Diceware Password Generation Algorithm Modification based on Pseudo-Random Sequences

Serhii Buchyk[1], Nataliia Lukova-Chuiko[1], Serhii Toliupa[1], Vitalii Piatyhor[1], and Oleksandr Buchyk[1]

[1] *Taras Shevchenko National University of Kyiv, 24 Bohdan Hawrylyshyn str., Kyiv, 04116, Ukraine*

Abstract

The authors investigated main algorithms for generating passwords that exist in the market and analysed their advantages and disadvantages. The complexity of the generated passwords was estimated based on the finite entropy and the complexity of memorization. The international requirements for password generation, provided in the NIST.SP.800-63b standard and the compliance of the received passwords with the requirements given in the document, were considered. The resistance of the password to brute force attacks and dictionary attacks was also evaluated. A software application has been developed that implements password generators using a modified Diceware password generation algorithm based on a given length and a given set of words and estimates the entropy of generated passwords. The list of words is written in a file. The English dictionary is used by default. The main goal of this program is to generate a set of passwords. The program can work autonomously without a network connection. A modified algorithm for generating Diceware passwords is recommended to use instead of the original one due to the higher entropy of the former.

Keywords

Password authentication, password, password generation, diceware, entropy, password strength estimation.

1. Introduction

Nowadays, a lot of attention is given to the issue of protecting information from unauthorized access. One of the security services that provides such protection is the authentication service. Currently, each system uses some type of user authentication, most often – password based.

However, without sufficient attention to user-generated passwords, such a system is quite vulnerable to hacker attacks. Users of the system often create easily crackable passwords, such as passw0rd, admin, etc. which can be guessed in a few seconds. Therefore, to improve the resilience of passwords to cracking, password generators are used based on random or pseudo-random number sequence generators, which can provide users with a complex password, that will be used in the system to verify the identity of the user.

Without the use of software, the system may impose additional restrictions to the length, characters and content of the password, but studies [1, 2] show that imposing too complex rules on the password significantly increases the chance of users forgetting or entering the wrong password. The same arises when using the passwords that have been generated by password generation algorithms that focus on the complexity and use of all characters in a random sequence rather than on the user ability to memorize it.

Therefore, to improve the complexity of the password and make it easier to remember at the same time, it is recommended to use the Diceware random password generation algorithm. Otherwise, it is recommended to use a centralized keystore, where user passwords will be stored.

CPITS-II-2021: Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2021, Kyiv, Ukraine
EMAIL buchyk@knu.ua (S. Buchyk); lukova@ukr.net (N. Lukova-Chuiko); tolupa@i.ua (S. Toliupa); vp5gor@gmail.com (V. Piatyhor); butechkomp@gmail.com (O. Buchyk)
ORCID: 0000-0003-0892-3494 (S. Buchyk); 0000-0003-3224-4061 (N. Lukova-Chuiko); 0000-0002-1919-9174 (S. Toliupa); 0000-0002-7621-1299 (V. Piatyhor); 0000-0001-7102-2176 (O. Buchyk)

2. Task Formulation

The aim of the study is the modification of the Diceware password generation algorithm based on pseudo-random sequences.

The object of research is the process of generating passwords based on pseudo-random sequences.

The subject of research is algorithms for generating passwords based on pseudo-random sequences, the complexity of the passwords generated by them.

Thus, the task is to improve the algorithm for generating Diceware passwords based on pseudo-random sequences, which will make the algorithm more resistant to attacks than the standard implementation and increase the entropy value per character.

3. Solving the Task

Let's define some basic terms used in the article.

A pseudo-random sequence is a sequence of numbers that has been computed according to a certain arithmetic rule, but has all the properties of a random sequence of numbers within the boundaries of the problem to be solved. Although the pseudo-random sequence at first glance has no consistent pattern, but any pseudo-random generator with a finite number of internal states will be repeated after a very long sequence of numbers. This can be proved by the Dirichlet’s principle.

The generation of random numbers that no observer can ever predict requires a causal-indeterminate process where events are not completely determined by previous states. Due to the physical inability to obtain sufficient information to predict the outcome of such an event, its results are guaranteed to be random for all.

