Dynamic key generation according to the starting time

A.S. Kiryantsev < reyzor2142@gmail.com>
I.A. Stefanova <aistvt@mail.ru>
Volga Region State University of Telecommunications and Informatics,
77 Moskovskoe sh., Samara, Russia

Abstract. The article analyses the problem of data persistence while transmitting the messages and looks into possible solutions. The central part of the article describes the algorithm of data encryption and digital signature algorithm according to the starting time of the session. In the algorithm the session key is symmetrically generated for each pair of subscribers; further the data are encrypted with this key. In its turn the session key is also encrypted with a public asymmetric key of a recipient and with an asymmetric encryption algorithm. Then the decrypted session key with the decrypted message are sent to the recipient. This client employs the same asymmetric encryption algorithm and his/her secret decryption key to decrypt the asymmetric session key. The decrypted session key is used for decryption of the received message. Thus, every time new symmetric keys are generated according to the starting time of a session, which enables high speed of encryption along with an open to public temporary encryption keys transmitting. Besides, the article contains examples of Diffie-Hellman protocol work and the hash-function algorithm MD5. They are used for encryption of generated temporary keys and for transmitting common private key to both clients. According to the suggested algorithm, the prototype of key and signature generation has been created and probated. The article illustrates the stages of Diffie-Hellman and MD5 protocol work. The prototype was tested with the help of a computer and two phones (2013 and 2015 production years)**.**

Keywords: Diffie-Hellman protocol; MD5-functoin; cryptography; encryption; decryption; digital protection; digital signature; symmetric and asymmetric cryptosystems.

DOI: 10.15514/ISPRAS-2016-28(3)-4

For citation: Kiryantsev A.S., Stefanova I.A. Dynamic key generation according to the starting time. Trudy ISP RAN / Proc. ISP RAS, 2016, vol. 28, issue 3, pp. 51-64. DOI: 10.15514/ISPRAS-2016-28(3)-4.

1. *Introduction*

The necessity of serious approach to information security brings us to the basic concepts of cryptography: digital protection, digital signature and encryption. As you know, cryptography is engaged in the search for solutions to such important security issues as confidentiality, authentication, integrity and control of participants in the interaction.

Encryption is the process of converting data into a form, which is not possible to read the keys. It uses the encryption – decryption keys. The encryption process of the original message helps to ensure privacy by keeping information secret from someone it is not addressed. A set of conversion algorithms and keys used by these algorithms for encryption, key management system, as well as the original and the encrypted text form a cryptographic system. In turn, cryptosystems ensure the secrecy of transmitted messages as well as their authenticity and a user's authentication. The article offers new ideas for dynamic generation of keys and signatures depending on the starting time of the interaction between two subscribers.

2. *Approaches to the construction of cryptosystems*

There are two methods of cryptographic information processing with the keys – symmetric and asymmetric [1]. A symmetric (private) method implies that the sender and receiver use the same key, which they agree before the interaction for both encryption and decryption. If the key has not been compromised, then decrypt database automatically authenticates the sender, since it is only the sender who has the key, which he/she can use to encrypt information, and it is only the recipient who has the key to decrypt the information.

The symmetric encryption algorithms use keys that are not very long and can quickly encrypt large amounts of data. Symmetric encryption systems have a common drawback – that is the complexity of the keys distribution. When an external party intercepts the key, the system of cryptographic protection will be compromised. When it is necessary to replace a key, it should be sent confidentially to the participants of the encryption. Obviously, this method is not suitable when one needs to establish a secure connection with a large number of Internet subscribers. The main problem of this method is how to generate and securely transmit keys to the participants of the interaction. How is it possible to establish a secure communication channel between the participants of interaction while sending keys through insecure communication channels? The lack of a secure key exchange method limits the expansion of symmetric methods of encryption in the Internet.

