Migration of the perfect cipher to the current computing environment

PETR VOBORNÍK
Department of Informatics
University of Hradec Králové
Rokitanského 62, Hradec Králové, 500 03
Czech Republic
petr.vobornik@uhk.cz www.petrvobornik.cz

Abstract: - Algorithm of the perfect cipher has been known since the beginning of the last century. However, this procedure requires respecting of strict conditions that complicated the practical use of the cipher. Original conditions of Vernam perfect cipher will be discussed at first in detail in this article. The original approach of characters will then be transferred to the bit level. The modification of work with the encryption keys with the use of modern hash algorithms will be proposed so that the major factors that prevent the practical use of the cipher will be removed and most of their benefits will be preserved. The proposed new encryption algorithm will be subjected to the measurement of speed in conclusion, and these results will be compared with other known encryption algorithms.

Key-Words: - Perfect cipher, encryption, communication, Vernam, one-time pad, hash function.

1 Introduction

The principle of the perfect cipher has been known for almost a century. Unfortunately conditions that must be complied with its use still complicate its practical use. The cipher is suitable in the case that extremely high confidentiality is required and when extremely high costs associated with the production and distribution of keys do not mind. [1]

In this article the principle of perfect cipher will be summarized and on this base, with use of modern hash algorithms, will be proposed one of possible modifications of work with keys so that the major factors that prevent the practical use of the cipher will be removed.

1.1 Basic terms

The following terms are commonly known but they may have different meanings in different contexts. Therefore will be defined what is under their designation meant here. Other terms will be explained in the article.

- **Data** – the source data that are to be protected by encryption against reading and misuse by third parties.t
- **Message** – is the set of information sent between two communicating parties. It consists of encrypted **data** and other information such as sender ID, destination recipient, date and time sent, etc.
- **Key** – the secret set of data which can encrypt and then decrypt the source **data** by the computing operation.
- **Password** – a word, phrase or combination of characters that can decrypt the encrypted data and vice versa. I.e. password is either directly used as the **key** or the key is generated on its basis. It will be the case of this article.

2 Perfect cipher

American Gilbert Sandford Vernam (1890-1960) constructed an interesting cryptographic system in 1917. It was based on Vigenèr's cipher from 1586 [2], but it included several major changes, supposedly inspired by the German cryptologist Hermann from 1892 [3]. The system was based on a randomly generated one-time password that is the same length as the message.

Claude Elwood Shannon (1916-2001) mathematically proved that the Vernam cipher (also called "one-time pad", also see [4] or [5]) is absolutely safe (unbreakable) encryption system in 1949 [6]. The perfect secrecy of the cipher can be achieved only when the three strict conditions of reliability are observed:

1. The key must be perfectly random.
2. The key must be as long as the encrypted message.
3. The key must not be used repeatedly. [7]

When the conditions of confidence are complied, the cipher is completely safe against any attempt to break, including a brute force attack. If the correct key is not known, there is no way to decipher the message not even in an arbitrarily long period of time. It is possible to find a key that could convert encrypted data into readable text of the same length, but as more keys like that can be found, this data can in fact make arbitrary sense, and it is not possible to predict which of these interpretations was correct. [8]

Vernam proposed in his subsequent patent [9] a simple device that worked with 31 characters (26 letters, space, carriage return (CR) and line feed (LF) and signals "following numbers" and "following letters"). Needed is 5 bits, which the device encrypts by a random key using XOR operation.