A random password generator is a software or hardware device that receives data from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually using simple randomness sources such as dice or coins, or they can be generated using a computer.

Although there are many examples of “random” password generator programs available on the Internet, generating random cases can be difficult, and many programs do not generate random characters in a way that provides high security. A common recommendation is to use open source security tools where possible, as they allow you to independently check the quality of the methods used.

It should be noted that simply generating a password at random is not a guarantee that the password is secure, as it is possible, though very unlikely, that the generated password will be easily cracked.

Random password generators usually output a string of characters of a given length. These can be individual characters from a character set or words from a word list to form a passphrase. You can configure the program so that the generated password follow the local password policy, for example, always creating a mixture of letters, numbers and special characters. Such policies tend to slightly reduce the strength of the password, the formula for which is calculated below, because the characters are no longer created independently.

The strength of a random password against a specific attack (brute force attack) can be calculated by calculating the information entropy of the random process that generated the password. If each character in the password is generated independently and with uniform probability, the entropy in bits is given by the following formula.

$$H = L\log_2 N = L\frac{logN}{\log 2}, \qquad (1)$$

where N is the number of possible characters and L is the number of characters in the password.

For comparison, a mechanical 72-character password generation algorithm, an algorithm using AES-256 PBKDF2-SHA256, a standard Diceware implementation, and a own modification of the latest were selected.

The mechanical generator is implemented using the ANSI X9.17 pseudo-random number generator and uses 72 characters, which include lowercase and uppercase English letters, numbers and some characters. Generation is performed by obtaining three pseudo-random numbers using a generator, which are used as indices in a three-dimensional array to access the desired character. The operation is repeated until the desired password length is reached [3, 4].

As an example of a commercial product for generating passwords, an algorithm that uses AES-256 PBKDF2-SHA256 was chosen, which was provided by the software application LastPass [5]. This application is commercial, thus the entire implementation of this generator is closed, so the information about the generation tools was taken from the product description and documentation.

Diceware is a method of creating password phrases, passwords and other cryptographic variables, using ordinary dice as a hardware generator of random numbers [6]. Each word in the passphrase requires five dice rolls. The resulting numbers 1 to 6 are added as a five-digit number, such as 43146. This number is then used to search for a word in the word list. By generating several words sequentially, you can build a long, easy-to-remember, and easy-to-choose password.

Arnold Reinhold, the inventor of the algorithm recommends using only the tools of “true” randomness and not to use tools for programmatic generation of passwords in this way because of the use of pseudo-randomness in such programs, in other words there is always a repeatable pattern. However, with the use of modern means of generating pseudo-random numbers, such dependencies are very difficult to trace.

Modification of this algorithm involves generating a password not only by choosing words, but also by randomly changing a capital letter in each word and replacing the letters with some numbers, such as the letter “o” by 0. More details will be given below.

In Ukraine, there is no legal framework that would regulate the requirements for the passwords used, because of this as the indicator of a good password the requirements of the standard NIST.SP.800-63b [7], which regulates the properties that a certified password must have, will be used.

There is the section in this document designed for passwords that are to be remembered. The following requirements are put towards them:

- Length from 8 to 64 characters.
- All ASCII and UNICODE characters must be present.
- For generated passwords approved random bit generators specified in the regulatory document NIST.SP.800-90Ar1 [8] must be used.
- Passwords should not be previously compromised, in the database of light passwords, depend on the context and have repetitive characters, as well as use words from the dictionary.

The flowchart of the modified Diceware password generation algorithm based on pseudo-random sequences is presented in Fig. 1.

To show the results of the modified algorithm, a software application was developed that implements this algorithm, as well as calculates the entropy and entropy per symbol for the generated password. The application has the ability to set a custom password length from 3 to 20 characters, add a custom word list and select a password generation algorithm. The right half of the application displays the result of the program, namely 5 generated passwords, from which file the words for generating the Diceware password were read, the entropy value and the entropy value per character for the generated passwords. The interface of the software application is shown in Fig. 2.