This problem is resolved in an asymmetric (public) encryption method. In an asymmetric system, the document is encrypted with one key and decrypted with another one. Each participant of the information transfer generates two random numbers (private and public keys). The public key is transferred through public communication channels to another participant of the encryption, but the private key is kept in secret. The sender encrypts the message with the public key of the recipient, and it is only the private key owner who may decrypt the message. This method is suitable for a wide usage. If each Internet user is assigned to his/her own pair of keys and the public keys are published as the numbers in the phone book, almost all users can exchange encrypted messages with each other.

All asymmetric cryptosystems are the object of direct attacks through the direct key enumeration, and, therefore, they must use much longer keys than those used in symmetric cryptosystems to provide an equivalent level of protection. This immediately affects the calculation resources required for encryption.

There is the necessity to verify that there is no distortion into the information in an e-document. Digital signature is used for this sake. Digital signature in a cryptosystem protects a document from changes or substitution and, thereby, guarantees its validity. It is a line, where the attributes of the document (for example, checksum of a file, etc.) and its contents are encoded, so that any change in the file even with the unchanged signature may be detected. When a document is protected by a digital signature, it verifies the document itself along with the private key of the sender, and the recipient's public key. The owner of a private key is the only one who can sign the document correctly. To verify the digital signature of the document, the recipient uses the sender's public key. No other key pair is suitable for verification. Thus, unlike an ordinary signature, digital signature depends on the document and the sender's public key. Therefore, it is several times safer than an ordinary signature and a seal.

Despite the fact that digital signature certifies the authenticity of the document, it does not protect it from unauthorized reading. Both symmetric and asymmetric encryption systems have their advantages and disadvantages. The shortcomings of symmetric encryption are in the complexity of replacing a compromised key, and the disadvantages of asymmetric encryption are in a relatively low speed of work.

These problems are addressed to the encryption systems that use the combined algorithm, which enables high-speed encryption and sending of the encryption keys through the public channels. In order to avoid low-speed of asymmetric encryption algorithms, a temporary symmetric key is generated for each message. The message is encrypted with a temporary symmetric session key. Then this session key is encrypted with a public asymmetric key of a recipient and an asymmetric encryption algorithm. Due to the fact that a session key is much shorter than a message itself, the time of encryption will be relatively short. After that this encrypted session key is transferred to the recipient along with the encrypted message. The recipient uses the same asymmetric encryption algorithm and his/her private key to decrypt the session key and the received session key is used to decrypt the message.

The mentioned above makes it obvious that combined encryption algorithms currently have a promising line of development in modern cryptosystems.

3. *Algorithm description*

It is time to consider the operation principle of the suggested method to data encryption with the session symmetric key, generated at the moment of interaction between the two subscribers. The session key is encrypted with the exposed asymmetric key of the recipient and Diffie-Hellman’s algorithm [2]. The algorithm allows two sides to get common private key, using the channel that is unprotected from discreet listening, but protected from the channel substitution. The received key can be used for message exchange through symmetric encryption.

Diffie-Hellman's algorithm uses one-sided function *F(X)* with two attributes:

- there is a polynomial algorithm of values *F(X)*,
- there is not a polynomial algorithm of inverted function *F(X)*.

To put simply, this function doesn't include decryption of the encrypted text.

The function with a secret is the function Fk; it depends on k and has the following properties: there is a polynomial algorithm of calculation *Fk(X)* value for any k and X, and there is not a polynomial algorithm of the inverted Fk for unknown k; but there is a polynomial algorithm of inverted Fk for the known k parameter.

Fig. 1 presents encryption's block diagram according to the Diffie-Hellman's algorithm.