2.1 XOR operation

The logical operation of exclusive disjunction (usually called "exclusive or", abbreviated XOR) is marked as follows: "⊕". Its result is 0 if the two input values are identical, and 1 if they are different (see Table 1).

$$A \oplus B = C$$

A	B	C
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Table 1 – Status table of values of the XOR operation

XOR is a commutative operation, therefore does not depend on the order of individual values between which the XOR operation is performed.

$$(A \oplus B = C) \quad = \quad (B \oplus A = C)$$

Instead of the XOR operation can be also used function modulo 2 from sum of the two values.

$$(A \oplus B = C) \Leftrightarrow (A + B) \bmod 2 = C$$

From the result of the XOR operation (C) can be calculated one of input values (A or B) by using the additional XOR operation with the second input value (B or A).

$$(A \oplus B = C) \Leftrightarrow (C \oplus A = B) \Leftrightarrow (C \oplus B = A)$$

Thus, if the i-th bit of data D_i is to encrypt by XOR operation with i-th bit of key K_i to the encrypted string of C_i, then from C_i can be the input bit of data D_i deciphered again by performing XOR operation with the i-th bit of key K_i.

Encrypt: $D_i \oplus K_i = C_i$

Decrypt: $C_i \oplus K_i = D_i$

If the key is random, then the probability that $K_i = 0$, is the same as the probability that $K_i = 1$ and it is equal to ½, i.e. 2^{-1} or 50%. The same is true even for encrypted values C_i. Unless the known key K_i, the probability of "guessing" the correct values is exactly half. Under these circumstances the probability of determining the correct character for whole symbol with 8 bits (1 byte) is only 2^{-8}, i.e. 1/256 or less than 0.4%.

2.2 Principle of the Vernam cipher

The original version of the cipher worked with only the base English alphabet of 26 letters. Numbers 0 through 25 are assigned to these letters (A through Z) and for the i-th character of classified data D_i by the key K_i (key is also composed only of the characters of the alphabet) the character of encrypted data C_i is determined as follows [2]:

$$C_i = (D_i + K_i) \bmod 26$$

Decryption is performed by the inverse operation:

$$D_i = (26 + C_i - K_i) \bmod 26$$

For example, the word "AGE" would be encrypted with the key "UHK" as shown in Table 2.

	Characters			Numbers		
Data	A	G	E	0	6	4
Key	U	H	K	20	7	10
Encrypted data	U	N	O	20	13	14
Decrypted data	A	G	E	0	6	4

Table 2 – Example of encryption of characters by the original Vernam cipher

An improved version of this cipher works with binary data representation. The individual bits of data in the binary form are encrypted by the XOR operations with individual bits of the key. The advantage was the ability to machine processing of the cipher. In his time Vernam used an own conversion table for the basic characters of the alphabet in binary form, instead of the values 0 and 1 marked the characters + and –. [9] At present, when the data are stored and transmitted in electronic form

of bits by default, the situation for this style of encrypt is much easier.

The same data as in the previous case but encrypted in binary encoding (for conversion to bits is used the standard ASCII character table) looks as is shown Table 3.

	Characters			Bits		
Data	A	G	E	01000001	01000111	01000101
Key	U	H	K	01010101	01001000	01001011
Encrypted	20	15	14	00010100	00001111	00001110
Decrypted	A	G	E	01000001	01000111	01000101

Table 3 – Example of encryption of data by the Vernam cipher on the bit level

In the example the encrypted data are listed only as an index character in the ASCII table, because they are in text format undisplayable. In this case, that the random key not contains only letters but bytes of characters chosen from the entire ASCII table of 256 characters. Respectively, the key generator should operate at the bit level (randomly chosen sequence of 0 and 1), and don't take into account the resulting characters.

2.3 Consequences of breach of conditions of reliability

Breach of conditions of reliability leads to a lack of cipher's safety and allows to break it. Specifically, failure to comply of each individual conditions has the following consequences.

If **the key is used repeatedly**, this key can be easily determined from the knowledge of only two intercepted messages encrypted by the same key. The following relationship applies.

$$D1_i \oplus K_i = C1_i$$

$$D2_i \oplus K_i = C2_i$$

$$C1_i \oplus C2_i = D1_i \oplus D2_i$$

Where $D1_i$ is i-th character of of 1st data, $D2_i$ is i-th character of the 2nd data, K_i is i-th character of the key, $C1_i$ is i-th character of the 1st encrypted message and $C2_i$ is i-th character of the 2st encrypted message.

