Technical sciences

UDC 004.04

Koriakov Ihor

Head of Development Department

Scientific Innovation Firm Crypton LLC

CRYPTOPOLEMIADE

Summary. *This paper is polemical in character and considers some issues in cryptography (symmetric block ciphers) from a perspective other than a conventional one. The author has attempted to apply Altschuller's methods to cryptography (recently, it has become overburdened with mathematics and become an authoritarian science) and apologizes to professionals with a fundamental education in cryptology for using an engineer's lingo.*

Key words: *cryptography, symmetric block ciphers, random sequence generators, dependency probabilities.*

Introduction. The author bows to the creative work of the great inventor Genrikh Saulovich Altschuller who attempted to formalize the process of invention. His book "Algorithm of Invention" and respective tables, with a strict statement of the primary problem, helped obtain a solution in the form of a semiabstract recommendation of the type: "to reduce an ice-breaker's energy consumption, the requirement is that the ship section between the lower edge of the ice and its upper edge equal zero."

This paper is polemical in character and considers some issues in cryptography (symmetric block ciphers) from a perspective other than a conventional one. The author has attempted to apply Altschuller's methods to cryptography (recently, it has become overburdened with mathematics and become an authoritarian science) and apologizes to professionals with a fundamental education in cryptology for using an engineer's lingo.

Bad keys

Distrust in pure randomness

It is common in the world literature on cryptography to classify keys as "good" and "bad" ones. For instance, a key consisting exclusively of zeros, of course, is a bad one.

Imagine a one-time pad consisting exclusively of zeros. That would be a total disaster – such a key is inadmissible!

Theorem 6 in Claude Shannon's paper "Communication Theory of Secrecy Systems" defines the complete security of perfect secrecy systems because the key is an exclusively random sequence.

The requisite and sufficient condition of perfect secrecy is that conditional probability $P_M(E)$ of cryptogram E during transmission of message M should be equal to probability $P(E)$ of receiving cryptogram E for all M and E. This condition determines ciphers with a key, which is a truly random sequence with a length equal to (or greater) than that of the message.

Can the key of the one-time pad type take the value "0" in all digits? Yes, with an appropriate probability if it is a perfect secrecy system according to C. Shannon. All the combinations in the key sequence are possible and equally probable. Can any sections in the sequence of a one-time pad be removed for the pad to look more random? No, they cannot because Theorem 6 will not hold.

How can we overcome this contradiction? Assume that we are ciphering a classified telegram of special importance about the time of launching an assault by our army (a one-time pad is chosen as a perfect secrecy cipher) and the plaintext of the time of launching the assault, for instance, "5:30" is associated with ciphering sequence values equal to zero. Modulo 2 zero with any value is equal to this value and the cryptogram intended for transmission over an open channel shows plaintext "5:30" completely insecure!

This is all there is to the hype about "perfect secrecy systems"! With high probability, they will pass the plaintext to the output! Moreover, in any

ciphertext of any cipher half of the bits will match the plaintext! This is identical to a plaintext with a 0.5 probability!

Yes, all this is true.

Nevertheless, Theorem 6 remains operational and no one ever, without knowing the key, will be able to restore the plaintext. "Even when aliens from the Andromeda constellation will land on our planet in their huge spaceships with computers of incredible computing power they will fail to read the messages of Soviet agents ciphered with one-time pads" [1].

Next, we will try to look into the problem of bad keys from the angle of perfect secrecy systems and conclude that there is no contradiction and maybe there is no point in dealing with it.

Anthropomorphism in relations with randomness

Wikipedia: "The paradox of regularity – an observation consisting in that the majority of people, in seeing an obvious regularity in the outcome of a series of experiments, will be inclined to believe that the experiments were not random because they see the occurrence of a definite sequence in random experiments as a highly unlikely event. The human brain is specialized in identifying regularities in the environment with a view to using them to increase the chances of survival. Hence, there is nothing strange in that a person can see a perfect random sequence as not a random one."

The author acknowledges that he himself was trapped by the human attitude to randomness. For instance, a certified generator is generating huge arrays of random sequences and the author, upon opening a file with random numbers, o horror, sees its beginning:

1D BB BB C1 FD 93 05 27 B2 32 66 E9 99 0A 0D DA

It immediately catches the eye that half-byte "B" is repeated four times in a row; "1D" and "C1" are mirror reflection transposed and differ only by one bit. "FD 93 05 27" are so far, so good, whereas bytes "B2 32" also differ only by their most significant bit, and next we see "66 E9 99" – two sixes in a row and

almost next to them, three nines in a row. The end is sheer mockery: "0A 0D" differ only by one bit and in the last byte they are repeated as "DA". Besides, "0A 0D" are "Line feed" and "Carriage return" symbols at the end of each line in a text file! Can such a sequence be random?!

This file with all its "low-randomness" fragments passed the most stringent randomness tests, as all other files generated by this generator also did.

The second example concerns the ignorance of the fact that the force of statistical laws is very powerful, and in a sense, is invanquishable. In thermodynamics, this force is implied. We understand well that the temperature of water in a glass, representing the total kinetic energy of molecules, cannot randomly change by five degrees, and that it will be very stable. Our thermometers will have a measurement error much greater than the dispersion of this total energy.

However, when statistical properties of real objects are considered, a person can easily be mistaken in one's estimates. An unexpected result occurs in a simple example with a lump of matter comprising ten molecules randomly moving in one direction. We will assume that the lump of matter is flying if all ten molecules are moving in a direction limited by a sector of 10º. The number of directions of motion of each molecule is $(360/10)^2$ = 1,296 and if the directions are independent the probability of coincidence of directions of motions of all molecules is $(1/1296)^9 = 10^{-29}$. If the directions change each nanosecond, the mean time waiting for particle flight will be 10^{29} / 10^9 / 3,600 / 24 / 366, i.e. about 3,000 billion years. The result seems to be implausible: only ten molecules, and such a statistical power, which prevents the flying of even such a minute particle!