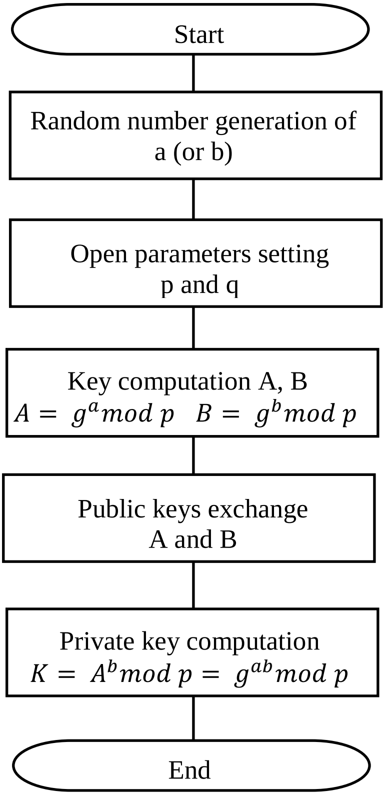

Start
Random number generation of a (or b)
Open parameters setting p and q
Key computation A, B
$A = g^a mod\, p \quad B = g^b mod\, p$
Public keys exchange A and B
Private key computation
$K = A^b mod\, p = g^{ab} mod\, p$
End

Fig. 1. Block diagram of Diffie-Hellman algorithm.

The algorithm operation is presented in the following example. Andrew defines variables g and p which are large numbers. And he also conceives his private number *a* and calculates the value *A* using the formula

$$A = g^{a} mod\, p \tag{1}$$

Then he transmits it to Natasha along with the conceived values of g and p. Natasha conceives her private number b. Through the same formula as Andrew does, she calculates her public number

$$B = g^{b} mod\, p \tag{2}$$

and sends to Andrew. It is possible that the malicious user can get both values, but he will not modify them, as he is unable to interfere in broadcasting process.

At the second stage Natasha calculates the value of K having number B and the received number A:

$$K = A^{b} mod\, p = g^{ab} mod\, p\ , \tag{3}$$

That is the key for encryption. Then, Andrew calculates his key using number B received from Natasha and his calculated number A

$$K = B^{a} mod\, p = g^{ab} mod\, p. \tag{4}$$

You can see in examples (3) and (4) that Andrew gets the same number k, as Natasha. As a result, there is a root key that will be used in generating temporary key and message's signature in the future.

If the root key is used as a private key, a malefactor will be forced to meet with a practically undecidable (for a reasonable period of time) problem of calculating the number $g^{ab} mod\, p$ having numbers $A = g^{a} mod\, p$ and $B = g^{b} mod\, p$, intercepted in the public channel if p, a and b are large enough numbers.

Now it is time to explain the process of temporary key generation. It follows the same HMAC (hash-based message authentication code) algorithm [3] and its standard RFC2104. According to them, information integrity is verified with private key. This standard allows to ensure that transmittable or stored at unreliable environment data were not change by unknown persons.

The HMAC algorithm contains the standard, describing the process of data exchange, the process of data integrity verification with the help of private key and hash-function. Depending on the hash-function, we can distinguish HMAC-MD5, HMDC-SH1 etc.

In the article, the hash-function is generated from the root key by the suggested algorithm, for example: md5 (rootKey + Time). The function md5 is a modification of hash-function MD5. At the generation of hash-function, the time, particularly its second value, will be rounded. As it is known, time is presented in the format HH:MM:SS and rounding happens in the last format's unit. If there are more than 30 sec. in the value of starting time SS, then they round upward, if there are less than 30 sec., then they round downward. The message will be encrypted exactly with this key, and also through this algorithm one can generate digital signature of message to verify the message. As a result, we get a resistant system of dynamic keys for messages encryption and signature, where participants do not need to exchange some data for generation and root key generally.

A generalized algorithm of messages encryption in cryptosystem with the key and signature generation is presented on fig. 2.