The result of the XOR operation of two encrypted data is XOR of two original data. This will remove all the randomness of the key, and both the original data thus the key can be calculated by a simple statistical cryptanalysis of the result. [10]

$$K_i = C1_i \oplus D1_i = C2_i \oplus D2_i$$

This allows each following messages (encrypted with the same key) decrypt in real time without need of any cryptanalysis.

Each key is therefore necessary safely completely to "destroy" on both of sides (sender and recipient) immediately after the first use.

If the **key has a shorter length than the transmitted message**, it must be used repeatedly for encryption of the part of data that it is not covered. This should result in the same effect as the repeated use of the key. If an attacker knows some part of the data, he can obtain a part or even the whole key by performing XOR operation, and he can use it on the rest of the data, which he doesn't know.

Fact that an attacker knows the part of the data is fairly common. Texts as "Hello", "How are you?", etc. are usually mentioned at the beginning of letters, the signature of the sender is then at the end. The situation is even easier when binary data are send, because most of files has a header that is always the same (JPEG, ZIP, WAV, DOC...) or least from a finite set of possibilities. The structure of word processing documents (RTF, XML, HTML...) then repeatedly contains a known sequences of characters that can be easily detected by frequency analysis.

Perfect randomness of the key as well as its sufficient length guarantees that every single character (bit) of encrypted data is completely independent of the other characters (bits). Knowledge of any part of the data thus to the attacker divulge nothing about any other unknown part of data or key.

Pseudo-random values cannot be used for perfect security of encryption. It is generated by a certain algorithm and this process is reproducible when same conditions are complied. Data are then deciphered in finite time, respectively it is possible find such a key, which converts the encrypted message to the understandable data and at the same time it is possible to demonstrate the relationship between the individual parts or to a default value (seed) and thus identify which of the possible decrypted versions of the message is the right one. To generate of the key is best to use physical methods, such as radioactivity, of which it is shown that its character is truly random. [1]

2.4 The long and random key

Conditions of reliability to guarantee the unbreakability of the cipher, but they also complicate its use. Specifically, the requirement of perfect randomness of the key, which makes difficult its

automatic generation. Need for length of the key that must be equals to the length of the encrypted data brings (if we ignore quantum cryptography) the same problem as the transfer of data themselves, to which also contributes one-time usability of each key. It may therefore be desirable, even at the expense of absolute unbreakability of the cipher, to allow the key (respectively password) can be shorter, not completely random (memorable) and reusable.

As already mentioned, the key composed of pseudo-random values does not prevent deciphering the encrypted data in a finite time. However, if the key will be "well random" and also comply with the other conditions of reliability, possibility of using computational and statistical cryptanalysis is excluded, excepted brute force attack. For example the attacker can use it to determine the key that makes sense to the encrypted data (or part of it) and then search for correlations between different parts of the key. A more effective means of brute force attack would have been (if an attacker knows used pseudo-random algorithm for generate of values for the key), to determine the default value of generator - the seed, respectively the password. If it is well selected, the data can be deciphered only by "guess the password" by using "brute force" and this final time of deciphering of data may be unrealistically long.

If the above mentioned advantages of password (short, non-random and repeatable) will beneficial for the encryption while the referred limitation of perfection of the encryption (data will be deciphered when the password will "guessing") are not insurmountable obstacles, then an algorithm is possible to created. This algorithm should repeatedly create from a short passwords the arbitrarily long key, which will be statistically verifiable random and from the uniform distribution. This means that all values in the range of byte[1] were generated with the same probability, respectively the created bit sequence was completely random.

The described properties directly fit into the definition of hash function. If the hash is used appropriately then it is possible to use it for achieve of all desired properties of the keys.

