applicable for a small set of devices. The problem is complicated if we take into account some specific services provided by the devices – each service should be treated as an independent entity, even if provided by the same device.

An authentication is, therefore, the first stage of user/device verification, and failure, in this case, prevents access and use of the offered functionalities. On the other hand, if such access is guaranteed as a result of successful authentication, the possibility to perform a specific action will always be preceded by an additional verification step, i.e., passing the authorization process. A widespread case is one in which the device completes the authentication process and gains the ability to communicate with all devices (not hidden in, e.g., internal networks) that are part of the network infrastructure. This fact, of course, is desirable and consistent with the intent of the network, but there may be occasions when such communication should be limited to a defined extent or prevented altogether.

In the solutions and default proposals proposed so far, but concerning another issue – access, for instance, to files in the network [16-19], a mapping is offered using an adequately prepared data structure storing information about who has access and to what. This is a convenient approach because everything is managed from one level and gives a preview and the ability to identify acceptable communication paths. However, this solution is subject to certain limitations. First of all, there is a need to remember this entire set, and if the structure is extended with additional parameters, the administrator should keep in mind the subsequent updates of individual components (e.g., the validity date range of a given entitlement). Secondly, to obtain information on whether the communication is allowed, the system must perform a verification operation, i.e., ask the entity (similar to a trusted third party) whether it is possible. If the mapping set is deleted or arbitrarily manipulated, it will be necessary to remove all associations and re-establish them from the beginning (or restore, if possible), causing all data exchanges to be suspended for some time. With each query, whether a given communication session has expired and is still in force is associated with sending the request, processing it by the entity, and sending an appropriate response. All this effectively slows down the entire process and increases response time. The more significant number of sent messages is accompanied by more intensive network traffic. This way, the available bandwidth is occupied that could be used otherwise.

In the paper, we suggest using amicable numbers to solve the above problem. Consider a diagram in which amicable numbers are a secret pair. Their values are known to devices (i.e., each knows its number and the pair corresponding to the other device), and they are not disclosed at the time of communication. As a result, the centralized table with communication permissions mapping is not obligatory, while the resignation from this component in its current form will not be irrelevant for the set of necessary confirmations and time response. Accordingly, the structure of the proposed solution could, in its simplest version, take the form shown in Fig. 1.



$\text{Dev}_{\text{id\_1}}\text{Numb}_{\text{d\_1}}$	**Amicable pairs**			
	Friend device id	**My value**	**Friend value**	**Common hash**
	$\text{Dev}_{\text{id\_2}}\text{Numb}_{\text{d\_2}}$	$\text{Dev}_{\text{id\_1}}a_1$	$\text{Dev}_{\text{id\_2}}a_1$	Hash_1
	$\text{Dev}_{\text{id\_3}}\text{Numb}_{\text{d\_3}}$	$\text{Dev}_{\text{id\_1}}a_2$	$\text{Dev}_{\text{id\_3}}a_2$	Hash_2
	$\text{Dev}_{\text{id\_4}}\text{Numb}_{\text{d\_4}}$	$\text{Dev}_{\text{id\_1}}a_3$	$\text{Dev}_{\text{id\_4}}a_3$	Hash_3
	$\text{Dev}_{\text{id\_5}}\text{Numb}_{\text{d\_5}}$	$\text{Dev}_{\text{id\_1}}a_4$	$\text{Dev}_{\text{id\_5}}a_4$	Hash_4
	$\text{Dev}_{\text{id\_i}}\text{Numb}_{\text{d\_i}}$	$\text{Dev}_{\text{id\_1}}a_i$	$\text{Dev}_{\text{id\_i}}a_i$	Hash_i