Or, for instance, the author, having a system of dozens of thousands of hardwired automatons that perform the sorting of 2^{27} states each, wrestled with the problem of how to use the resource more effectively because the automaton can terminate sorting at the first step and at the last one. In practice it turned out

that no optimization is needed because the automatons finish their sorting virtually at the same time: with an interval of 600 seconds, with the deviation of the time of termination being a fraction of a millisecond!

Wikipedia: "The general meaning of the law of large numbers – the joint action of a large number of identical and independent random factors yields a result ultimately independent of the event."

Pure randomness holds huge power and it would be unwise to disregard it.

The basic methodological error

Historically, random sequence generators used for generating keys had statistical characteristics of generated sequences, which were far from being perfect. Therefore, the sequences that they generated required thorough rejection by strict statistical criteria. This means that the hypothesis: "at the instance of generating the given sequence section, the generator temporarily became perfectly random" was tested.

Moreover, only sequences with a length equal to that of the key were considered, and the criteria were defined from this length. For instance, the admissible number of units or zeroes in a sequence of specified length N was tested based on confidence probability β as follows [4]:

$$N\left(p-\arg\Phi^{*}\left(\frac{1+\beta}{2}\right)\sqrt{\frac{pq}{N}}\right)<Np^{*}<N\left(p+\arg\Phi^{*}\left(\frac{1+\beta}{2}\right)\sqrt{\frac{pq}{N}}\right), \quad (1)$$

where p and p^{*} are, respectively, probability and its estimate; $q = 1 - p$.

For $N = 256$ and $\beta = 0.99$, the number of units/zeroes should be in the interval 112 to 144.

In this case, the unreliability of such estimates is that 1% of truly random sequences will fail the test and, most horribly, no one knows what percent of non-random sequences will pass it.

By increasing confidence probability β, we "open the doors" by admitting a greater number of non-random sequences. By decreasing it, we introduce a more stringent rejection, i.e., the filtering of sequences, thus making them the

more non-random. In any event, we reduce the probability of that the multitude of such sequences has been generated by the generator of Truly Random Sequences (TRS).

Classical works on random number generators [1; 2; 3] consider the null hypothesis test (H_0). It states that a random sequence generator generates truly random sequences.

However, it would be more proper to test the generator than rather the sequences.

We have no right to substitute the null hypothesis test with regard to the TRS generator with a similar test of a sequence fragment, which we want to use as a key. With such a substitution, a paradoxical process will occur: for some fragments, we will receive a confirmation of the null hypothesis, and for some we will not. This implies that our generator will be recognized as either truly random or not. However, the null hypothesis test for the generator will be confirmed until the TRS generator will remain so.

In view of modern outlooks [5], the TRS generator shall be considered to be metrological equipment. It is just as good as an oscilloscope or an exact time or frequency generator is. The TRS generator, as an instrument, should be certified and validated, and during its life cycle, the mechanism of confirming the null hypothesis shall be in force. The same paper gives a description of seven levels of testing a generator to confirm the null hypothesis starting from the stage of its design and to the process of generating TRS.

If the null hypothesis is not confirmed, the TRS generator should be immediately stopped, and its generated sequences should not be used if possible.

However, if the TRS generator is serviceable and the null hypothesis is confirmed, we are obliged to use its output sequences without any filtering.

This cancels the above-described contradiction, which occurs when obsolete techniques of generating random sequences used for generating keys and random parameters of crypt algorithms are used.

At the same time, tests of type (1) are performed of course, though not to test sequences, but rather to obtain an estimate of the probability of confirming the null hypothesis for the generator.

Examples of bad keys

A key consisting of all zeros or units, undoubtedly, reduces encryption algorithm security. If we assume such keys to be bad ones, then why do we have to presume admissible such keys as 01010101010... or 0000111100001111...? The list of suspicious keys can be very big and, no matter what, we will always miss something. So what should be done?

Apart from keys consisting only of zeros or units, known, for instance, are the keys of the GOST 28147-89 algorithm, in which $X_0 = X_7$, $X_1 = X_6$, $X_2 = X_5$, $X_3 = X_4$. All these keys have been acknowledged as weak keys, which radically reduce the algorithm security. Actually, the number of such keys is improbably large: 2^{128} ($3*10^{38}$).

Let us look into how often we will come across such keys. Let us have 100 billion key generators operating with the speed of 100 billion keys per second each. How often will we encounter such weak keys?

The answer is rather simple: $3*10^{38}$ / 10^{11} / 10^{11} / 3,600 / 24 / 366 = 1,076,080,100. During this time (1 billion years), we will obtain the first weak key with the probability of 0.63.

With keys containing only zeros or units, the case is worse – the probability of their occurrence is $1/2^{255}$.

"If you choose a key randomly, the probability of selecting a weak key is negligibly low. If you are a real paranoiac, you can test the generated key for "weakness". Some people believe that there is no point in worrying. Other argue that a test is easy, so why not try it" [1].

On secret parameters

As concerns the parameters of cryptographic transformations, irrespective of whether they are secret key elements or not, the problem is solved typically:

first, it is postulated that these parameters or transformations should be as close as possible to random ones. Next, a formal procedure, far from a random one, is specified for their generation, which imposes essential constraints on their possible values.