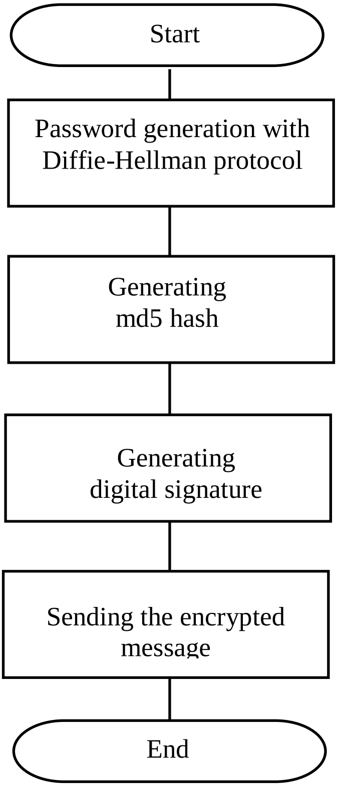

Start
Password generation with Diffie-Hellman protocol
Generating md5 hash
Generating digital signature
Sending the encrypted message
End

Fig 2. Block diagram of encryption by the key and signature generation algorithm.

4. Working prototype

Web technologies and JavaScript language were chosen for prototype realization. Due to it, the program will become a cross-platform and can be loaded everywhere, when there is a support of JavaScript specification (EcmaScript 5) and HTML 4 support. The JavaScript language was not chosen by chance, as at this moment it is the only “native” language for browser and it is supported by all browsers on default.

Below we can consider fragments of prototype code as an example.

Mass Math.random is used for generation of a large number p by Diffie-Hellman algorithm.

This approach is justified by the fact that the JavaScript language cannot work with large numbers (BigInt), as the algorithm requires it.

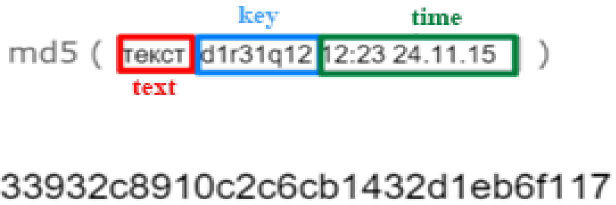
The code generation example of a large number in JavaScript language looks as this:

```
    random(1000000000,9999999999) + " + random(1000000000,9999999999) + "
+   random(1000000000,9999999999) + " + random(1000000000,9999999999) + "
+   random(1000000000,9999999999) + " + random(1000000000,9999999999) + "
+   random(1000000000,9999999999) + " + random(10000000,99999999);
```

Then the code of message's generation to JavaScript language seems:

```
    $scope.getSign = function()
       {
         return    md5($scope.msg    +    $scope.username    +    bigInt2str(a_sec,
10).toString() + datetime);
       }
```

Function md5(arg) returns the hash line from argument arg. Function bigInt2str is a function that allows to work with large numbers in JavaScript. $scope.username allows to insert a username. In this way we get a unique signature for each user. There is a screenshot of text values' substitution and the result of the performed program:



5c733932c8910c2c6cb1432d1eb6f117

The time test script execution was conducted through the prototype. In this test the following e-devices were used:

1. The computer – INTRL i5 (Windows 10/chrome)
2. The phone – Nexus 5 (android 6.0.1/ chrome)
3. The phone – Samsung galaxy ace (android 4.2.2/ browser).

In table 1 the results of the algorithm individual steps are provided. The steps are applied in different application. In fig. 3 there is a diagram that visualizes experiment results. From the table analysis it is obvious that the algorithm works very fast on the mobile phones.

Hash-function algorithm MD5 is not selected occasionally, it is the fastest, the most common one. It has the simplest hashing algorithm that may be used for signature generation. Besides MD5 possesses a very interesting property. For instance, if at least one byte in a line is changed, the view of the resulting hash line will change dramatically.

