Evaluation of the Probability of Breaking the Electronic Digital Signature Elements

Lakhno Valeriy, Sahun Andrii, Khaidurov Vladyslav, Panasko Elena, Chepynoha Anatolii, and Ustianovska Nataliia

Abstract This article analyzes the standards, algorithms, and hash functions used in electronic digital signature (EDS). It is determined that most of the modern hash functions and algorithms used in EDS schemes are based on elliptic curves above the field. Some of the collisions causes, methods, and algorithms for hash attacks are covered in the article. A mathematical apparatus for estimating the probability of hash functions breaking based on the "birthday paradox" is formed. The results of the probability of breaking for hash functions used in EDS were obtained. It is confirmed that in case of the same length of the hash, the probability of breaking the hash by the breaking the resistance to collisions method is much lower than using the method of breaking the strong resistance to collisions. It has been suggested that it is dangerous to add a key at the beginning or end of a message when working with key hash functions.

Keywords Birthday paradox · Hash function attacks · Birthday attack · Collision of the second kind

L. Valeriy · S. Andrii (✉)
National University of Life and Environmental Sciences of Ukraine, Kiev, Ukraine
e-mail: avd29@ukr.net

L. Valeriy
e-mail: lva964@nubip.edu.ua

K. Vladyslav
Institute of Engineering, Thermophysics of NAS of Ukraine, Kiev, Ukraine

P. Elena · C. Anatolii · U. Nataliia
Cherkasy State Technological University, Cherkasy, Ukraine
e-mail: lena.pa@ukr.net

C. Anatolii
e-mail: a.chepynoha@chdtu.edu.ua

U. Nataliia
e-mail: Nataustyanovska@ukr.net

P. Karrupusamy et al. (eds.), *Sustainable Communication Networks and Application*,
Lecture Notes on Data Engineering and Communications Technologies 93,
https://doi.org/10.1007/978-981-16-6605-6_48

1 Introduction

Today, electronic digital signatures (EDS) are a common technology for data integrity protection. Some of its mechanisms (hash functions) are very widely used in cryptocurrency generation systems. The presence of several different EDS schemes using different hash functions in application systems makes it necessary to assess the level of security of data that is protected using EDS. Many publications have studied the problem of crypto resistance or the probability of breaking various elements of EDS [1–7]. We can single out the study of the probability of collisions of the first or second kind in the generated values of hash functions [3–6]. To facilitate the assessment of the detection of collisions (the appearance of two identical values of hash functions) often use the paradox of birthdays [7, 8].

2 Steps of EDS Elements Reliability Evaluation

2.1 Estimation of the Probability of Break of EDS Elements

In some countries, including Canada, South Africa, the United States of America, Algeria, Turkey, India, Brazil, Indonesia, Mexico, Saudi Arabia, Uruguay, Switzerland, Chile, Russia, Ukraine, and the European Union, electronic signatures have legal significance. Therefore, the assessment of the probability of break of the EDS elements is of great practical importance.

At present, the stability of almost all modern EDS standards is based on the complexity of solving the problem of discrete logarithm in a group of points of elliptic curves [9]. Moreover, the standards for obtaining the main element of information security in EDS-hash functions are very similar. For example, now any EDS scheme must define at least three functional algorithms, in particular, the algorithm for generating a key pair for the signature and its verification, the signature algorithm, the signature verification algorithm.

Consider the EDS standards of Ukraine, Russia, and the United States of America. At present, the text of the specification DSTU 4145–2002 for the EDS algorithm exists in the form of a corresponding document [10, 11]. This algorithm is based on elliptic curves over the field F (2) (non-supersingular curves). The standard uses curves in polynomial and optimal normal bases. Moreover, the curves in the polynomial basis can be of the second and third types. The basis for the development of this standard is the international standard IEEE 1363a. DSTU 4145–2002 is authentic compared to IEEE 1363a, due to the use of its own hashing function. The interstate standard GOST R 34.311–95 (formerly GOST 28,147–89 in the mode of imitation insert) is used as an algorithm of the hash function. In the United States of America, there is a regulatory document FIPS 202 which describes the requirements for the hashing algorithm in EDS systems, whereas a hashing function it is recommended to

use "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions" [12].