3 Modified interpretation the perfect cipher

Individual terms of reliability will be solved gradually through the modern practices and technologies. [11]

3.1 The key must be perfectly random

To the work more effectively with Vernam cipher a function was used that was not available when the cipher was created and that is the hash. The hash is one-way (irreversible) computationally efficient function mapping binary strings of arbitrary length to strings of fixed length, it is called the hash-value.

The basic idea is that the hash-value serves as a compact representative of the input string. For cryptographic applications the hash function H is chosen that it is computationally impossible to find two different inputs, which have the hash-value identical (i.e. to find the X and Y such that it applies $H(X) = H(Y) \wedge X \neq Y$), and is also computationally impossible to determine the input X for the hash-value of Y (i.e. $H(X) = Y$). The probability that the n-bit hash-value (e.g. $n = 128$ or 160) of a random chain will have a specific n-bit hash-value is thus equal 2^{-n}. [10]

The statistical tests of randomness of hash code generated by the algorithm SHA-1[2] according to [12] and [13] demonstrated that the generated bit sequence meets from a statistical point of view the conditions of random uniform distribution. Other hashing algorithms (such as MD5, SHA-256, SHA-512, etc.) should by definition have the same characteristics, which can be checked according to the procedures specified in [10] using the software which is described in [12].

3.2 The key must be as long as the encrypted message

The key which satisfies the condition of randomness and it does not allow the re-calculation of the password it can be created by using the hash algorithm on any password. Its length however is predetermined to constant number of bits according to a specific algorithm applied. Nevertheless a key much longer than the hash code is needed.

The key of needed length can be created by using a multilevel hash. Its calculation is performed so that

[1] The range of the byte is 0–255, i.e. 256 (2^8) possible values.

[2] SHA-1 – Secure Hash Algorithm, returning hash code of 160 bits, which was designed by NIST for U.S. government applications. [8]

the hash of the original password is used to encrypt the first block of data while also serve as an input to generate a new hash code (hash level 2). It again encrypts the next block of data and a hash of the third level is generated from it as the key for the next block of data and so it continues until it covers the entire data message (see Fig. 1).

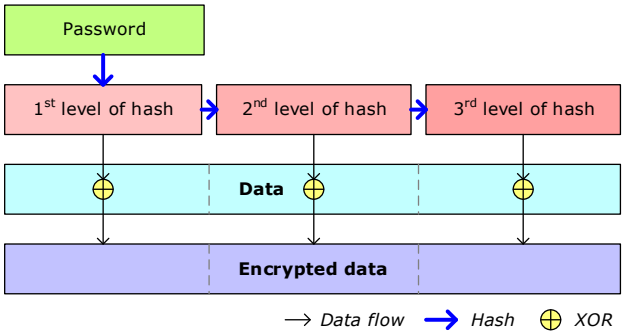

Fig. 1 – Diagram of data encryption by the XOR operations, where the key consists of a simple multi-level hash of the password

However this approach brings one major flaw. If an attacker knew some part of data, he would be able to decipher their other unknown parts. For example, when would he know the part of the data encrypted by second level of hash, he can then perform the XOR operation between these data and the encrypted data for acquiring a part of the key (hash level 2). He cannot calculate the first level of hash or password from it, but he can generate a third level of hash, then fourth level, etc. (see Fig. 2). Thanks to it, he is able to decrypt the data from the block whose content he knows (e.g., greeting or a document header) or he reveals by a dictionary attack.

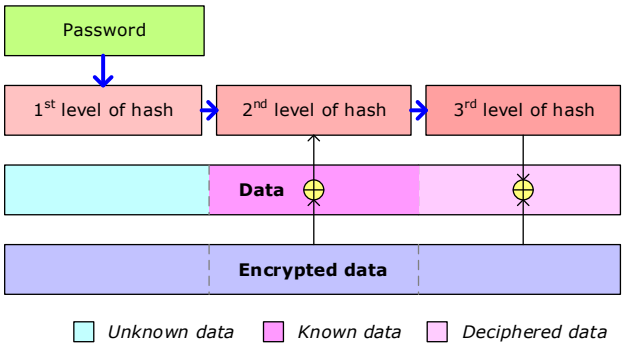

Fig. 2 – Diagram of deciphering the data encrypted by the key from a simple multi-level password hasht

The first level of hash was removed from the encoding process to eliminate this weakness. This block of the key cannot be determined nor with the knowledge of content whole message (data). The first level of hash is combined with each level of hashes by XOR operations (see Fig. 3).