The authors of such techniques are trapped immediately by a paradox – they want to make something more random by constraining randomness. Since this yields nothing clear and provable, they introduce certain subjective parameters whose choice should remedy the situation and save some randomness for the parameters. Here is a classic example [7]:

"...as applied to substitutions of order *n*, we introduce the notion of a random (quasirandom) substitution meant to be a substitution satisfying three criteria of randomness, including the realisation of the following properties:

Property 1. The number of inversions η_n in the substitute of order *n* satisfies condition

$$\left|\eta_n - \frac{n(n-1)}{4}\right| \le \alpha\sigma_\eta, \sigma_\eta = \frac{n^{3/2}}{6}.$$

Property 2. The number of cycles ξ_n in the substitute of order *n* satisfies condition

$$\left|\xi_n - \ln n\right| \le \alpha\sigma_\xi, \sigma_\xi = \sqrt{\ln n}.$$

Property 3. The number of ascendancies θ_n in the substitute of order *n* satisfies condition

$$\left|\theta_n - \frac{n}{2}\right| \le \alpha\sigma_\theta, \sigma_\theta = \sqrt{n/12}.$$

In these relationships, α is a parameter chosen substantially based on subjective considerations (at least on the condition that the multitude of admissible substitutions will not be very limited)."

Just imagine – rigorous mathematics and subjective considerations! Here, many authors admit that the attempt to improve an indicator of tolerance to a

certain kind of analysis by attributing some specific properties to parameters or functions will always degrade the indicators of tolerance to other kinds of analysis.

Key filtering

An excerpt from GOST 28147-89:

"Keys that define the filling in of the key memory (KZU) and the tables of substitution block K are secret elements and are provided to an established procedure."

This means that both the main 256-bit keys and the substitution block tables are a key, the sole requirement to which is the equal probability selection from a set of keys. If for the main key the requirement that the probability value of all bits is 0.5 and independence is natural and indisputable, a strange dualism is observed for the substitution block tables.

On the one hand, the content of these tables should be close to equal probability and unpredictability, and on the other hand, it should be predefined according to certain formal rules.

The need to generate GOST substitution nodes as permutations of values 0-15 in 16 positions (bijectivity) is usually taken as an axiom. However, this is needed only in the case of reversibility of transformations, which for the Feistel network, the basis of the GOST algorithm, is not obligatory.

Andrey Vinokurov: "Everything works even in the case when the substitution node has duplicate elements and the substitution defined by such a node is irreversible; however, cipher security in this case is reduced. Why this is so is not discussed in this paper. However, it is easy to ascertain the fact. For this end will suffice, by using the demo program for enciphering data files, which is appended to the paper, to encipher and then decipher the data file by using for this procedure the "inadequate" substitution table whose nodes contain duplicate values."

If we generate a block substitution table with no constraints, simply as

random values of each element within 0-15, then the set of tables will be very large: 2^{512}. This is really very large. Whatever the "weak" contents of these tables, we will be unable to indicate an effective method of predicting the transformation after 32 rounds because their contents will be random.

Imposing constraints on bijectivity reduces the power of the set of substitution tables to 2^{354}, but this anyway yields preposterously impossible probabilities. Even if we reduce a virtual distance and demand that the eight tables are ultimately far from each other, and prove that these are the rows of a Latin square, and try to evaluate the number of possible variants, we will be surprised to find that to the square order of 12 this number is still known, and for the square order of 16, it is unknown. The world mathematical community, let alone us, simply does not know how many variants exist. By the way, the same authors [7] have concluded that security against different attacks is determined, primarily, by the width and number of cipher rounds, rather than by a special content of the substitution tables, which can be simply random.

I reiterate once more: whatever the criteria we invent, which reduce the size of the set of admissible keys, the power of this set in practical cases is so large that excluding from it or adding to it a subset of inadmissible keys DOES NOT AFFECT the power of the set. This means that it does not matter whether inadmissible keys are present in this set or not.

For instance, returning to the example of "bad" GOST keys, let us do a computation on a powerful calculator:

Total keys 2^{256} = 1.157920892373161954235709850086910652e+77

and the remaining part without "bad" ones: $2^{256} - 2^{128}$ = 1.157920892373161954235709850086910652e+77.

As the saying goes, "find ten differences."

Conclusions

Of course, the constraints that transform a random sequence to a non-random one are inadmissible in no way. For instance, the author knows the

technique of "biting out" from the ciphering sequence of a series of zeros with a more defined length to prevent sufficiently long sequences of plaintext symbols from penetrating into the cipher text. But such a sequence fails the tests of NIST Special Publication 800-22 [9]. It cannot be considered random and is unusable as a ciphering sequence.

If the space of keys is homogeneous and its size is sufficient for excluding a bruteforce key search, then there is no need to classify keys into good and bad ones because the laws of statistics are very rigid and will protect the cipher securely without the need to find "secure" keys. Looking among keys only for good ones against specific criteria is a loss of time at best, and in the worst case, it means acting in favour of the opposition and reducing the search domain when the cipher is broken because any rule of filtering a random sequence makes it non-random.

Keys should be generated by using exclusively generators of truly random sequences [5]. Filtering a random sequence that affects its statistics should be prohibited even if we believe that the sequence is not completely random or is unfit to be a key by certain criteria.

Let us not forget that the grandmas in Bletchley Park picked letters from a lottery drum and wrote them down in a pad (Cryptonomicon). When they "felt" that the letter was not random, they returned the letter to the drum and did not write it down. This made a hole in randomness, which the enemy managed to exploit. World fiction, or course [6]... But.

Ciphering modes

The single mode

In the book "Practical Cryptography" [8], Ferguson and Schneier in paragraph 5.7 "Which mode should I use" conclude clearly on the expediency of using only one ciphering mode – running key ciphering (counter mode, CTR):

"In our opinion, the advantages of CTR are more than adequate to prefer this mode apart from those situations when you cannot control the mode of

applying the encryption function. On the one hand, we have mentioned that each part of the system should ensure its own security and be independent of its other parts. On the other hand, we recommend using the CTR mode, the uniqueness of occasions of which is ensured by other parts of the system."