If we compare the state standards for EDS algorithms in countries where there are legal requirements for algorithms and functions in EDS schemes, then, for example, in Ukraine, Russia and the United States of America, they are similar but differ only in some numerical parameters and details of key pair generation, calculation mechanism, and signature verification. Specifically, these three algorithms are variants of the El Gamal scheme [13].

Therefore, taking into account the similarity of the mathematical apparatus of cryptocurrencies when finding the basic in cryptographic algorithms EDS, it is possible to extrapolate the obtained values of estimates of the probability of breaking hash functions in EDS to the above schemes. Thus, in comparison with the existing methods considered in the [7–9, 12], the described approach allows us to support the possibility of using the existing mathematical apparatus for systems with an EDS.

2.2 Mathematical Apparatus for Estimating the Probability of Breakage

The birthday paradox is an imaginary paradoxical statement that the probability of coincidence of birthdays (dates) for at least two members of a group of 23 or more people exceeds 0.5. For 60 or more people, the probability of such a match exceeds 0.99, although the probability of 1.0, it theoretically reaches only when the group is at least 367 people. This statement may seem unobvious because the probability of coincidence of birthdays of two people on any day of the year ($1/365 = 0.0027$) multiplied by the number of people in the group of 23 gives only $23/365 = 0.063$. This reasoning is incorrect as the number of possible couples (253) significantly exceeds the number of people in the group. There is no logical contradiction in this, and the paradox is only the difference between intuitive perception and mathematical calculation. However, under the conditions of using this paradox in cryptanalysis, and especially if the attack is carried out on a set of known open messages, it allows to estimate the probability of breaking a hash function with a fairly high accuracy [14, 15]. This will allow a «birthday attack», in which random values are tried, until two are found that work.

«Birthday attack» is used in cryptanalysis as a method of cracking ciphers or finding collisions of hash functions. The essence of the method is to significantly reduce the number of transmitted hash functions arguments required to detect the collision, because if the hash function generates n—bit values, the number of random arguments of the hash function, which is likely to detect at least one hash function collision, is not 2^n, but only about $2^{n/2}$ [16].

It should be noted that these conditions of "birthday paradoxes" do not necessarily hold true in the real world. In particular, in the real world, people are not born with uniform randomness. There may be accurate statistics on the days people are born.

But this model is not an accurate reflection of the real world, its simplicity makes it much easier to analyze the problem. In particular, in cryptanalysis, this model has satisfactory accuracy. So, for example, this paradox can be applied to assess the cryptographic strength of various elements of the blockchain technology due to the fact that it also uses hash functions similar to those under consideration [17].

2.3 Methods and Algorithms for Hash Attacks

You can consider all possible underlying hash attacks related to value generation collisions:

(1) Frontal attack. Having a convolution of type $H(m_1)$ of the message m_1, the cryptanalyst must use the search method to find the message m_2 ($m_2 \neq m_1$), for which $H(m_1) = H(m_2)$. If this condition is met, the cryptanalyst may declare that the resulting convolution corresponds to the message m_2, but not to the message m_1. If the hash function outputs an n—bit string, then the complexity of this method of cryptanalysis is estimated as $O(2^n)$.
(2) Attack based on "birthday paradoxes." This paradox is used for many applied problems (including cryptanalysis problems). It is based on a well-known statistical problem—"birthday paradox."

Therefore, the general task of cryptanalysis can be reduced to the task of finding collisions, namely how many open messages the cryptanalyst needs to review to find messages with the same hashes [17].

The probability of encountering the same hashes for messages from two different sets containing, respectively, n_1 and n_2 plaintexts will be equal to:

$$P \approx 1 - e^{\frac{n_1 n_2}{l}} \tag{1}$$

If in expression (1) $n_1 = n_2 = 2^{l/2}$, then the probability of success of such an attack will be: $P \approx 1 - e^{-1} \approx 0.63$, and the complexity of the attack can be estimated as $2^{\frac{l}{2}+1}$. In order to determine the collision, it is necessary to generate two pseudo-random sets of messages so that there would be $2^{n/2}$ messages in each set, and then you need to find the corresponding hash values. Then, according to the "birthday paradox," the probability that among them, there will be a pair of messages with the same hash values will be more than 0.5. Such an attack requires a large amount of memory to store sets of plaintext and apply effective methods of sorting them [16].