The application implements two ways to generate a password: a modified Diceware and a mechanical one. The mechanical algorithm for generating a password is based on the principle of generating three random values: two of them have values from 0 to 5 and one value from 0 to 1. The alphabet used has the form of two tables of 36 characters. The first 36 characters include lowercase Latin characters and Arabic numerals 0 to 9. The rest are uppercase Latin characters and special characters. Values from 0 to 1 determine which of the two alphabets to use to search for a character, the other two values determine the index of the character in the symbol table. The process is repeated until the desired set password length is reached. The result of the application in this mode is shown in Fig. 3.

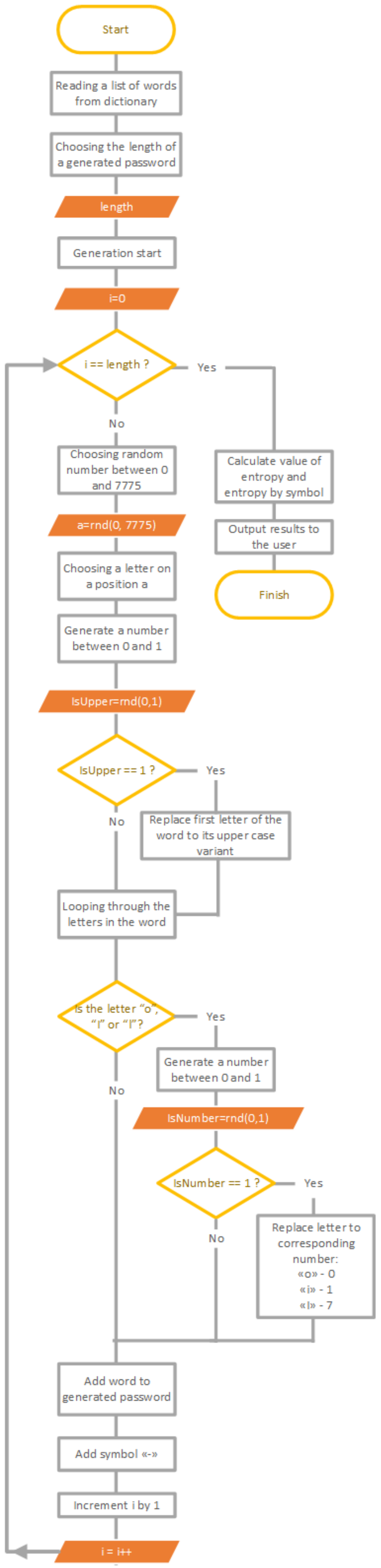
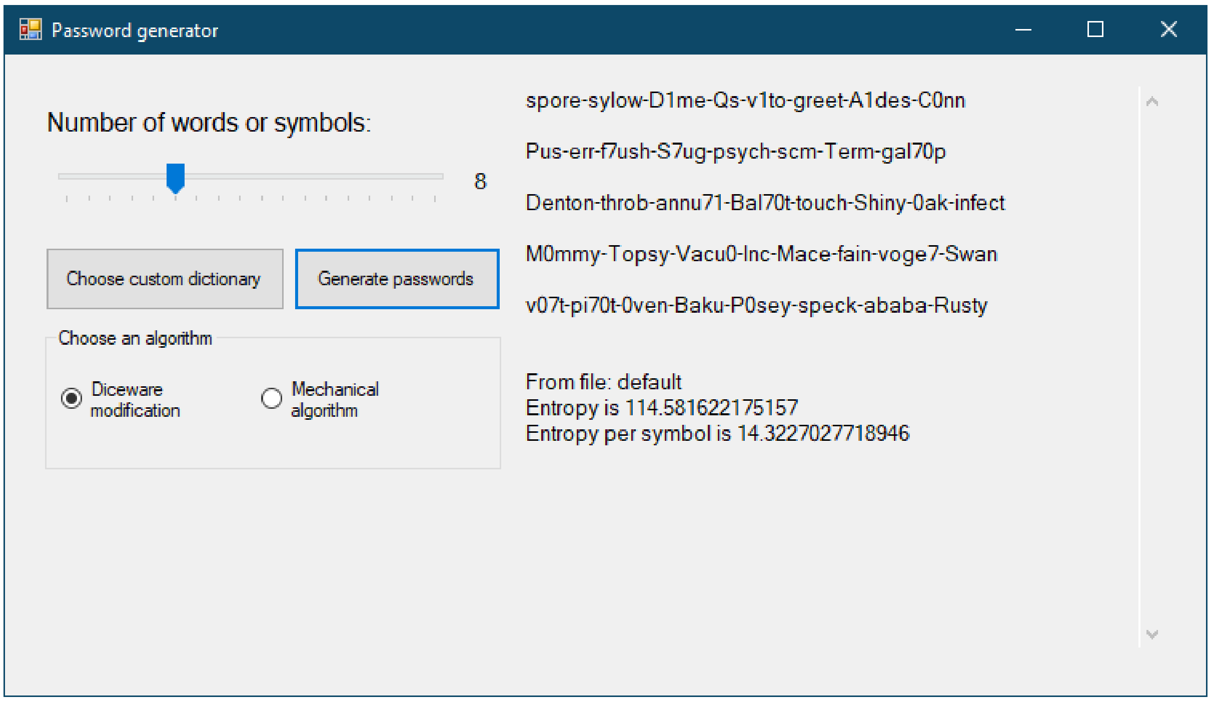