It bears reminding that, in the counter mode, the first to be encrypted is the block called the initialization vector, the synchrosignal or occasion, with the encrypted result being input to the counter, which is incremented and encrypted to obtain each successive ciphering sequence block. Encryption and decryption is performed identically: modulo 2 addition of the source text with the ciphering sequence.

What is good in the CTR mode and what are its constraints?

Advantages:

- Cryptosecurity of enciphering in the running key ciphering mode is defined strictly only by block cipher security.
- The message need not be complemented to a complete block.
- The CTR mode allows for parallel computation of any amounts of ciphering sequence blocks. Due to this, CTR realizations can achieve higher speeds up to one clock cycle per block with hardware pipelining.
- It suffices to realize only the function of encrypting the block cipher. Moreover, the function can be an irreversible one.
- The CTR mode ensures inequality of the plaintext for each pair of blocks of encrypted text.
- In the event of collisions, information leakage in the CTR mode is less than in other modes.

Features and constraints:

- It is required to ensure the uniqueness of each initialization vector when encrypting with the same key.

– The paradox of the birthday problem. If the block size is 128 bits, one can expect the first repetition of the encrypted text block by encrypting about 2^{64} blocks. Virtually, one key can ensure the secure encryption of 2^{64} bytes.

Optional occasion

A situation exists when a key is exclusively a one-time thing, i.e. each message has its own key. In classical secret communication systems, this is the All-points or Individual communication.

In this case, an initialization vector value can be used, which is the same for all messages (e.g., "12345..." or even "0") and, hence, not transmit it with the cryptogram.

For instance the ECB mode, requiring no initialization vector, is recommended for encrypting key information with no distinct statistics. Instead, the CTR mode can be used with a fixed initialization vector if the entire array of keys is encrypted with one master key.

In summarizing, all encryption modes can be replaced with success by the CTR mode.

Integrity

Modification attack

If the opposition knows the message structure, it can potentially modify a certain fragment of the message, having combined the cipher text of the fragment with a definite code.

Modification attacks can be countered using the following methods.

1. Computing a cryptographically strong message authentication code or a hash function of the plaintext or encrypted text and transmitting this value in the cryptogram.

2. Text shift: a plaintext is shifted cyclically to a random number of positions, and this number is transmitted in the encrypted message heading.

3. Archiving with plaintext compression, with integrity control being

performed by the archiving procedure. In spite of the cryptographically insecure integrity control with the help of CRC, modification attack defence is ensured by the compressed text elements being defined by the set of plaintext elements. In order to modify a plaintext by operations with the encrypted archive, it is necessary to know the plaintext or the archived text. But the opposition, fortunately, does not know it.

Archiving with compression

Generally, it is desirable to make the archiving procedure obligatory and automatic with running key ciphering. That provides the following benefits:

- Protection from modification;
- Exclusion of statistics from the encrypted text;
- Reducing the size of the encrypted text.

Key encryption

Classical scheme

The key generation centre (KGC) usually contains the TRS generator and means for writing keys to key carriers (CC). The KGC, for subscribers of a secret communication network, generates sets of keys, which are encrypted with the master key (MK).

The MK is entered into each encryptor in the network and each encryptor generates its own random mask (RM), which is modulo 2 added with the MK. The masked MKs are secure and stored in encryptors, whereas unique RMs for each encryptor are stored in a separate carrier readily removed in case of a compromise and, probably, are destroyed.

The keys are used as follows.

1. The RM mask is read.
2. The RM is modulo 2 added with the masked MK.
3. Each used encrypted key is decrypted with the MK.
4. After application, all open keys in the encryptor memory are destroyed.

In spite of the key encryption system, KGC is a risky object, which handles sets of unencrypted keys of the entire classified network and requires expensive measures to prevent leakage of key information.

Virtual encryption of keys

Since the values of the encrypted and unencrypted one-time keys are undistinguishable from a random sequence, the keys need not be first generated, encrypted with a master key, distributed, and then decrypted at application points.

Keys can be not encrypted, but rather by generating random sequences in the KGC, these keys can be assumed as already encrypted.

In application, these conventionally (virtually) encrypted keys are decrypted with the master key.

The KGC becomes a relatively secure object because it works only with (virtually) encrypted keys. Open secret keys appear only inside the encryptor and are removed after application.

Here is yet another interesting property: in case of a compomised master key and its substitution with a duplicate master key in all encryptors, there is no need to re-encrypt virtually encrypted keys. We will simply nod our head and believe that the available keys were virtually encrypted with a new master key.

Key expansion

Classical scheme

Key expansion procedures with time become even more complicated and sophisticated, and they approach the algorithms of cryptographic transformations. If in "Magma" (the old, prior to 1989 name of the GOST 28147-89 algorithm) the subkeys were simply parts of a key used in the direct and reverse order, in "Kalina", subkeys are generated by using the primitives of the encryption algorithm proper. This is done to protect from even more advanced attacks, which in reality are even less probable, but are not excluded from consideration.

Why does the key expansion problem appear? This is because, as a rule, the number of cipher rounds multiplied by the length of the round key significantly exceeds the required key length. For instance, the stated key length is 256 bits, and in each of the 10 rounds, 128 bits of key information take part. Thus, the total length of subkeys is 1,280 bits.

The second reason: possible weaknesses of the encryption algorithm in case of a one-time application of key elements. It is assumed that if the key elements are used several times in different rounds and interact with different elements of the cipher state, then this is good.