The scenario of an attack on a hash function based on the "birthday paradox" where the attacker A wants to attack party B and sign a contract that is unfavorable for party B looks like this:

(1) Attacker A is preparing two versions of the contract—a "good" option M_1 and a "bad" option M_2. Then, he makes minor changes to each document, manipulating periods, commas, other punctuation marks, and spaces.

(2) Attacker A calculates the values of hash functions for all versions of the contract created by him, compares sets of hashes, and searches for identical pairs. If the length of the hash is 64 bits, then for this purpose 2^{32} pairs are enough. Attacker A selects a pair of messages for which the value of the hash function is similar.

(3) Attacker A selects the selected pair with similar hashes and passes the "bad" version of the contract M_2 for signature to party B. In this case, the attacker A can prove in court that party B signed an unfavorable contract knowingly.

Thus, for a given hash function H(m), the purpose of the attack is to find a collision of the second kind.

The reliability of EDS is determined by the resistance to cryptological attacks of its components, such as: (1) hash functions, (2) EDS algorithms.

The main task of EDS cryptanalysis is to assess the probability of breaking the hash function under the following attacks: (a) brute-force (rough selection); (b) selection of the hash value, grouped per affiliation.

2.4 Methods and Algorithms for Hash Attacks

Based on the cryptanalysis apparatus formulated above, we estimate the probability of breaking the hash function. To attack with this method (brute-force), we considered two methods:

(1) **The method of breakage resistance to collisions**. According to the known value of the hash function $H(M)$, the attacker creates another document $M^{'}$ such that $H\left(M^{'}\right) = H(M)$. An attacker tries to find two random messages M and $M^{'}$ such that $H\left(M^{'}\right) = H(M)$.

Suppose that a hash function is absolutely reliable and unidirectional, and there is no other method of breaking it than a complete search of its values. At the output of this function, we obtain a number whose bit size is m. Then, the number of output values of the function $H = 2^m$. Let us denote $P(n, k)$ the probability that for a specific X value, at least one Y_i value from the range of values $Y_1 \ldots Y_k$ equality $H(X) = H(Y)$ is satisfied. For one Y, the probability of $H(X) = H(Y)$ will be $1/n$. From here, it is easy to find that the probability of identity of the expression $H(X) \neq H(Y)$ will be $(1 - (1/n))$. If we generate k number of values, then the probability that none of them will be repeated is defined as the product of the probabilities, calculated as $\left(1 - (1/n)^k\right)$. Therefore, the probability of at least one match will be equal $P(n, k) = 1 - (1 - (1/n))$. For the case of absolute break, the probability of such a case is $P(n, k) = 1$.

Using the "birthday paradox" discussed above to estimate the probability of a collision in a hash function, we can estimate the probability of such a case as $P(n, k) = 0.5$, whence the value of the parameter $k = n/2 = 2^{(m-1)}$. Thus, it can be argued that the relative probability of a successful attack by the method

of breaking the resistance to collisions (finding such a hash generated for a given value of an open message) is achieved with $2^{(m-1)}$ random messages. Then, the probability of breaking the hash function is $1/2^{(m-1)}$ [14–16]. The obtained values of the hash function breakage probabilities for different n-bit message lengths (Table 1).

Function graph showing estimation of the probability of breaking the hash function by the method of burglary resistance to collisions we can see in Fig. 1.

(2) **The method of breaking the strict resistance to collisions**. Let $P(n, g)$ be the probability that in a set of g elements, each of which can acquire n number of values, there are at least two elements whose values are identical. Then, the value that must acquire g in order for the condition to be fulfilled:

Table 1 Estimation of the probability of breaking the hash function by the method of burglary resistance to collisions

Hash length, bits	Probability of breaking the hash function
64	1,0842021724855044340074528008699e-19
128	5,8774717541114375398436826861112e-39
192	3,1861838222649045540577607955354e-58
256	1,7272337110188889250772703725601e-77
384	5,0758836746312984465480491116611e-116
512	1,4916681462400413486581930630926e-154
1024	1,1125369292536006915451163586662e-308
2048	6,1886920947651565509603667399424e-617