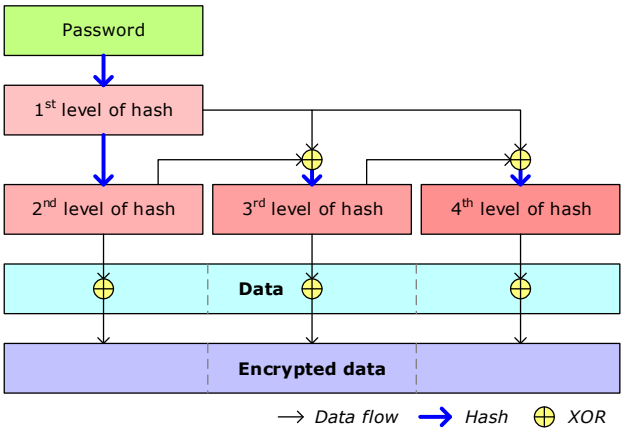

Fig. 3 – Diagram of data encryption using a key from the combined multi-level password hash

If the attacker knows a certain part of data, he can decipher the part of the key by which was encrypted this section, but he cannot determine the key for the next (and of course previous) block of data. He would need to know either the first level of password hash or the source of hash for the key of the next block of data. Both requirements would mean determination of the reverse hash, which is computationally impossible according to the basic definition of this function.

The only way to determine the unknown part of the data is "guess" the password or first level of hash. This mainly depends on the "strength" of chosen password, i.e. how it can withstand dictionary attack and brute force attacks. There are a number of techniques how to select easy-to-remember passwords (e.g. see [14]) which are resistant to these types of attacks (e.g. see [15]).

3.3 The key must not be used repeatedly

In order the key for data encryption was different every time, even if the password was still the same, we proceeded from the fact that for the complete change of the whole key (consisting of a multi-level hash) it is needed to change only a single bit of the password or first level of hash. In this type of change the description of this change can be a part of the message containing the encrypted data. For example this can be an additional text string that could be added to the password before calculating the first level of hash. This supplement of passwords is called "salt" and it is a good tool against dictionary attacks based on prepared dictionaries of hashes [16]. Its publication does not reduce the difficulty of deciphering the data because for the calculation of the key is still necessary to know the original part of the password.

With the addition of the salt the key for the data is completely different every time and if its part (or the whole key) in some previous data transmission is determined, the data transferred in the future has no reduced security, without requirement to change the password. It is only necessary to ensure that the salt is always different. This can be achieved for example using a generator of the GUID[3] values.

The salt can be transmitted as one parameter of the message in communication between two sides. But when the cipher is used for the long-term self-saved files the salt needs to be saved directly into the encrypted file, preferably right at the its beginning (see Fig. 4).

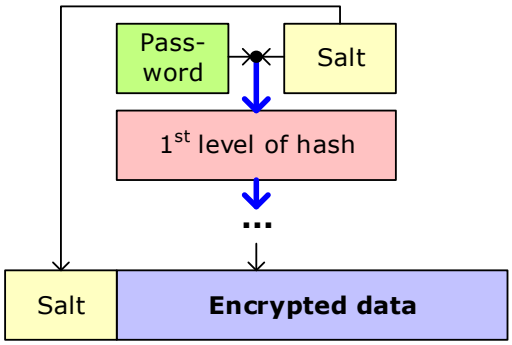

Fig. 4 – Diagram of data encryption using the salt

When the salt is saved at the beginning, the encryption and the decryption of the data can be proceed by the usual way as a block or a stream. Beginning of the data where the salt is stored must be read always and thus it is not possible to decrypt only certain sections of the data independently of the order. It also complicates the need to recalculate the appropriate level of the hash.