Preliminary expansion

Currently, memory currently is widely available, and there is no point in saving on it. What is the difference what to store, a 256-bit key or 1,280 bits of preliminarily generated subkeys?

Moreover, there are critical applications demanding frequent and rapid key changes. In such applications, the even more involved key expansion procedures can reduce the cipher speed dramatically.

Virtual expansion

Let us refer to the great Altschuller for advice: how to solve the key expansion problem most effectively? The answer is simple: no need to solve it.

Since ideal expansion implies the mutual independence of subkeys and their generation from a key represented by a random sequence, then not all the properties of the set of subkeys should differ from the properties of a random sequence.

So what then? Since during preliminary key expansion we refused to transport and store the key proper, but operate only with expanded subkeys, then nothing stops us from not generating a random sequence, declaring it a key and generating subkeys from it. Conversely, we would rather directly generate a set of subkeys as a random sequence. Let us call such an operation "virtual key expansion".

If the subkeys will be as random as the key itself, then we can conventionally say, "Yes, the effective length of the key is 256 bits and it is used to generate subkeys with a length of 1,280 bits by a virtual algorithm, which cannot be broken. Hence, making it impossible to attack a key by the values of subkeys." Since modern expansion algorithms prevent restoring the source key by the values of the subkeys, then without knowing the key it will be impossible to determine whether the keys were expanded from a source key with the expansion procedure or whether this is a random sequence – virtually expanded subkeys.

Complete virtualisation

This is the last step: generating virtual keys with their virtual expansion.

If earlier the KGC had generated sets of keys in the form of random 256-bit sequences and encrypted them with a master key, and then sent them to the subscribers of a secret network, our new virtualised KGC will generate sets of virtually encrypted and virtually expanded keys in the form of random 1,280-bit sequences and, without encrypting, will send them to subscribers.

The benefits are ample: the KGC does not work with the secret representation of the key. When the keys are used, they need no expansion and, finally, these are ideal subkeys because they were obtained not by a deterministic expansion procedure, but they rather are perfect randomness.

Violation of the law of large numbers

Estimating the dependency probabilities

Known is an estimate of the degree of closeness of a substitution to a random substitution (RS) by the parameters of the Strict Avalanche Criterion (SAC) [13], i.e. the estimated probability $k_{SAC}(i,j)$ of change of output bit j with a change of input bit i, which determines the degree of dependence of change of substitution outputs on the change of its inputs (dependency probabilities).

Known is the iterated logarithm law (a variant of the law of large numbers), which, simply speaking, states that the deviation of the sum of

identically distributed independent random values divided by n does not exceed $\sqrt{(2\ln\ln n)/n}$.

The estimate of the dependency probabilities for a true RS tends to 0.5 with growing number of trials n. However, due to RS finiteness, eventually, the hidden regularities will begin revealing themselves as shifts in estimates of dependency probabilities that violate the law of large numbers. Hence, with a definite n, the estimate deviation from 0.5 will exceed $\sqrt{(2\ln\ln n)/n}$.

If we take a certain substitution, normalize the estimate deviation from the ideal value to $\sqrt{(2\ln\ln n)/n}$ and build a graph of the dependence of deviation on n, we will obtain certain curves that remain within the limits of two parallel boundaries with the values ±1.

The tassel of estimates

Fig. 1 shows the graph of estimates $k_{SAC}(i,j)$ for a "pure" 16×16 bit RS (width B = 16). In the graph, the black curves represent 32 estimates $k_{SAC}(i,j)$ – 0.5 for all i and all j normalized to $\sqrt{(2\ln\ln n)/n}$. Respective boundaries ±1 are designated with green lines, and the red marks show the values of n in the logarithmic scale.

As was found experimentally, for $n > 2^B$, the estimates start diverging and cross the boundary $\sqrt{(2\ln\ln n)/n}$. The result is a "tassel" of estimates with a rapidly expanding tail. Boundary $n = 2^B$ (the vertical green line) is called conventionally the "boundary of indistinguishability" of the substitute: if our substitution yields estimates remaining within the boundaries of the iterated logarithm and approach the indistinguishability boundary, we assume that differences of the substitute from RS are not found.