Start
Reading a list of words from dictionary
Choosing the length of a generated password
length
Generation start
i=0
i == length ?
Yes
No
Calculate value of entropy and entropy by symbol
Output results to the user
Finish
Choosing random number between 0 and 7775
a=rnd(0, 7775)
Choosing a letter on a position a
Generate a number between 0 and 1
IsUpper=rnd(0,1)
IsUpper == 1 ?
Yes
No
Replace first letter of the word to its upper case variant
Looping through the letters in the word
Is the letter "o", "i" or "l"?
Yes
No
Generate a number between 0 and 1
IsNumber=rnd(0,1)
IsNumber == 1 ?
Yes
No
Replace letter to corresponding number:
«o» - 0
«i» - 1
«l» - 7
Add word to generated password
Add symbol «-»
Increment i by 1
i = i++

Figure 1: Flowchart of a modified algorithm for generating Diceware passwords based on pseudo-random sequences

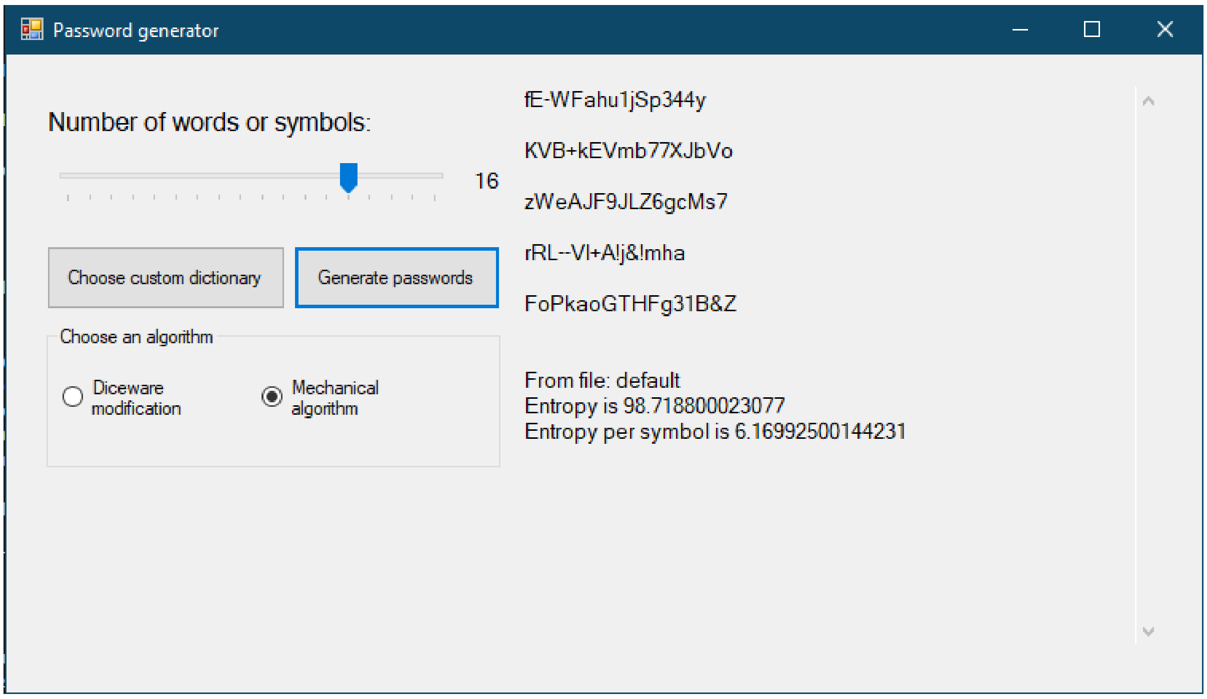


Number of words or symbols: 8

Choose an algorithm
[✓ Diceware modification] Mechanical algorithm

```
spore-sylow-D1me-Qs-v1to-greet-A1des-C0nn

Pus-err-f7ush-S7ug-psych-scm-Term-gal70p

Denton-throb-annu71-Bal70t-touch-Shiny-0ak-infect

M0mmy-Topsy-Vacu0-Inc-Mace-fain-voge7-Swan

v07t-pi70t-0ven-Baku-P0sey-speck-ababa-Rusty

From file: default
Entropy is 114.581622175157
Entropy per symbol is 14.3227027718946
```