4 Speed of the encryption

The proposed cipher algorithm is built directly on a specific key, which consists of multi-level hash, which is also in each level again combined with the first level of the hash. Calculating of the hash is of course not trivial computational operations, but a complicated algorithm that takes some computing time. It is obvious that this hash algorithm has the greatest share of the speed of encryption. Since the cipher can work with any hash algorithm, we tried to choose the fastest of them.

For this purpose, the following comparisons were performed (see Table 4). In this test there were used different hash algorithms (table rows) which calculated keys of given lengths (table column) as multi-level hash. The experiment was repeated 3 times and each measured time required for the calculation of each key was recorded. Of these three repeated measurements there was always recorded the shortest time (table cell), expressed in milliseconds. Basic features of each hash algorithm (irreversibility and randomness) has been taken as matter of course and further no tested.

The measurement were performed under identical conditions, i.e. on the same computer with the same running processes. Parameters of the tested machine were following: CPU 2,2GHz Core i7-3632QM, RAM 8GB, OS Windows 8 Pro 64bit. To calculate of keys was used hash algorithms implemented in the library HashLib4 version 2.0.1 in programming environment Microsoft .NET Framework 4.5, language C#. Researched cipher algorithm was implemented and subsequently tested in the same environment. The program ran only on a single thread.

Data size [MB]	1	5	10	20	30	50	100
SHA-3-256	218	1 084	2 169	4 341	6 517	10 865	21 693
SHA-3-512	110	550	1 088	2 182	3 271	5 440	10 911
SHA-1	86	431	857	1 715	2 576	4 314	8 603
MD5	140	406	809	1 619	2 427	4 067	8 122
SHA-2-256	70	352	704	1 407	2 113	3 519	7 050
SHA-2-512	63	319	638	1 285	1 925	3 199	6 401

Table 4 – Comparison of computation speed [ms] of multi-level hash of the various algorithms

The comparison in Table 4 shows that the fastest of the test hash algorithms is SHA-2 with a length of 512 bits (also called as SHA-512). This algorithm was then subsequently used for comparison test of the speed of existing encryption algorithms (see Table 5). This comparison was made in a similar test under the same conditions as compared to the speed of hash algorithms. Again it was only the calculation of values within the computer's memory without the need loading or saving from/to the hard disk. The data were processed in stream in blocks of size 5 kB.

[3] GUID – Globally Unique IDentifier. Randomly generated value with negligibly small probability that anyone would ever have been generated by two identical values.

[4] HashLib project, see http://hashlib.codeplex.com

Data size [MB]	1	5	10	20	30	50	100
AES	26	130	259	527	774	1 289	2 611
DES	32	159	314	629	949	1 582	3 158
RC2	38	188	374	748	1 121	1 864	3 742
AES-Managed	43	219	431	865	1 292	2 161	4 309
TripleDES	61	301	601	1 202	1 802	3 008	6 028
Tested cipher	**95**	**472**	**945**	**1 891**	**2 848**	**4 740**	**9 458**

Table 5 – Comparison of speed encryption [ms] of datasets of the various sizes and by the different algorithms

Comparison shows in Table 5 and in the graph in Fig. 5 that the proposed cipher algorithm is slower than other algorithms, but about 67% of the time spans calculation of the key, even by the fastest hash algorithm SHA-512. Using a faster hash algorithm would lead to a significant acceleration of the cipher.