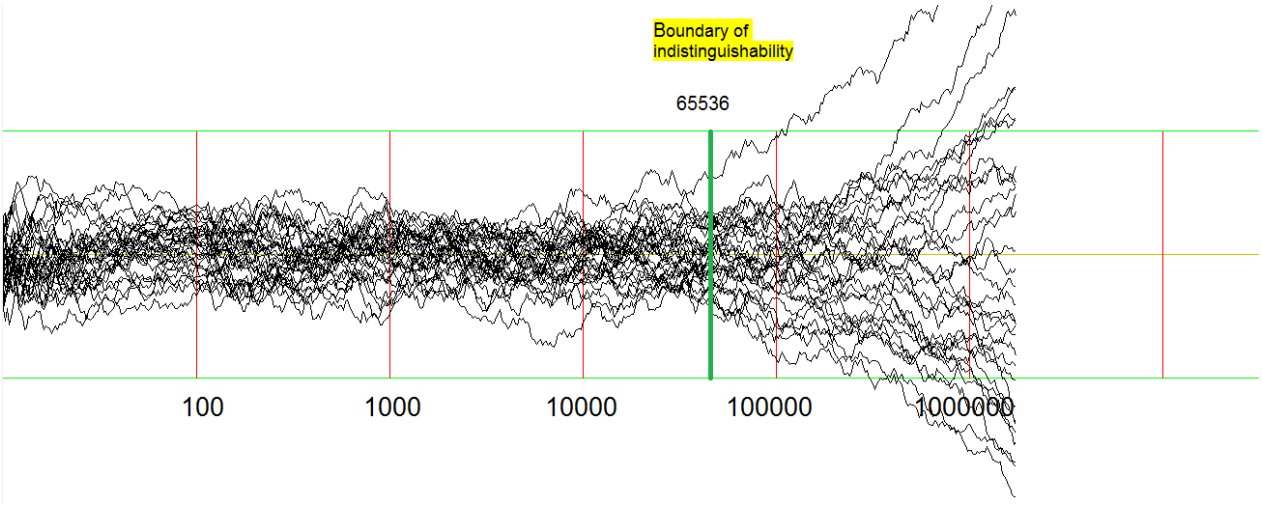
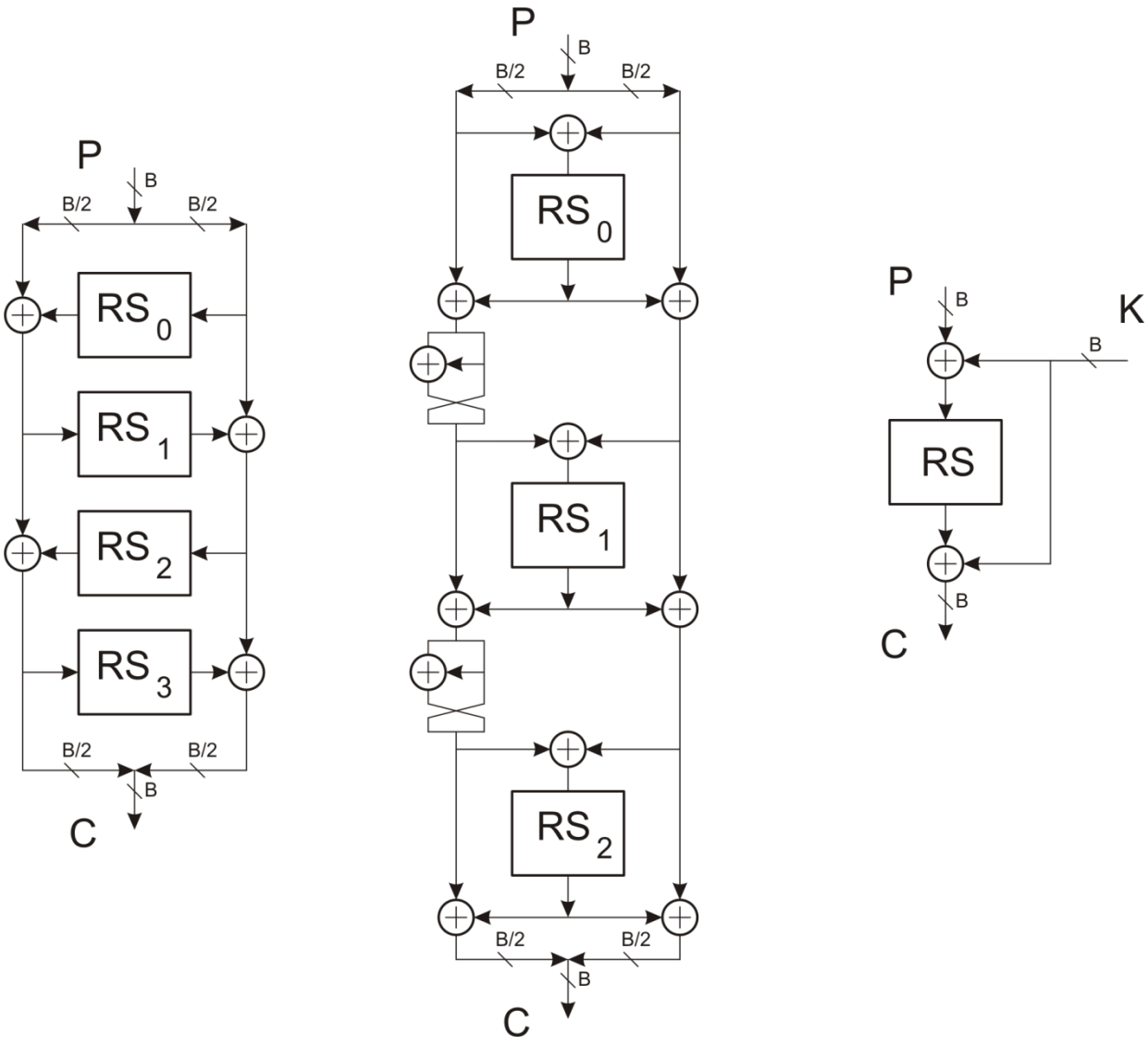


Fig. 1. Normalized estimates $k_{SAC}(i,j)$ for a "pure" random substitution 16×16

Decomposition of a random substitution

Luby-Rackoff

A random substitution (RS) of sufficient width is considered an ideal of a block cipher. Naturally, for instance, it cannot be implemented in practice in any way even for $B = 128$. A random substitution with preservation of cryptographic security can be replaced with a "strong pseudorandom permutation" (SPP) according to Luby-Rackoff [10]. It is a 4-round Feistel network that performs operations with the left-hand and right-hand parts of input word P with a $B/2$ length. The crypto secure round functions for such a network can be four different RS with a half-width $B/2$ as compared with that of the primary B. Fig. 2*a* is a schematic representation of such a permutation.



a *b* *c*

Fig. 2. Luby-Rackoff (*a*), Lai-Massey (*b*) and Even-Mansour (*c*) structures

A strong pseudorandom permutation is convincingly secure against attacks (adaptively) by selected input and output texts (virtually all known attacks are reduced to such ones).

But why do we need to stop? Each of these four RS can also be represented as a strong pseudorandom permutation with round functions with a width of *B*/4. In continuing, we will substitute each RS in the upper level with four RS in the bottom level with a half-width.