Table 1. Time of algorithm application in different devices at different stages (msec).

devices algorithm	Intel i5 (Windows/chrome)	Nexus 5 (android 6.0.1/ chrome)	Samsung galaxy ace (android 4.2.2/ browser)
Diffie-Hellman generation	20,915	166,706	220,53
MD5 generation	0,81	3,315	6,21
Sign generation	0,27	0,48	0,72
Total time	22,883	170,89	229,416

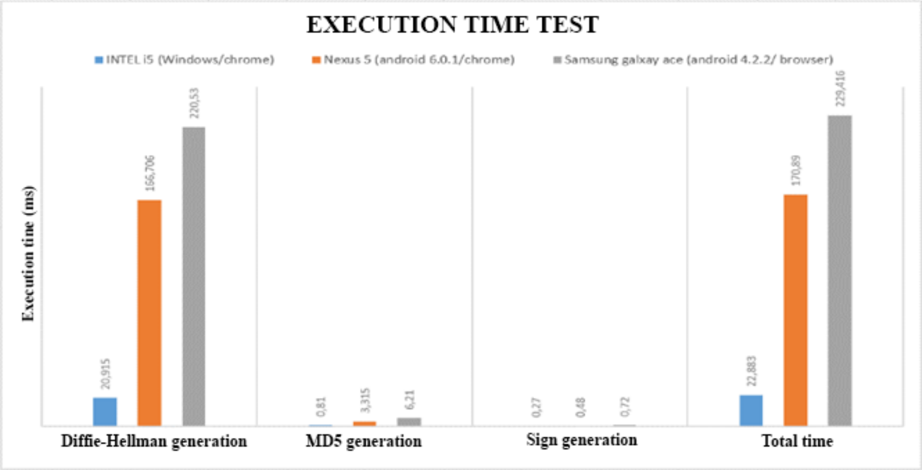


Fig 3. Histogram of algorithm performance time by different e-devices.

The logic of the encryption algorithm can be considered in five steps. After the data are received there is the process of preparing the data flow to the calculations.

Step 1. First, the flow line requires alignment for hashing. At the end of the stream one on-bit and the necessary number of off bits are registered. After the input data alignment, the length of the stream should be equal to 512 · N + 448.

Step 2. At the end of the message, one should add 64-bit result for alignment. There are 4 low-order bits that are put first, then high-order bits follow. If the stream length exceeds 2^{64}– 1, only low-order bits are written down. After that, the stream length becomes 512-fold. The calculations are made with data flow presented as an array of 512-bit words.

Step 3. Then it is necessary to initialize 4 32-bit variables (A, B, C, D) and to set their initial values with hex numbers: "low-order byte comes first". For example,

A = 01 23 45 67; // 67452301h

B = 89 AB CD EF; // EFCDAB89h

C = FE DC BA 98; // 98BADCFEh

D = 76 54 32 10. // 10325476h

The results of intermediate calculations will be stored in these variables. Then it is time to initialize constants and functions required in further calculations.

Four laps will require 4 functions with the logical operators XOR (⊕), AND (∧), OR (∨), NOT (¬):

$$FunF(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z),$$

$$FunG(X,Y,Z) = (X \wedge Z) \vee (\neg Z \wedge \mathrm{Y}),$$

$$FunH(X,Y,Z) = (X \oplus \mathrm{Y} \oplus Z),$$

$$FunI(X,Y,Z) = Y \oplus (\neg Z \vee \mathrm{X}).$$

The 64-element table of invariables is structured as follows:

$$T[n] = \ int(2^{32} \cdot |\sin(n)|)$$

Each 512-bit block of the flow passes through 4 stages of calculation, 16 laps each. For this the block is presented as an array X of 16 32- bit words. All the laps are of the same type, but they differ in the rotate shift by *s* bits of a 32-bit argument. The number s is defined for each lap.

Step 4. Steps in loop calculations. Base n element into the block from an array of 512-bit blocks. The values A, B, C, D, remain after operations with the previous blocks (or their values in case the array goes first).