Fig. 1. Representation of amicable numbers structure data

As mentioned before, each device is uniquely identified on the network. In the proposed scheme, the device sending data $\text{Dev}_{\text{id\_1}}$ has a unique number labeled $\text{Numb}_{\text{d\_1}}$. Moreover, in place of the centralized module responsible for the organization of connections (to which a request would have to be sent each time), this information is already stored in the device. Since the device can send/receive data with any number of nodes, there is a need to create a set consisting of the identifiers of these devices $\text{Numb}_{\text{d\_i}}$, first (own) and the second (device's) amicable number of the pair and the parameter hash_i. At the time of sending the data, the sent packet is supplemented with a certificate, which proves (verified by the other party) that the transmission comes from a friendly device. To create it, one of the cryptographic primitives will be used, namely the one-way hash function [20]. That way, if the device with the identifier/address $\text{Numb}_{\text{d\_1}}$ wants to send data to the device $\text{Numb}_{\text{d\_4}}$, it generates proof Hash_4:

(18) $$\text{Hash}_4 = \left(\text{Dev}_{\text{id\_1}}\text{Numb}_{d1}\ \&\&\ \text{Dev}_{\text{id}_1}a_1\ \&\&\ \text{Dev}_{\text{id}_5}\text{Numb}_{d5}\ \&\&\ \text{Dev}_{\text{id}_5}a_4\right).$$

Once the target node receives the packet, it first checks the proof. This is done by generating a similar acknowledgment of receipt and comparing the result obtained with the value received – the proof of shipment. If the proofs are identical, it means that the recipient found in its set the device that sent the message to it, selected the necessary parameters, and generated a hash, which turned out to be the same as the received one.

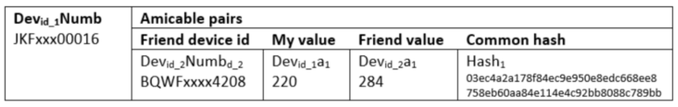
The second and different cryptographic scheme is more complicated than the previous one. For efficient implementation of anonymity, it involves a mechanism modeled on the group isomorphism and evidence with zero knowledge [21]. They are a particular case of challenge-answer protocols [22, 23], in which one of the participants proves that he has specific information and proves it by answering random questions. In this case, we will prove that the devices are "friendly" without revealing any information about the amicable numbers. None of the messages sent by the proving person reveals information about the secret. The primary flow of the algorithm is as follows: Party A claims to know the isomorphism between G1 and G2. Party B (verifier) requests proof. In response, side A sends the graph H, but receives from the verifier the number v = 1 or v = 2 in response. A sends an isomorphism between H and G1 or G2, depending on the value v. If for A side isomorphism between G1 and G2 is not foreign, this side can generate H by changing the graph's vertex labels and then generating an isomorphism to one or the other graph. The Authors suggest generating functions for each of the friendly pairs of functions $f(x)$. This means that both sides of the communication have one shared function, while the other party's task is to prove that it does know it. The mentioned function will depend entirely on amicable numbers because its points will be the coordinates determined on the basis of the divisors of both numbers. If number *A* has divisors $a1$, $a2$, $a3$, $a4$, while the divisors of its amicable number *B* are values $b1$, $b2$, $b3$, $b4$, then the coordinates could be as follows: ($a1$; $b1$), ($a2$; $b2$), ($a3$; $b3$), ($a4$; $b4$). They will be used to determine the function $f(x)$, to demand the answer in the next steps, what is the value of the function $f(x)$ for the given x.

Two different diagrams are detailed in the next section. Theoretical considerations were extended with a working implementation of the solution.

4. Scenarios of usage and application examples

Proposal to use amicable numbers in the topic of de facto authorization, outlined in the previous chapter, will be presented in the described scenarios with specific values. First, a scenario for generating a schema using the common hash and function will be given. The first stage is the installation phase. During this process, all parameters necessary for further communication are initialized. It is worth mentioning that target devices are exempt from this obligation, because the data is generated before the transmission begins by another device (with the appropriate computing resources). Thanks to this, using, e.g., OTA technology, the target data is saved in non-volatile flash memory. Two parameters are needed: a common hash and an arbitrary function (linear, quadratic, polynomial). The above parameters are generated using a pair of amicable numbers.

Its task is to send a message to one of the friendly devices, the serial number of which is BQWFxxxx4208. Assume that the transmitting device is uniquely identified with the serial number JKFxxxx00016. Suppose a pair of devices is associated with a friend pair 220 and 284 – for example, this is the first, smallest pair. Therefore, the data organization in the first (transmitting) node will be shown in Fig. 2.