Then we will see that RS with the width $B = 128$ were substituted with 256 RS with a width $B = 8$. Hence, we can easily generate 256 truly random 8 ×

8 bit (256 bytes) substitutions. The total size of the table memory will be only 65,536 bytes.

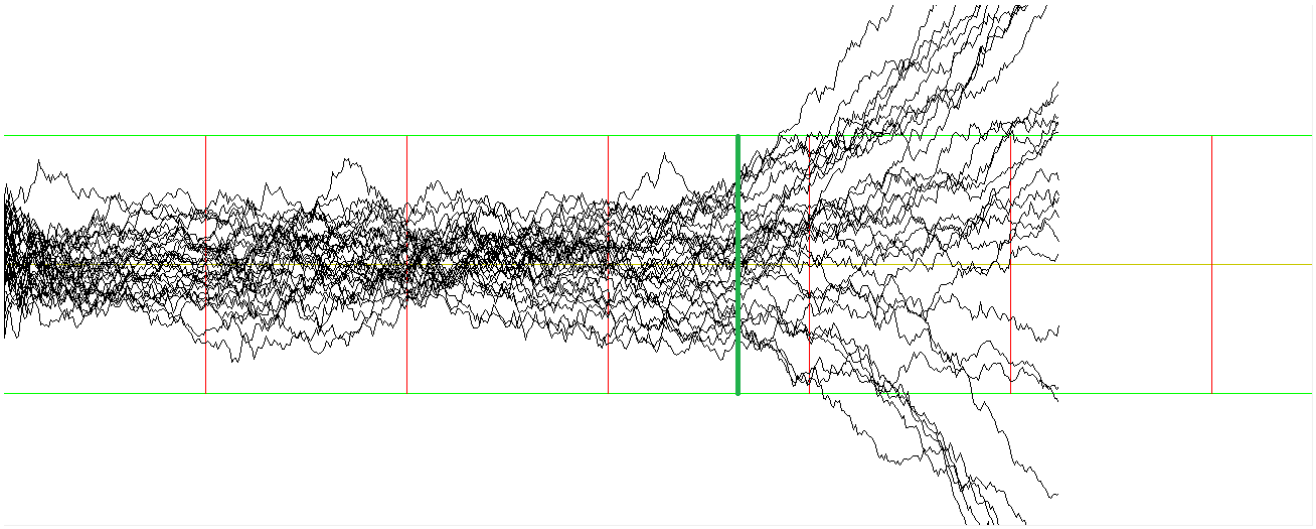
We can stop with 64 RS tables with a width of 16 × 16 bits, though we will need more memory – 8 MB.

Lai-Massey

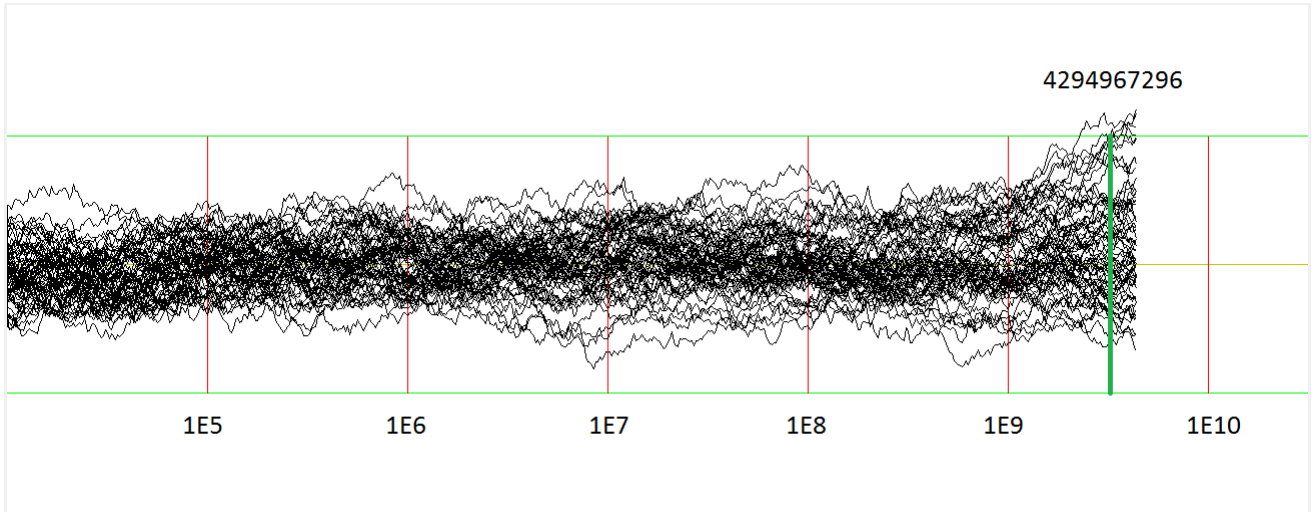
These exists a Lai-Massey strong pseudorandom permutation structure [11] shown in Fig. 2*b* with the same properties as in the Luby-Rackoff scheme, though containing only three different crypto secure round functions (in the left-hand side of the structure, the word is split into two more halves, for which the single-round keyless Feistel network is realised). For B = 128, eighty-one truly random permutations of 8 × 8 bits will be needed with a total memory table size of 20,736 bytes or twenty-seven substitutions of 16 × 16 bits with a memory of about 3.5 MB.

Fig. 3*a* shows the graphs of estimates of dependency probabilities for the recursive decomposition of RS in the form of the Luby-Rackoff structure comprising four RS 8 × 8, each being represented with four RS 4×4 (a total of 16 substitutions 4 × 4). As evident, the decomposition by this criterion can be assumed indistinguishable from RS.