AA = A

BB = B

CC = C

DD = D

Sum the values with the result of the previous loop:

A = AA + A

B = BB + B

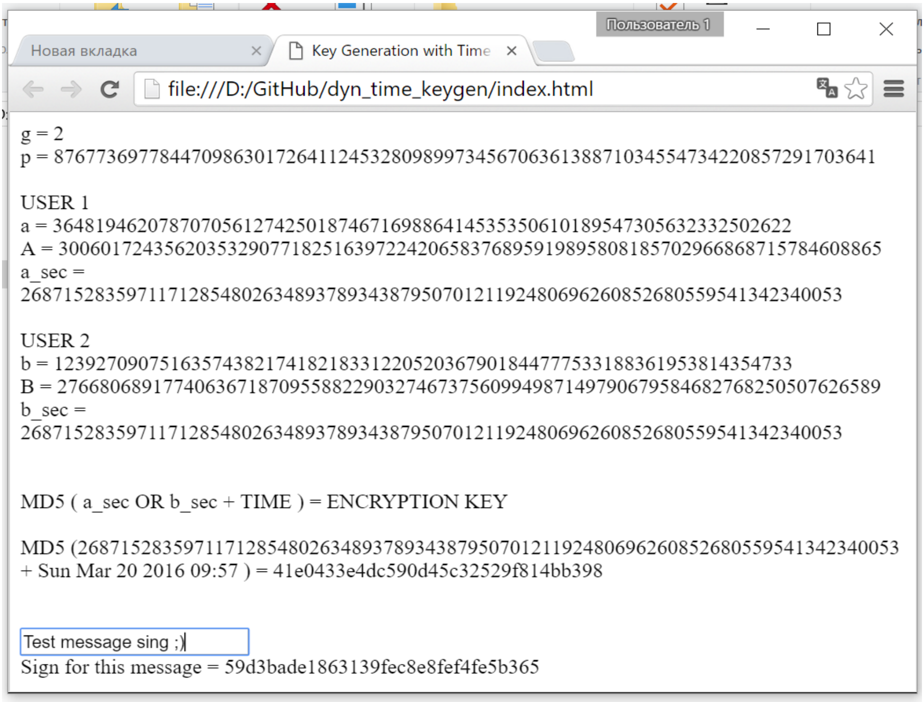
C = CC + C

D = DD + d

After the loop ends, check if there are any blocks for calculations left. If there are some, go to the next array element (n+1) and the loop repeats.

Step 5. The result of the hash-function calculation is formed in ABCD buffer. If the result starts with the low-order byte A, one gets MD-5 hash.

Fig. 4 presents a screenshot of md5 hash function working prototype in the CRYP2CHAT app [4]. It resorts to a modified MD5 hash function.



```
g = 2
p = 876773697784470986301726411245328098997345670636138871034554734220857291703641

USER 1
a = 3648194620787070561274250187467169886414535350610189547305632332502622
A = 300601724356203532907718251639722420658376895919895808185702966868715784608865
a_sec =
268715283597117128548026348937893438795070121192480696260852680559541342340053

USER 2
b = 1239270907516357438217418218331220520367901844777533188361953814354733
B = 276680689177406367187095588229032746737560994987149790679584682768250507626589
b_sec =
268715283597117128548026348937893438795070121192480696260852680559541342340053


MD5 ( a_sec OR b_sec + TIME ) = ENCRYPTION KEY

MD5 (268715283597117128548026348937893438795070121192480696260852680559541342340053
+ Sun Mar 20 2016 09:57 ) = 41e0433e4dc590d45c32529f814bb398


Test message sing ;)
Sign for this message = 59d3bade1863139fec8e8fef4fe5b365
```

Fig 4. Prototype of Application work.

5. *Evaluation of algorithm effectiveness*

The cryptographic strength of the proposed algorithm for key and signatures generation depends on the encryption method that combines the algorithms of symmetric and asymmetric encryption.

The cryptographic strength is a quantitative characteristic of encryption algorithms – intrusion into a particular algorithm requires a certain number of resources. This is the amount of information and time required to perform the attack, as well as the memory required to store information used in the attack.

An attacking encryption algorithm typically aims at solving the following tasks:

- to get public text version from the encrypted one,
- to calculate the encryption key.