Dev_{id_1}Numb	Amicable pairs			
JKFxxx00016	**Friend device id**	**My value**	**Friend value**	**Common hash**
	$\mathrm{Dev}_{id\_2}\mathrm{Numb}_{d\_2}$ BQWFxxxx4208	$\mathrm{Dev}_{id\_1}a_1$ 220	$\mathrm{Dev}_{id\_2}a_1$ 284	Hash_1 03ec4a2a178f84ec9e950e8edc668ee8 758eb60aa84e114e4c92bb8088c789bb

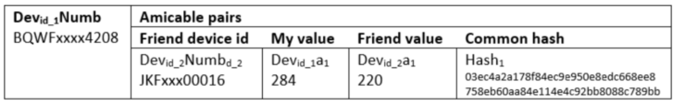
Fig. 2. Representation of friendship data in Device number 1

The basic structure describing a friendship contains the serial number/address of the second device and a pair of amicable numbers – the first one is for himself, and the second one is for the friend's device BQWFxxxx4208. The next parameter is the common hash for a given pair. Suppose there was no prior communication. Therefore, this variable is required to be initialized. The value is initially empty, so before starting the transmission, it is obtained by concatenating the serial numbers of devices and their amicable numbers. For each side, in the beginning, it will be a value:

(19) $$\text{common hash} = (\mathrm{Dev}_{\mathrm{id\_1}}\ \&\&\ \mathrm{min}_{a1}\ \&\&\ \mathrm{Dev}_{\mathrm{id\_2}}\ \&\&\ \mathrm{max}_{a2}).$$

This parameter is common to both parties.

In turn, the second device will have the data organization shown in Fig. 3.



$Dev_{id\_1}Numb$	Amicable pairs			
BQWFxxxx4208	**Friend device id**	**My value**	**Friend value**	**Common hash**
	$Dev_{id\_2}Numb_{d\_2}$ JKFxxx00016	$Dev_{id\_1}a_1$ 284	$Dev_{id\_2}a_1$ 220	$Hash_1$ 03ec4a2a178f84ec9e950e8edc668ee8 758eb60aa84e114e4c92bb8088c789bb

Fig. 3. Representation of friendship data in Device number 2

At this stage, a common hash value, one of the required parameters for future communication, is already established. The next step involves determining the binding function f(x), whose form is derived based on a pair of amicable numbers. Regarding the proposed scenario, the first proposition in the initial stage involves determining the multiplicity of the divisors of each number in the pair. For example, amicable numbers 220 and 284 will be used, whose divisors are listed in (1) and (3) formulas in the previous chapter, respectively. There are fewer divisors in the number 284 – five of them. Therefore, the degree of the generated function can be up to the fourth degree (one less than the number of elements). In the proposal, the target function does not have to be of the highest degree – it can be dynamically determined in the initial phase of making friends, and it is the generating mechanism that chooses the number of divisors (and their signs), determining the degree of the function. It can be either a linear function, a quadratic, or an n-degree polynomial. For example, suppose that from the group of divisors selected first and fourth divisors are (1, 71). In the next step, we choose the divisors of the second number that are related to the same index; it is 1 and 5. In this way, the obtained points – coordinates allow to creation of an equation of functions $f(x)$. Selected coordinates are as follows: (1, 1), (5, 71). On this basis, will be determined its equation $f(x) = ax + b$,

(20) $$1 = a + b,$$

(21) $$71 = 5a + b.$$

After calculations, the result is $f(x) = 17.5x - 16.5$. The appropriate graph is presented in Fig. 4.

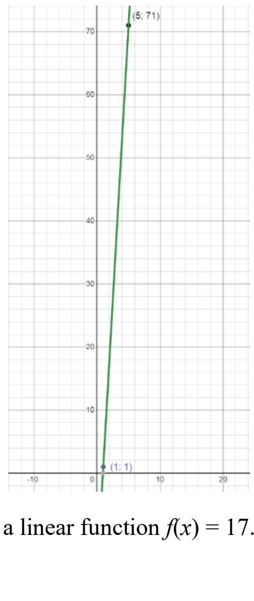


Fig. 4. Graph of a linear function $f(x) = 17.5x - 16.5$

The function is known to friendly devices, and there is an additional obligation to compute f(x) for any transmitted packet. If the value of the challenge is, e.g., $x = 8$, then the expected answer should be a value:

(22) $$f(8) = 17.5x - 16.5,$$

that is $f(8) = 123.5$.