Figure 2: Software application interface



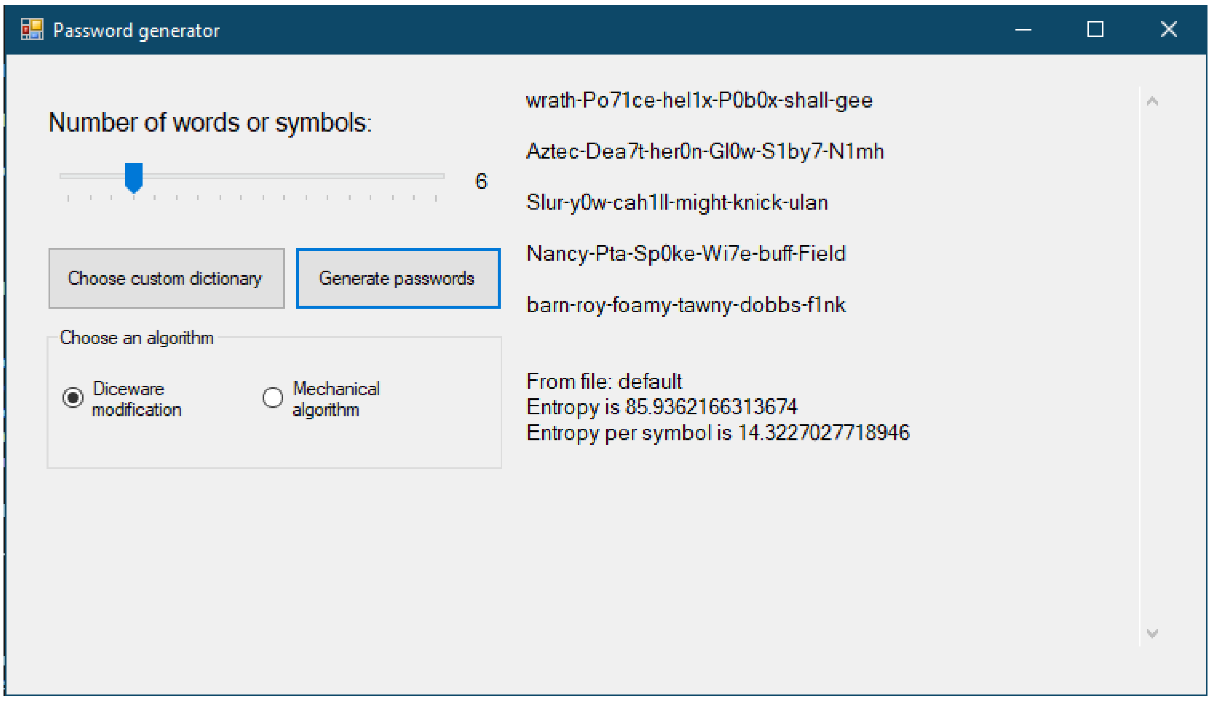
Number of words or symbols: 16

Choose an algorithm
Diceware modification [✓ Mechanical algorithm]

```
fE-WFahu1jSp344y

KVB+kEVmb77XJbVo

zWeAJF9JLZ6gcMs7

rRL--Vl+A!j&!mha

FoPkaoGTHFg31B&Z

From file: default
Entropy is 98.718800023077
Entropy per symbol is 6.16992500144231
```

Figure 3: Example of application output in “Mechanical algorithm” mode

The initial stage of the modified Diceware algorithm consists of reading a set of words from a file, by default - an English dictionary. The next step is word generation. First, a number from 0 to 7775 is generated, a word is selected from a set of words, the index of which corresponds to the generated number. A number from 0 to 1 is then generated, which determines whether the word will have a capital letter or not. The next step is to search for the remaining characters. If the symbol is “o”, “i” or “l”, then a number will be generated between 0 and 1, which determines whether the symbol will be replaced by “0”, “1” or “7” respectively, or not. The process is repeated until the desired password length is reached. In this case, the length of the password is determined not by the number of characters, but by the number of words generated in the resulted password. The final step is to combine the resulting words into a single one, with the words separated by a hyphen. An example of generated passwords is given in Fig. 4.



```
wrath-Po71ce-hel1x-P0b0x-shall-gee

Aztec-Dea7t-her0n-Gl0w-S1by7-N1mh

Slur-y0w-cah1ll-might-knick-ulan

Nancy-Pta-Sp0ke-Wi7e-buff-Field

barn-roy-foamy-tawny-dobbs-f1nk

From file: default
Entropy is 85.9362166313674
Entropy per symbol is 14.3227027718946
```

Figure 4: Example of application output in "Diceware modification" mode

Users are given the opportunity to choose their own set of words to generate passwords. The file requirements are as follows:

- The file should have only one set of index and word per term.
- Word index and word must be separated by a tab character.
- File encoding must be UTF-8.