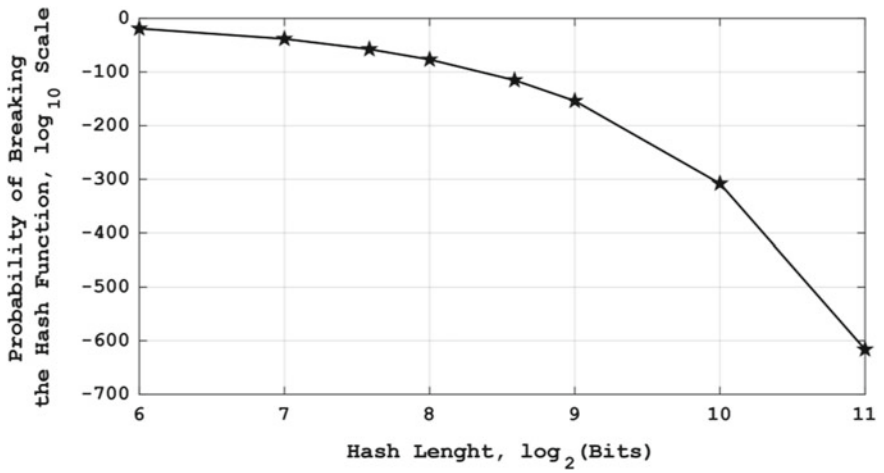

Fig. 1 Estimation of the probability of breaking the hash function by the method of burglary resistance to collisions

$$P(n, g) > 0.5 \tag{2}$$

In (2), condition for estimating the probability of meeting identical values is formulated. From expression (2), it can be understood that the number of ways to select elements such that there were no repetitions is $n!(n - g)!$ The total number of ways to select such elements is equal to n^g. The probability that the elements are not repeated is $n!/[(n - g)!n^g]$. Expression (2) can be written to determine the cases of probability of repetition:

$$P(n, g) = 1 - \left[n!/((n - r)!)n^g\right] \tag{3}$$

In (3), overall probability estimate for cases of repetition of several elements is formulated. If $P(n, g) = 0.5$, then $g = \sqrt{nln2} = ln2\sqrt{n} \approx n^{1/2}$. Let us assume that the length of the hash is m—bit, then the number of values that the hash acquires is equal to 2^m, then $g = \sqrt{2^m} = 2^{m/2}$. Therefore, to receive two messages with the same content (collision), it is necessary to calculate a hash from $2^{m/2}$ random open messages. The probability of breaking the hash function is estimated as $(1/2)^{m/2}$ [14–16]. Using expression (3), we can obtain the values of the probability of breaking the hash function at different values of the bit size of the obtained hash (Table 2).

From the obtained results of comparisons of hash function break probabilities, it is clear that under the same hash value length, the hash value break by collision resistance break method is much lower than with the use of strict collision resistance break method.

From the obtained results of comparisons of hash function break probabilities, it is clear that under (Fig. 2).

On the first (Fig. 1) and second (2) graphs, you can see serious differences in the probability of breaking hashes when using the first and second methods.

Suppose that for a given cryptographic hash function f, the aim of the attack is to find a collision of the second kind. To do this, it is need to calculate function f values for randomly selected blocks of input data until two blocks are found that have the same hash. Thus, if f has N different equally probable output values and

Table 2 Estimation of the probability of breaking the hash function for an attack by the method of strict resistance to collisions

Hash length, bits	Probability of breaking the hash function
64	0,00,000,000,023,283,064,365,386,962,890,625
128	5,421010862427522170037264004349 7e-20
192	1,262177448353618888658765704452 5e-29
256	2,938735877055718769921841343055 6e-39
384	1,593091911132452277028880397767 7e-58
512	8,636168555094444625386351862800 4e-78
1024	7,458340731200206743290965315462 9e-155
2048	5,562684646268003457725581793331e-309