The divisor will be chosen as before in the next scenario, which will be common to the quadratic function and any higher-order polynomials. A divisor of the second number for its index will also be selected. Then we choose one number from each pair and determine its sign. Accordingly, suppose the selection of two divisors whose indices are 3 and 4. Getting divisors from selected indexes creates pairs (4, 4) and (5, 71). On this basis, let us assume that the points – ultimately the zeros of the generated function will be 4 and –71. As a result of such assumptions, the quadratic function will be of the form:

(23) $$f(x) = (x - 4)(x + 71).$$

The next transformations allow us to obtain a general form:

(24) $$f(x) = x^2 + 67x - 284 .$$

Graph of the function (27), i.e., the parabola, is shown in Fig. 5a.

As a result of the query for $x = 8$, the other party should seamlessly compute $f(x)$, resulting in the value 316. In other words, the other party, having the function and challenge value, will prove that it can find a point belonging to the graph of the function, i.e., compute the value of $f(x)$ for the sent x. A fragment of the graphical representation of the calculation result is shown in Fig. 5b.

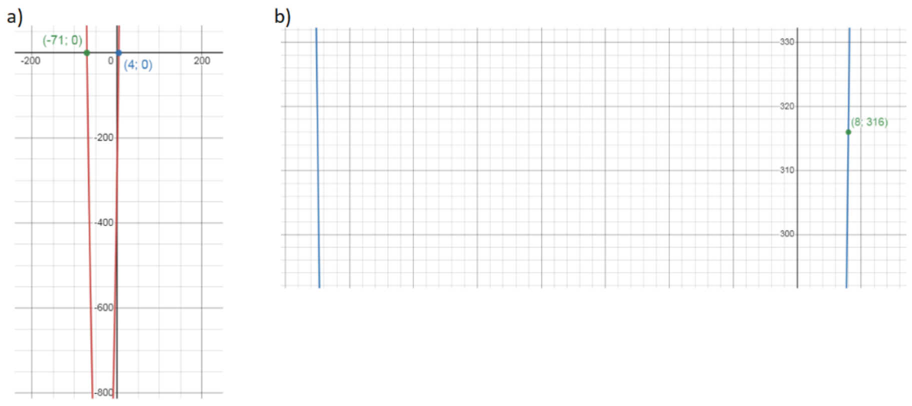


Fig. 5. Graph of the function $f(x) = x^2+67x - 284$ (a), and the computed value of the function $f(x)$ for $f(8)$ (b)

Notably, both the number of selected points and the signs (positive and negative) influence its form. A minor change modifies the function, making it impossible to confirm friendship. What function will be in force is established before the start of communication. At the moment of associating amicable numbers with devices, any function is generated, which in turn is created based on any/dynamically selected number of their divisors. If more than one point is selected (e.g., as in the example of zeros), then the function will traverse all of them and be a polynomial of n degree. However, the scheme of the procedure will be kept and common to each of them.

Each number used as a challenge may be added to the set of already used numbers, which is the same as treating this approach as a set of one-time passwords. It can also be assumed that the generated number must be greater than the previous one (equivalent to a timestamp). Nothing can prevent the addition of floating-point numbers with a certain precision (number of decimal places) as parameters to extend the set of challenges. Presented diagrams and guidelines constitute a general solution that should be adapted to the already specific requirements for the needs of the implementation being prepared and the defined functionalities.

5. Implementation and performance results

To confirm the applicability and feasibility of the proposed solution, an implementation was developed. The code was written in C and executed on target IoT devices, specifically, popular and widely used ESP32-series processors. In this case, the ESP32-C6 variant was used, which features a hardware module for hash value computation (SHA). Accordingly, two versions of the code were prepared. In the first version, the hardware-accelerated hash calculation module was utilized to optimize computation time. In the second version, the hash value was computed purely in software, within the program code. The results of this evaluation are presented later in this chapter.