The password generation process is the same for word sets with Latin characters as for a standard set of words. In the case of Cyrillic, only the capital letter and the symbol "o" are replaced. Depending on the selected set of words, both the entropy and the entropy per symbol change. This is due to the difference in the number of characters that can be replaced in different sets of words.

The ANSI X9.17 generator, which is adopted as the Federal Standard for Information Processing in the United States, is used as a pseudo-random number generator. The generation process consists of 4 steps:

1. Get the current time with the maximum possible accuracy.
2. Calculate the temporary value by encrypting the current date with the Triple DES encryption algorithm.
3. Calculate the result of the exclusive "or" between the temporary value and the random initial value. The result is encrypted using the Triple DES algorithm and is the resulting random value.
4. Calculate the result of the exclusive "or" between the obtained random value and the temporary value calculated in step 2. The result is encrypted using the Triple DES algorithm and replaces the value of the random initial value.

In the implemented case, at the end of the generation, the module is taken from the original random number.

Determining password complexity in a software application is determined by entropy and character entropy. Formulas 1 and 2, respectively, are used to calculate them.

$$H_{symbol} = \frac{H}{L}, \tag{2}$$

where H is the total entropy, L is the number of characters in the password.

The results of the analysis of entropy and entropy per symbol for passwords generated by each algorithm are given in Table 1.

Table 1

Entropy values for different password generation algorithms and length

Algorithm	6 symbols	8 symbols	16 symbols	Entropy per bit
	Entropy, bit	Entropy, bit	Entropy, bit	
Mechanical	32.02	49.36	98.72	6.17
AES-256 PBKDF2-SHA256	39.33	52.44	104.87	6.55
Diceware	77.55	103.41	206.88	12.92
Diceware modification	85.94	114.58	229.16	14.32

The results show that using the modified Diceware algorithm, the entropy per symbol is 1.4 bits higher than the standard algorithm.