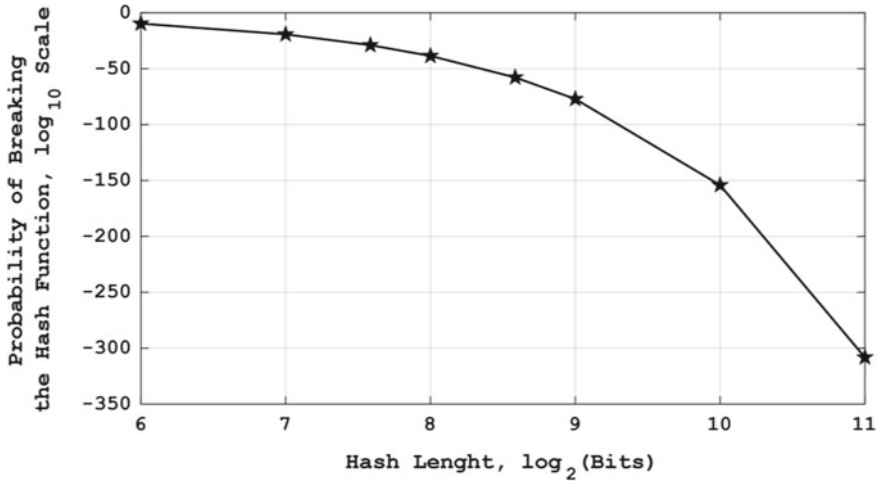

Fig. 2 Estimation of the probability of breaking the hash function for an attack by the method of strict resistance to collisions

N is large enough, then from the birthday paradox it follows that, on average, after enumerating $1{,}25\sqrt{N}$ different input values, the required collision will be found.

If the hash function generates an N—bit value, then the number of random input data for which the hash codes are likely to give a collision is not 2^N, but only about $2^{N/2}$. The light cells show the amount of input data at which a collision will occur with a given probability (analogy with a paradox, the number of different output chains is 365). This why there is no need to create data sets. Therefore, the goal of this research is fully achieved. Thus, without forming data samples for a given cryptographic hash function f, one can easily find a collision of the second kind. For this, it is sufficient to use only expression (3).

3 Conclusion

Estimation of probabilistic modeling demonstrates a systematic approach to cryptographic analysis of the method. The probabilistic approach makes it possible to assess the dependence of input and output data during encryption, as well as the selection of "bad" keys, which may not be reliable in practice. It can be concluded that to increase the resistance of the hash function in the EDS scheme to break at least to the level of 10^{-25}, hashing of at least 192-bit or even greater should be used.

When working with key hash functions, it is dangerous to add a key at the beginning or end of the message. This can be explained as follows:

- Let the key k be added at the beginning of the message, and the convolution function be constructed according to the Mercle-Dogmar scheme. Then, according

to the message M known to the cryptanalyst and its convolution function can be constructed.

- $H = H(k \| M)$, it is possible to determine the convolution value for all messages of the type $M \| M^{'}$, where after the message M any message $M^{'}$ is added. Indeed, due to the iterative nature of the hash function, there is no need to know the key to determine $H^{'} = H\left(k \| M \| M^{'}\right)$. It is enough to use the intermediate value of the convolution H when calculating;
- Let the key k added at the end of the message $H = H(M \| k)$. The collision value for the hash function (pair M_1, M_2) provided that $M_1 \neq M_2$, but $H(M_1) = H(M_2)$ can be calculated through the hashes $H(M_1, k) = H(M_2, k)$ for any k value. From this we can conclude that the complexity of changing the message M_1 is no longer estimated by the value of $O(2^n)$. Due to the use of the paradox of birthdays, the complexity will be significantly reduced and will be $O\left(2^{n/2}\right)$. In order to reduce the probability of successful cryptanalysis, it is recommended to write the key when calculating the convolution in the hash function several times. For example, as shown in an expression $H = (k \| y \| M \| k)$ that can be rewritten like $H = H(k \| y_1 \| H(k \| y_2 \| M))$ where y, y_1, y_2 is a complement of the key to a size multiple of the length of the hashing block.

Keyless functions are built on this principle. They are extremely reliable for attacks of collisions of the first and second kind, but their disadvantage is the large length of the resulting convolution, which significantly complicates their use.

For example, the standards of the Russian Federation and Ukraine have not only one, but several variants of hashes, because the cipher parameter is a set of replacement nodes. In this case, we get our own hash for each set. This nuance has the advantage of cryptanalysis complication, but creates compatibility problems, because each of these algorithms determines the step hashing function, which receives two blocks of data at the input:

- the current hash value from the previous step;
- another fragment of the input data array.