As described in the previous chapter, during the device installation phase, all necessary data, including the shared hash and the function $f(x)$, are made available to the device upon startup. This means that the target devices are, among other things, relieved from the need to perform additional computations that could reduce battery life. When transmitting data, the sender uses this preloaded information to generate a friendship certificate. To do so, it selects a parameter x and computes $f(x)$ using the provided function. The resulting value is then concatenated with the shared hash and passed to the SHA (Shared Hashing Algorithm). For example, using the function defined in Equation (22), the result for $x = 8$ is $f(8) = 123.5$. This computed value is then combined with the shared hash value stored in the device's memory. Based on the data presented in the previous chapter, the sender will generate the following value (for SHA-128):

(25) $$\text{friend\_proof} = (123.500000 \,\&\&\, 1cdb261cb397cec1dc3cf467e083a2b5) = \\ = \mathrm{d}a1a90ba2ece3ed92b476aa49d3ddcd5,$$

or for SHA-256:

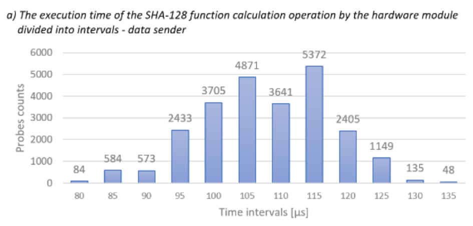
(26) $$\text{friend\_proof} = (123.500000 \,\&\&\, 1cdb261cb397cec1dc3cf467e083a2b5) = \\ = 47756ba9b9c845d01c9383fc49b5f4cae2c376792482b664be3e3c05ea4981e8.$$

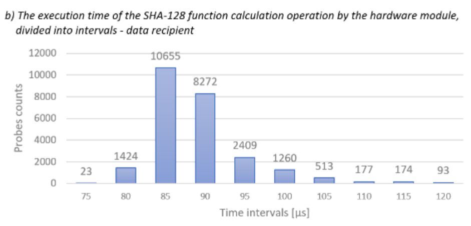
The tests involved the transmission of 25,000 messages. Using the hardware optimization module, the average time required to generate data for SHA-128 was 105 μs. On the recipient's side, the average processing time was 86 μs. When using the SHA-256 function, the time for a single iteration was 122 μs for the sender and 99 μs for the recipient. Table 1 presents a detailed breakdown of the results, grouped by time intervals.

Table 1. Results of measurements using the hardware module

Intervals (µs)	Sender (SHA-128)	Recipient (SHA-128)	Sender (SHA-256)	Recipient (SHA-256)
0 - 45	0	0	0	0
55 - 59	0	0	0	0
60 - 64	0	1	0	0
65 - 69	1	0	0	0
70 - 74	3	22	0	1
75 - 79	80	1424	0	4
80 - 84	584	10655	9	94
85 - 89	573	8272	266	3303
90 - 94	2433	2409	315	8827
95 - 99	3705	1260	120	3940
100 - 104	4871	513	350	3096
105 - 109	3641	177	3586	2392
110 - 114	5372	174	3225	1551
115 - 119	2405	68	2020	592
120 - 124	1149	24	3005	593
125 - 129	135	1	5533	405
130 - 134	32	0	3863	163
135 - 139	2	0	1686	28
140 - 144	0	0	569	6
145 - 149	2	0	295	3
150 - 154	2	0	144	1
155 - 159	0	0	5	0
160 - 164	1	0	0	0
165 - 169	2	0	2	0
170 - 174	1	0	1	0
175 - 179	4	0	2	0
180 - 184	0	0	2	0
185 - 189	0	0	0	0
190 - 194	1	0	0	0
195 - 199	0	0	0	0
More than 200	1	0	2	1
SUM	25,000	25,000	25,000	25,000

The results for SHA-128 are presented in graphical form in Fig. 6. Variant (a) corresponds to the sender, while variant (b) represents the recipient.