According to RFC 4086 [9], it is recommended to use keys with an entropy value starting from 29 bits for online attacks to 96 bits for cryptographic keys. As of the end of 2011, distributed.net estimated the time required to crack a 72-bit password as 124.8 years. Currently, it is recommended to use passwords with an entropy of at least 80 bits. Following this, for the password generated by the above algorithms to implement the recommendation it must be a length of 13, 13, 7 and 6 characters or words, respectively [10].

Fig. 5 shows the comparison of the entropy of the generated passwords depending on the algorithm and length.

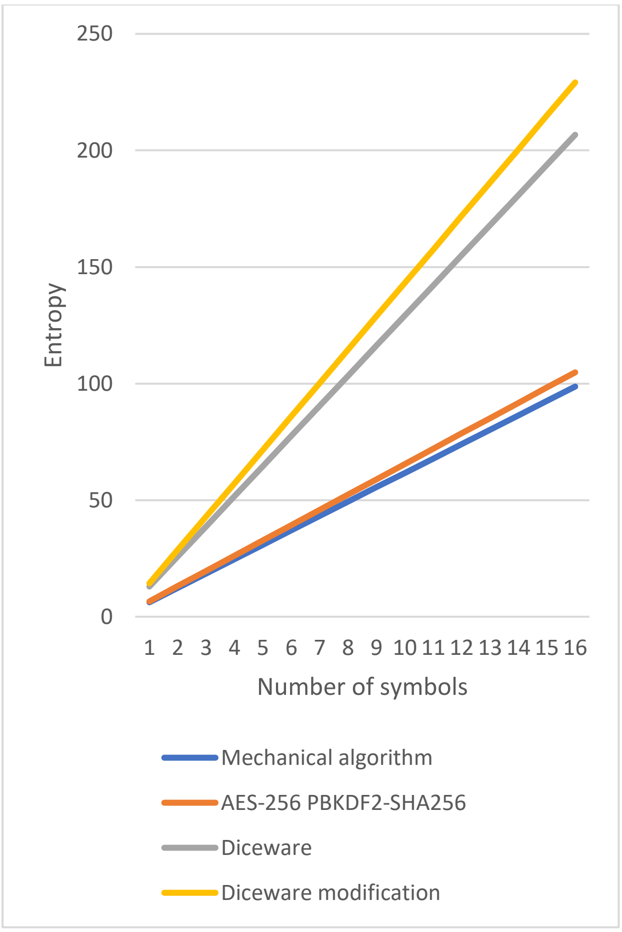


Figure 5: Comparison of entropy of generated passwords

To evaluate the memorability of passwords, in Table 2 five variants of generated passwords are presented as examples.

Table 2
Examples of generated passwords

Mechanical algorithm	AES-256 PBKDF2-SHA256	Diceware	Diceware modification
bGyfZX?T	$IcSJzaWDse%	bid-net-blunt-snafu-frenzy-knurl	pr1ze-dewy-k0ng-p1ggy-wt-Tax
xO&4$6GU	iG5Q5Y*9egw1	deja-frock-hunk-clime-wise-holm	k71ne-tone-basin-Ag0ny-Gt-B700p
rM?KLyMm	cbFyv94Rj*MC	ea-zk-bribe-luxe-world-raul	Carb0n-Um-Pion-exact-thumb-Nasty
w0tKMh?V	1Mh*5XZbaLUi	bind-ultra-bethel-septa-texan-onion	Coc0-Veery-ex-Chaos-P7ague-Avert
+91-kGnU	qdUiS5e@qDq$	siena-chock-er-gander-stein-43	laban-gab0n-M1st-cpa-we77-pab7o

As you can see, the standard Diceware algorithm is the easiest to read and remember. Non-Diceware based algorithms are too difficult to remember, so it's best to use them with a password manager.

Although a modified Diceware is more difficult to remember than a standard implementation, it is not as complex as, for example, non-Diceware based ones.

The results show that increasing entropy has made it more difficult to read and remember a password, although it can still be remembered by the average user. It should be noted that this assessment is subjective [11–14].