The second task is usually more challenging than the first one. However, having the encryption key the cryptanalyst can later decrypt all the data encrypted with a key.

The algorithm is considered to be secure, if a successful attack at it requires from an attacker unattainable calculating resources in practice, or open intercepted and encrypted messages, or if decryption is so time-consuming that currently protected information would lose its relevance. In most cases, the cryptographic strength cannot be mathematically proven, you can only prove the vulnerability of the algorithm or to calculate the time required to find the key. For this sake, one should take into consideration the difficulty of a given mathematical problem that serves as the basis for the encryption algorithm.

To estimate the time of the password configuration to gain an unauthorized access to the channel of two subscribers, we have the equation [5]:

$$t = \frac{N^0+N^1+N^2+N^L}{V} \tag{5}$$

It estimates time in the worst case. Here t is the time required for the guaranteed password configuration, V is the number of combinations per second in brute search, N is the number of characters in the configurated password, L is the length of the password.

In case with the md5 algorithm, the number of characters is 36. This number includes the 26 symbols-letters in the Latin alphabet (a...z) and 10 symbols of Arabic numerals (0..9). The number of symbols in the secure key for encryption or signing is 32. To calculate speed of the brute symbol search, we'll take an Intel i7 and a video card Radeon HD5850 1024 MB. Their power equals to 65 000 passwords per second, calculated empirically.

As a result of substitution of values in (5) the estimated time will be:

$$t = \frac{36^0 + 36^1 + 36^2 + 36^{32}}{6500} = 9.745\text{x}10^{44}\text{с.}$$

Converting the seconds into a larger value, we get the result 3.09 $\times$ 10^{37} years.

Conclusion: this algorithm can be considered secure from attack and the encryption key calculation, as the time for the key search outweighs the actual time of work with data.

In sources [5, 6] an algorythm of dynamic key generation is offered. It is presented as a self-authenticated method with timestamp. In the patent the author employs asymmetric encryption-decryption algorithm. In contrast in this article the described algorithm is symmetric. This helps exclude sending and receiving any key, which increases security of data transmission. Moreover, Google team uses slightly similar algorithm of key generation. However, its development group employs another hash function that is not connected with encryption. Additionally, password configuration is a part of the algorithm that we provide.

6. *Conclusion*

The algorithm for temporary keys and signatures generation can be used to teach students the basics of cryptography, and used in real projects. Coupled with a VPN or TOR networks it becomes more secure due to the new encryption level [7].

References

[1]. Mikhail Adamenko. The basics of classical cryptology. The secrets of ciphers and codes. DMK Press [DMK Publishing], 256 p., 2014 (in Russian).

[2]. Diffie, W. and Hellman, M. E. New directions in cryptography. IEEE Transactions on Information Theory, vol. 22, issue 6, 1976, pp. 644-654.

[3]. Maurer U.M, Wolf S. The Diffie-Hellman Protocol. Retrieved. Designs, Codes and Cryptography. Special Issue: Public Key Cryptography, № 19, 2000, pp.147-171.

[4]. The construction of the password generator. Retrieved from www.scritub.com/limba/rusa/194620205.php, 2013-08-02 (accessed February, 2016) (in Russian).

[5]. Self-authenticated method with timestamp. Patent US 20140325225 A1. Retrieved from http://www. google.com/patents/US20140325225 (accessed Oct. 30, 2014).

[6]. SELF-AUTHENTICATED METHOD WITH TIMESTAMP - DIAGRAM, SCHEMATIC, AND IMAGE. Retrieved from http://www.faqs.org/patents/imgfull/ 20140325225_06 (accessed Oct. 30, 2014 Sheet 5 of 5).