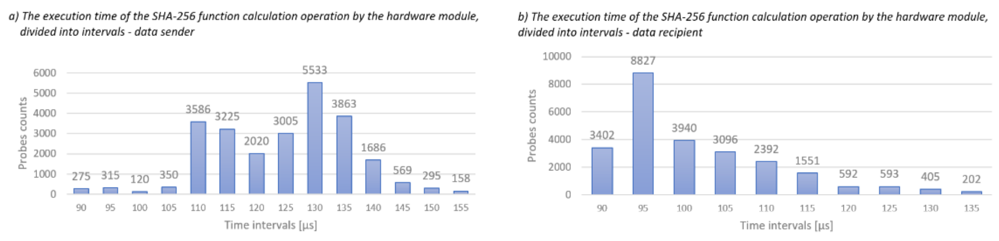




(a) (b)

Fig. 6. Execution time of SHA-128 by the hardware module by sender (a) and recipient (b)

The results for SHA-256 are presented in graphical form in Fig. 7. Variant (a) corresponds to the sender, while variant (b) represents the recipient.



(a) (b)

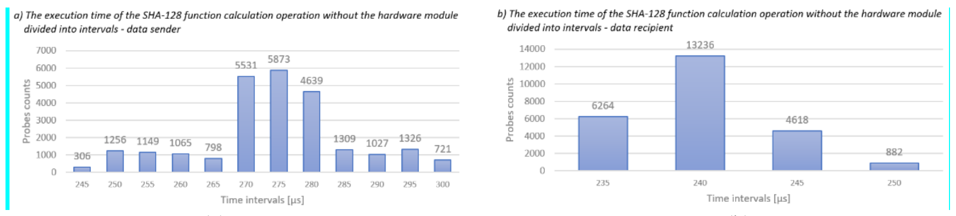
Fig. 7. Execution time of SHA-128 by the hardware module by sender (a) and recipient (b)

In the case of SHA-128, within less than 100 µs, a total of 3674 requests (14.696%) were processed on the sender side and 22,783 requests (91.132%) on the recipient side. In the 100-110 µs interval, the sender processed 8576 requests (34.304%) and the recipient 1773 requests (7.092%). In the next interval, from 110 to 120 µs, the number of operations was 9013 for the sender (36.052%) and 351 for the recipient (1.404%). For times above 120 µs, 3737 operations were recorded on the sender side (14.948%) and 93 on the recipient side (0.372%).

In the case of SHA-256, within less than 100 µs, 590 requests (2.36%) were processed on the sender side and 12,229 requests (48.916%) on the recipient side. In the 100-110 µs interval, the sender executed 470 operations (1.88%) and the recipient 7036 (28.144%). Between 110 and 120 µs, 6811 requests (27.244%) were processed by the sender, and 3943 (15.772%) by the recipient. In the 120-130 µs range, the sender handled 5025 operations (20.1%), while the recipient processed 1185 (4.74%). During the 130-140 µs interval, the number of operations performed was 9396 (37.584%) on the sender side and 568 (2.272%) on the recipient side. For times above 140 µs, the sender processed 2708 operations (10.832%), and the recipient 39 (0.156%).

For 25,000 iterations, the average execution time of the complete instruction set for SHA-128 was 105 µs on the transmitting device and 86 µs on the receiving device. In the case of SHA-256, the average time for a single operation was 122 µs for the sender and 99 µs for the receiver.

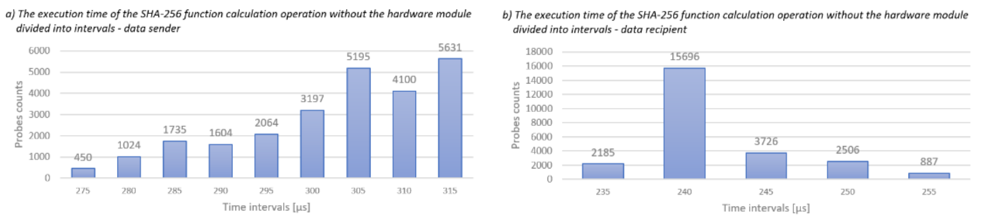
The remainder of this chapter presents the results obtained under the same test scenario, but with all operations performed without the use of the hardware hash calculation module. Without the use of the hardware optimization module, the average time required to generate data for SHA-128 was 272 µs. On the recipient's side, the average processing time was 238 µs. The results without the use of the hardware hash calculation module are shown in Fig. 8. Variant (a) on the left represents the sender, while variant (b) corresponds to the recipient.