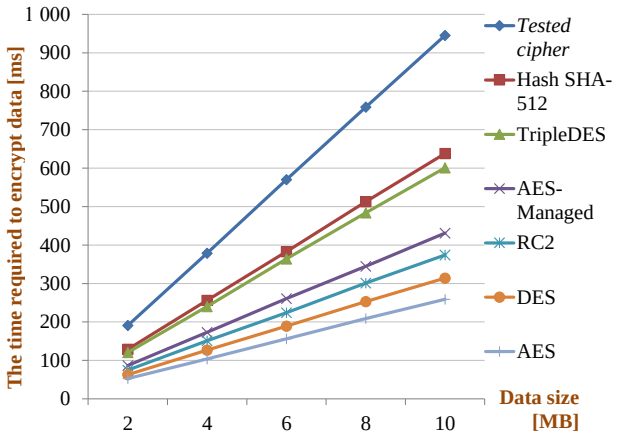

Fig. 5 – Comparison of speed encryption [ms] of datasets of the various sizes and by the different algorithms. Included is also the speed of generating the key using a hash algorithm SHA-512.

Other cipher algorithms had an advantage in that they have not been implemented in .NET controlled (managed) code, but they were built directly into the operating system. Here they are processed under the application layer through the Windows CryptoAPI without unnecessary overhead [17]. This difference is evident when comparing the speed of encryption by AES and AES-Managed, which in both cases is the same algorithm, but AES-Managed is implemented in .NET, as well as tested cipher. Other algorithms under the same conditions would be then also significantly slower, roughly in the same proportion (approximately 1.67x).

The speed of cipher is so limited for use in the classical application in real time such as on-line communication between two sides (or video transmission as in [18]), where the speed of the connection is the one of the main parameters. Its use could be thus more in cases where the level of security has a higher priority than the time required to encrypt (e.g. archiving of confidential files).

5 Conclusion

The original principle of Vernam perfect cipher was outlined in the article. On the basis of it was proposed modifications to work with the keys, which previously very complicated its practical use. When combined with modern hash algorithms is possible to encrypt data by using mathematically proven of reverse incalculable procedures and also can be repeatedly used a "simple" password. So the strength of ciphers is always directly proportional to the strength of the chosen password.

Implementation of this procedure is a programmatically very simple. Speed of encryption and decryption of data mostly dependent on the speed of calculation of a hash code, i.e. on the selected hash algorithm. Slightly faster than the selected SHA-512 has been doing so Shabal-512 a candidate for the SHA-35 [19], or in a multi-threaded processing the winner of this competition Keccak [20].

The proposed cipher can be used due to its current lower speed to protect the transmission of data over a public network internet only in cases where does not matter a deceleration needed for data encryption (also see [21]). For encryption of archives and files for long-term storage, where is usually more important their safety than the time required to encrypt, this cipher can be used just now.

Acknowledgements:
This article is supported by the project Research of cipher algorithms, maintained on the Faculty of Science on the University of Hradec Králové.

References:

[1] Singh, S., *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, New York: Anchor Books, reprint edition, 2000. ISBN 978-0385495325.

[2] Piper, F., Murphy, S., *Cryptography: A Very Short Introduction*, Oxford: Oxford University Press, 2002, ISBN 978-0192803153.

[3] Janeček, J., *Gentlemani nečtou cizí dopisy*, Brno: Books, 1998, ISBN 80-85914-90-5.

[5] The SHA-3 Cryptographic Hash Algorithm Competition, see http://csrc.nist.gov/groups/ST/hash/sha-3/

[4] Panagiotis, M., Lambrinoudakis, K., Gritzalis, S., Antonidakis, E., *Digital design of a Cryptographic card (LAM) embedded Smart Card Reader*, Proceedings of the 11th WSEAS International Conference on Computers, Crete Island: WSEAS Press, Agios N., 2007, pp. 571-576, ISBN 978-960-8457-92-8.

[5] Poriazis, S., *The Modulo-10 Partition Counter*, Proceedings of the 10th WSEAS international conference on Circuits, Athens: WSEAS Press, 2006, pp. 41-44, ISSN 1790-5117, ISBN 960-8457-47-5.