Thus, for a 128-bit hash function, the breakage probability differs for both methods by 9,223,372,036,854,775,808 times. You can also estimate the machine time required to break by the first and second methods. For example, the probable time to find the collision by the first method, provided that the hash has dimension m = 128 it is necessary to calculate $2^{128} = 3,4 \cdot 10^{38}$ variants. Then using the «birthday paradox», the number of these options will be $2^{m/2} = 1,8446744073709551616 \cdot 10^{19}$ by the second method and $2^{m-1} = 1,7 \cdot 10^{38}$ by the first method.

The results obtained in this work can find practical application in solving a number of problems of modern cryptography, for example, in evaluating the effectiveness of parallel implementation of methods for determining multicollisions of hash functions, as well as methods of discrete logarithm, similar to the Pollard ρ-method [18].

References

1. X. Wang, D. Feng, X. Lai, H. Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. CRYPTO National Center for Biotechnology Information (2006). https://eprint.iacr.org/2006/381.pdf
2. T. Okamoto, M. Tada, A. Miyaji, Efficient 'on the fly' signature schemes based on integer factoring. in *Proceedings of the 2nd International Conference on Cryptology in India*, INDOCRYPT'01, LNCS 2247 (2001) pp. 275–286
3. A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems. in *Advances in CryptologyCRYPTO'86*, LNCS 263 (1986) pp. 186–194
4. National Institute of Standards.: FIPS 180–1, Secure hash standard, NIST. US Department of Commerce, Washington D.C., April (1995)
5. P.R. Kasselman, W.T. Penzhorn, Cryptananlysis od reduced version of HAVAL. Electron. Lett. **36**(1) (2000)
6. A.V. Sahun, V.A. Lakhno, P.Y. Kravchuk, S.S. Kosenko, E.M. Kisiliuk, Elliptic curves in modern cryptographic systems. Int. J. Adv. Trends Comput. Sci. Eng. **9**(4). 5949–5955 (2020). http://www.warse.org/IJATCSE/static/pdf/file/ijatcse259942020.pdf
7. M. Bellare, T. Kohno, Hash function balance and its impact on birthday attacks. EUROCRYPT 2004, LNCS 3027 (2004) pp. 401–418
8. D. Coppersmith, Another birthday attack. in *Proceeding CRYPTO '85. Advances in Cryptology*, (1985) pp. 14–17
9. P. Shor, Algorithms for quantum computation: discrete logarithms and factoring. Foundations of Computer Science. in *IEEE 1994 Proceedings. 35th Annual Symposium* (1994) pp. 124–134. ISBN 0-8186-6580-7. https://doi.org/10.1109/SFCS.1994.365700
10. National Ukrainian standard of digital signature based on elliptic curves (DSTU 4145-2002), https://sourceforge.net/projects/dstu4145-2002/
11. GOST 28147-89, Sistemy obrabotki informaczii. Zashhita kriptograficheskaya. Algoritm kriptograficheskogo preobrazovaniya. https://web.archive.org/web/20120226123825/http:/gostshif.narod.ru/
12. National Institute of Standards, FIPS 180–4, Secure Hash Standard (SHS). https://csrc.nist.gov/publications/detail/fips/180/4/final
13. E. Taher, A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074
14. M.C. Borja, J. Haigh, The birthday problem. Significance. Royal Stat. Soc. **4**(3), 124–127 (2007). https://doi.org/10.1111/j.1740-9713.2007.00246.x
15. F.H. Mathis, A generalized birthday problem. SIAM Rev. **33**(2), 265–270 (1990). https://doi.org/10.1137/1033051.ISSN0036-1445.JSTOR2031144.OCLC37699182
16. A.D Kozhukhivs'kij, A.V. Sahun, O.A. Kozhukhivs'ka, O.Y. Snisarenko, Y.V. Stepanecz, Analiz kriptostijkosti elementiv elektronnogo czifrovogo pidpisu. Visnik ChDTU № 3, (2013) pp. 80–85. ISSN 2306–4455
17. V.V. Khilenko, Application of blockchain technologies for improving the quality of ACS of complex dynamic systems. Cybern Syst Anal **56**, 181–186 (2020). https://doi.org/10.1007/s10559-020-00233-w
18. J.M. Pollard, B.J. Green, Theorems on factorization and primality testing. in *Mathematical Proceedings of the Cambridge Philosophical Society*. vol. 76(3), (Cambridge University Press, 1974) pp. 521-528. ISSN 0305-0041; 1469–8064. https://doi.org/10.1017/S0305004100049252