(a) (b)

Fig. 8. Execution time of SHA-128 without the hardware module by sender (a) and recipient (b)

The results for SHA-256, based on the same test conditions, are shown in Fig. 9.



(a) (b)

Fig. 9. Execution time of SHA-256 without the hardware module by sender (a) and recipient (b)

Replacing the hash function with SHA-256 does not result in a significant increase in the average execution time. The values are 276 µs for the sender and 239 µs for the recipient, respectively.

The use of a hardware module has a significant impact on the execution time. Generating a friendship certificate using SHA-128 takes an average of 105 µs when the hardware module is utilized, whereas disabling it increases the average time to 272 µs. In the case of SHA-256, the average execution time is 122 µs with the hardware module and 276 µs without it. In both cases, this represents an increase of more than 100%. It is worth noting, however, that even without the hardware module, the execution times for a single iteration remain fully acceptable. The system response time is still short enough that the user will not perceive any delay when receiving a response. With relatively simple resources, a secure data transmission mechanism has been implemented, where the level of security corresponds to the strength of the chosen hash function, either 128 or 256 bits, depending on the use case and requirements.

Due to the limited resources of the target devices, it is worth highlighting an important benefit. Although the proposed solution does not directly address the process of changing the set of friendly numbers (including the shared hash and the $f(x)$ function), this can be achieved using OTA (Over-The-Air) technology. In such a case, the necessary data can be stored in non-volatile flash memory, thereby preserving RAM and further optimizing resource usage.

6. Conclusion

The paper proposes a solution to solve the significant problem of authorization and authentication of communication without involving known, popular, and currently used cryptographic means for this purpose. Resignation from them results from the nature of the devices it is dedicated to. We are talking about IoT devices, for which the computing power is limited, and the implementation in these classic systems, in the original form of solutions, is ineffective.

The proposed mechanism, which will remove from devices the need to implement demanding methods of cryptographic algorithms, is related to amicable numbers. Their feature is that the summary of the proper divisors of one number is equal to the other (it works both ways). Using this feature, "amicable" devices can also be paired successfully. Such an association based on numbers is an expression of the friendship of devices – that is, the possibility of their mutual communication. In this way, devices that are "friends" can perform legitimate operations because they have a pair of amicable numbers in common – something that connects them and that each of them can verify in a fast and efficient manner. Another issue is the implementation of the "proving friendship" process. This paper describes two proposals in this regard. The first one uses a one-way, appropriately parameterized hash function, while the second case is related to a zero-knowledge approach and the proposed challenge-answer protocol using any function $f(x)$.

The proposed technique has minimal hardware requirements, yet it enables a significant increase in security at a low cost – even for the smallest devices with very limited RAM and ROM. Additionally, the solution provides both anonymity and uniqueness in data transmission, which effectively hinders potential intrusion attempts and enhances overall system resilience. The proposed approach enables a notable increase in the security level of data transmission at minimal computational and hardware cost. It offers an effective balance between cryptographic strength, performance, and compatibility with limited-resource embedded platforms, making it a compelling option for modern IoT applications.

Searching for alternative solutions to traditional methods in the context of the Internet of Things is a challenge that must be met. It is a world only seemingly identical to the Internet of Humans and classical solutions known from "big" computers. Hardware, software, and transmission limitations mean that traditional data storage and transmission approaches fail. At the same time, technological progress does not disappear from these limitations. On the contrary, the miniaturization of devices makes these limitations even more stringent, and there is a common tendency to reduce the size and computing power, memory, and transmission bandwidth, mainly to save energy and radio bandwidth. Basic laws of physics and mathematics cannot be bypassed. Instead, they should be used to find alternatives and propose new technologies, as discussed in the text.