[6] Shannon, C. E., *Communication Theory of Secrecy Systems*, Bell System Technical Journal, 1949, 28, pp. 656–715.

[7] Hála, V., *Kvantová kryptografie*, Aldebaran Bulletin, 14/2005, vol. 4, ISSN 1214-1674, [Online], Available: http://aldebaran.cz/bulletin/2005_14_kry.php [12 Feb 2014].

[8] Voborník, P., *Téměř dokonalá šifra*, Matematika, fyzika, informatika, 2014, vol. 23, no. 1, pp. 54–69. ISSN 1210-1761.

[9] Vernam, G. S., *Secret signaling system*. U.S. Patent 1310719, 07/22/1919.

[10] Menezes, A., J., Oorschot van, P., C., Vanstone, S., A., *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1996, ISBN 978-0-8493-8523-0.

[11] Voborník, P., *Modification of the perfect cipher for practical use*, Advances in Mathematical Models and Production Systems in Engineering, Proceedings of the 7th International Conference on Manufacturing Engineering, Quality and Production Systems 2014, Brasov, Romania, Athens: WSEAS Press, 2014, pp. 64-68, ISSN 2227-4588, ISBN 978-960-474-387-2.

[12] Rukhin, A., Soto, J., Nechvatal J, Smid, M. Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S., Bassham, L., E., *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publications (800 Series), 2010, SP 800-22 Rev 1a, [Online], Available: http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf [25 Feb 2014].

[13] Pierre, L., Richard, S., *TestU01: A C Library for Empirical Testing of Random Number Generators*. Université de Montréal: ACM Trans, 2007, Math. softw. 33, 4, article 22, [Online], Available: http://dx.doi.org/10.1145/1268776 [10 Feb 2014].

[14] Strnadová, V., *Interpersonální komunikace*, Hradec Králové: Gaudeamus, 2011, 543 p., ISBN 978-80-7435-157-0.

[15] Hubálovský, Š., Musílek, M., *Počítačová bezpečnost ve výuce informatiky (Tvorba hesel a steganografie)*, Matematika-fyzika-informatika, Praha: Prometheus, 2010, vol. 20, November 2010, ISSN 1210-1761.

[16] Voborník, P., *Secure authentication of the client-server application on the internet by using forced unique salt*, Internet, bezpečnost a konkurenceschopnost organizací 2011, Zlín, UTB, pp. 347–354, ISBN 978-80-7454-012-7.

[17] Boon, C., Philippaerts, P., Piessens, F., *Practical experience with the .NET cryptographic API*, Katholieke Universiteit Leuven, CW Reports vol. CW531, 2008, [Online], Available: https://lirias.kuleuven.be/bitstream/123456789/208274/1/CW531.pdf [24 Feb 2014].

[18] Chiunhsiun, L., Ching-Hung, S., Hsuan, S., H., Kuo-Chin, F., *MLB Sports Frames Retrieval Using Color Cipher Similarities*, NAUN International Journal of Circuits, Systems and Signal Processing, Issue 6, Volume 5, 2011, pp. 565-580, ISSN 1998-4464.

[19] Canteaut, A., Chevallier-Mames, B., Gouget, A., Paillier, P., *Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition*, Shabal, 2008, [Online], Available: https://www.shabal.com/wp-content/uploads/Shabal.pdf [19 Feb 2014].

[20] Bertoni, G., Daemen, J., Peeters, M., Assche G., V., *The Keccak reference*, The Keccak sponge function family, 2011, [Online], Available: http://keccak.noekeon.org/Keccak-reference-3.0.pdf [20 Feb 2014].

[21] Soulioti, V., Bakopoulos, Y., Kouremenos, S., Vrettaros, Y., Nikolopoulos, S., Drigas, A., *Stream Ciphers created by a Discrete Dynamic System for application in the Internet*, WSEAS Transactions on Communications, Issue 2, Volume 3, April 2004, ISSN 1109-2742.