[7]. Kiryantsev A.C., Stefanova I.A. Constructing Private Service with CRYP2CHAT application. Trudy ISP RAN / Proc. ISP RAS], vol. 27, issue 3, 2015, pp. 279-290. DOI: 10.15514/ISPRAS-2015-27(3)-19

Генерация динамических ключей и подписей с зависимостью от времени

А.С. Кирьянцев < reyzor2142@gmail.com>
И.А. Стефанова <aistvt@mail.ru>
Поволжский государственный университет телекоммуникаций и информатики, Самара, Московское шоссе, 77

Аннотация. В статье рассмотрена проблема сохранности переписки при передаче и пути решения этой проблемы. Центральную часть статьи составляет описание алгоритма генерации паролей для шифрования данных и генерации подписей для сообщений с зависимостью от времени начала взаимодействия двух абонентов. В предлагаемом комбинированном алгоритме используется генерация временного симметричного сеансового ключа для каждой пары абонентов с последующим шифрованием этим ключом передаваемого сообщения. В свою очередь сам сеансовый ключ шифруется с помощью открытого асимметричного ключа получателя и асимметричного алгоритма шифрования. Далее зашифрованный сеансовый ключ вместе с зашифрованным сообщением передается получателю, который использует тот же самый асимметричный алгоритм шифрования и свой секретный ключ для расшифровки симметричного сеансового ключа, а полученный сеансовый ключ используется для расшифровки самого принятого сообщения. Таким образом, симметричные ключи генерируются каждый раз новые, в зависимости от времени установки связи между парой абонентов, что позволяет при высокой скорости шифрования использовать открытую пересылку временных ключей шифрования. А для их сохранности уже использовать асимметричные методы шифрования. Кроме того в статье рассмотрены примеры работы протокола Диффи-Хеллмана и алгоритма хеширования MD5, используемые для шифрования генерируемых временных ключей и позволяющие двум сторонам получить общий секретный ключ. По предложенному алгоритму был создан прототип с реализаций генерации пароля и генерации подписи, который наглядно показывает этапы работы протокола Диффи-Хеллмана и MD5. С помощью прототипа было проведено тестирование на предмет времени исполнения алгоритма на трёх устройствах: на одном стационарном компьютере с видеокартой и двух телефонах (2013 и 2015 годов выпуска).

Ключевые слова: Протокол Диффи-Хеллмана; MD5 функция; криптография; шифрование; дешифрование; цифровая защита; цифровая подпись; симметричные и асимметричные криптосистемы.

DOI: 10.15514/ISPRAS-2016-28(3)-4

Для цитирования: Кирьянцев А.С., Стефанова И.А. Генерация динамических ключей и подписей с зависимостью от времени. Труды ИСП РАН, том 28, вып. 3, 2016 г. стр. 51-64 (на английском). DOI: 10.15514/ISPRAS-2016-28(3)-3.

Список литературы

[1]. Михаил Адаменко. Основы классической криптологии. Секреты шифров и кодов, Издательство ДМК 2014 - 256 с.

[2]. Diffie, W. and Hellman, M. E. New directions in cryptography. IEEE Transactions on Information Theory, vol. 22, issue 6, 1976, pp. 644-654).

[3]. Maurer U.M, Wolf S. The Diffie-Hellman Protocol. Retrieved. Designs, Codes and Cryptography, Special Issue: Public Key Cryptography, № 19, 2000, p.147-171.

[4]. Как устроен генератор паролей?: www.scritub.com/limba/rusa/194620205.php, 2013-08-02 (доступ февраль, 2016).

[5]. Self-authenticated method with timestamp. Patent US 20140325225 A1. Retrieved from http://www. google.com/patents/US20140325225 (accessed Oct. 30, 2014)

[6]. Self-authenticated method with timestamp - diagram, schematic, and image 06. Retrieved from http://www.faqs.org/patents/imgfull/20140325225_06 (accessed Oct. 30, 2014 Sheet 5 of 5)

[7]. Kiryantsev A.C., Stefanova I.A. Constructing Private Service with CRYP2CHAT application. Trudy ISP RAN / Proc. ISP RAS], vol. 27, issue 3, 2015, pp. 279-290. DOI: 10.15514/ISPRAS-2015-27(3)-19.