References

1. A s h t o n, K. That "Internet of Things" Thing: In the Real World Things Matter More than Ideas. – RFID Journal, 2009.
2. S o r r i, K., N. M u s t a f e e, M. S e p p ä n e n. Revisiting IoT Definitions: A Framework towards Comprehensive Use. – Technological Forecasting and Social Change, Vol. **179**, 06.2022, 121623.
3. H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, Eds. Vision and Challenges for Realising the Internet of Things. European Commission, Information Society and Media, Brussels, 2010.
4. L i u, R., J. W a n g. Internet of Things: Application and Prospect. – MATEC Web of Conferences, 2017, 100. 02034. DOI: 10.1051/matecconf/201710002034.
5. R a h o o f, P., L. N a i r, V. I j y a s. Trust Structure in Public Key Infrastructures. 2017, pp. 223-227. DOI: 10.1109/Anti-Cybercrime.2017.7905295.
6. V o l l a l a, S., N. R a m a s u b r a m a n i a n, U. T i w a r i. Public Key Cryptography. 2021. DOI: 10.1007/978-3-030-74524-0_2.
7. C a n t e a u t, A. Stream Cipher. 2011. DOI: 10.1007/0-387-23483-7_412.
8. S i n g h, M., R., K h a n d a k a r. Digital Signatures. 2021. DOI: 10.1007/978-3-030-60890-3_14.

9. Z h a o, Z. An Efficient Anonymous Authentication Scheme for Wireless Body Area Networks Using Elliptic Curve Cryptosystem. – Journal of Medical Systems, Vol. **38**, 2014, No 13. DOI: 10.1007/s10916-014-0013-5.
10. K a h n, D. The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. Scribner, 1996. ISBN 0684831309.
11. B a r n e s, J. Amicable Numbers. 2016. DOI: 10.1007/978-3-319-46831-0_2.
12. P o m e r a n c e, C. On Amicable Numbers. 2015. DOI: 10.1007/978-3-319-22240-0_19.
13. S a d i k u, M., C. A k u j u o b i. Fundamentals of Computer Networks. 2022. DOI: 10.1007/978-3-031-09417-0.
14. C h a h a r, N. Computer Network Security. – International Journal of Innovative Research in Science, Engineering and Technology, Vol. **7**, 2022. 1031. DOI: 10.5281/zenodo.6448170.
15. L i, Y., S. S c h ä g e, Z. Y a n g, F. K o h l a r, J. S c h w e n k. On the Security of the Pre-Shared Key Ciphersuites of TLS. 2014. DOI: 10.1007/978-3-642-54631-0_38.
16. H w a n g, M i n-S h i a n g, et al. An Access Control Based on a Scheme Chinese Theorem Remainder and Time Stamp Concept. 2003.
17. P a r k i n s o n, S., S. K h a n a. Identifying High-Risk Over-Entitlement in Access Control Policies Using Fuzzy Logic. – Cybersecurity, Vol. **5**, 2022, No 6. DOI: 10.1186/s42400-022-00112-1.
18. D e s m e d t, Y., A. S h a g h a g h i. Function-Based Access Control (FBAC): From Access Control Matrix to Access Control Tensor. 2016, pp. 89-92.
19. P e n e l o v a, M. Access Control Models. – Cybernetics and Information Technologies. Vol. **21**, 2021, No 4, pp. 77-104.
20. D e n i s, T., S. J o h n s o n. Hash Functions. (2006). DOI: 10.1016/B978-159749104-4/50008-X.
21. B o g d a n o v, A. A Gentle Introduction to Zero-Knowledge. 2022.
DOI: 10.1007/978-3-031-04206-5_1.
22. R a s t o g i, N., P. A v i n a v, R. S h w e t a. Enhanced Authentication Scheme Using Password Integrated Challenge Response Protocol. – International Journal of Computer Applications, Vol. **62**, 2013, pp. 15-19. DOI: 10.5120/10107-4764.
23. A l-M a l i k i, O., H. A l-A s s a m. Challenge-Response Mutual Authentication Protocol for EMV Contactless Cards. – Computers & Security, Vol. **103**, 2021, 102186. DOI: 10.1016/j.cose.2021.102186.

Received: 07.04.2025. Revised version: 09.06.2025. Accepted: 17.06.2025