In the Table 3 shows the matrix of compliance of password generation algorithms to the NIST SP 800-63b standard. Only those criterias that depend on the progress of the program or algorithm, and not those that provide, for example, pre-verification of compromised passwords were selected. Such actions can be easily added to the algorithm, but are not required for now.

Table 3
Password generation algorithms compliance with NIST SP 800-63b standard

Criteria	Algorithm			
	Mechanical	AES-256 PBKDF2-SHA256	Diceware	Diceware modification
Length	+	+	+	+
All ASCII and UNICODE symbols	72	96	70	96
Repeatable words/symbols	Possible	Possible	Possible	Possible
Dictionary words	–	–	+	+

As you can see, none of the algorithms meet NIST standards.

4. Conclusions

In this paper, a modification of the Diceware password generation algorithm based on a pseudo-random number generator was presented and it was compared with existing algorithms. As can be seen from the results of the analysis, this modification of the algorithm is more resistant to attacks than the standard implementation and has a character entropy of 1.4 bits higher, but there is a drawback—increasing the difficulty of remembering this password.

Also in this work, for the sake of demonstration, a software application was developed that implements the above-mentioned modification of the algorithm and a mechanical algorithm for password generation, as well as calculates the complexity of the password based on entropy.

In conclusion, the use of this modification of the algorithm is simpler than the use of 12–16-character not Diceware-based algorithms generated passwords, due to the ease of memorization and provides a greater level of password protection against "brute force" attacks. Therefore, in the case of memorized passwords and the absence of a password manager, the use of this modification is desirable.

However, it is not recommended to use this algorithm to generate the password alone. The best solution is to combine the password with another authentication method, i.e. to introduce multifactor authentication.

5. References

[1] Y.-Y. Choong, Password Usability, 2015. https://csrc.nist.gov/CSRC/media/Presentations/Password-Usability/images-media/oct23_choong_password.pdf.

[2] S. Komanduri, et al., Of Passwords and People: Measuring the Effect of Password-Composition Policies: ACM.D.4.6; ACM.H.1.2 / Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman. Vancouver, BC, Canada, 2011: https://www.andrew.cmu.edu/user/nicolasc/publications/KSKMBCCE-CHI11.pdf.

[3] V. Buriachok, V. Sokolov, P. Skladannyi, Security rating metrics for distributed wireless systems, in: Workshop of the 8th International Conference on Mathematics. Information Technologies. Educatio (MoMLeT), vol. 2386, 222–233, 2019.

[4] I. Bogachuk, V. Sokolov, V. Buriachok, Monitoring subsystem for wireless systems based on miniature spectrum analyzers, in: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology, 2018. https://doi.org/10.1109/infocommst.2018.8632151.

[5] LastPass Password generator. https://www.lastpass.com/ru/password-generator.

[6] The Diceware Passphrase Home Page. http://world.std.com/~reinhold/diceware.html.

[7] P. A. Grassi, et al., Digital Identity Guidelines Authentication and Lifecycle Management, 2017. https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret.

[8] E. B. Barker, J. M. Kelsey, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2015. https://www.nist.gov/publications/recommendation-random-number-generation-using-deterministic-random-bit-generators-2.

[9] D. Eastlake, J. Schiller, S. Crocker. Randomness Requirements for Security, 2005. https://www.semanticscholar.org/paper/Randomness-Requirements-for-Security-Eastlake-Schiller/c1a69463e79ebcfaf3ab4258b607894a5b335ede.

[10] Password Checker Online. http://password-checker.online-domain-tools.com.

[11] Kipchuk, F., et al. Investigation of Availability of Wireless Access Points based on Embedded Systems. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PICS&T), 2019. https://doi.org/10.1109/picst47496.2019.9061551

[12] Shevchenko, H., et al., Information security risk analysis SWOT, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 2923, 309–317, 2021.

[13] Kipchuk, F., et al., Assessing Approaches of IT Infrastructure Audit. In 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology, PICST, 2021. https://doi.org/10.1109/picst54195.2021.9772181

[14] Astapenya, V., et al., Analysis of Ways and Methods of Increasing the Availability of Information in Distributed Information Systems. In 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PICST), 2021. IEEE. https://doi.org/10.1109/picst54195.2021.9772161