Fig. 3*b* shows an identical "tassel" for the decomposition of RS 32 × 32 with three decomposition stages and sixty-four terminal RS 4 × 4. Here, the indistinguishability boundary is about 2^{32} = 4,294,967,296.



a



b

Fig. 3. Decomposition of RS 16 × 16 based on sixteen RS 4 × 4 (*a*) and RS 32 × 32 based on sixty-four RS 4 × 4 (*b*)

This can be challenged in the sense that nesting SPP into one another yields a new construction whose security can be unclear. This point calls for investigation.

Even-Mansour

But if the security is preserved with SPP nesting, which is confirmed by a multiple experimental estimates of the dependence probabilities, then we can see a way of constructing a large pseudorandom permutation for the Even-Mansour cipher [12] whose variant is shown in Fig. 2*c*.

This cipher includes one random or pseudorandom substitution with width *B* whose input and output are modulo 2 added with key K with a length of *B* bits.

We substitute RS with our strong pseudorandom permutation obtained by decomposition of RS of required width, and the cipher with a proved security is at hand!

Fig. 4 shows the structure of such a cipher.

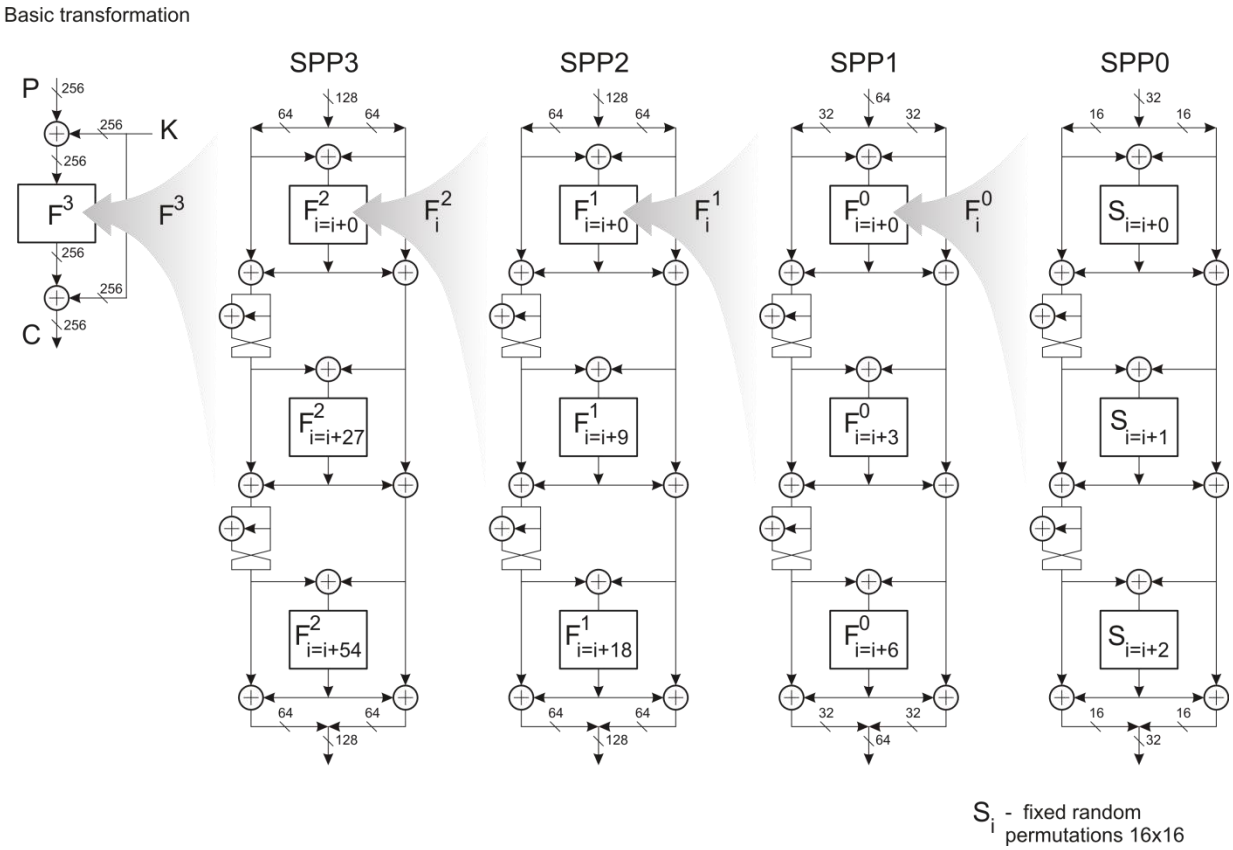


Fig. 4. Structure of the Even-Mansour cipher with decomposition of SPP based on the Lai-Massey schemes

Basic transformation enciphers a block of a 256-bit plaintext P with a 256-bit key K to yield an enciphered 256-bit block C. The strong pseudorandom top-level permutation (F^3) used for RS 256 × 256 is represented by the Lai-Massey SPP3 scheme with three SPP2 (F^2) with a width of 128 bits. Each of them comprises three SPP1 (F^1) with a width of 64 bits, Each of them comprises three SPP0 (F^0) with a width of 32 bits, and each of them includes three truly random substitutions (permutations) S 16 × 16 specified in tabular form. Index *i*

in each SPP determines the numbers of substitutions S included in the higher-level SPPs. The total size of the tables is 10.6 MB – this is rather much. One can pass to the next level of SPP nesting: 243 RS 8 × 8 – that is not much memory – only 62,208 bytes.

Have we invented a new enciphering algorithm? No, we have invented nothing. We have simply shown that a symmetric block cipher should not necessarily be an iterative one and have no proof of its security (for the sake of beauty, we will call this cipher "Kpin" (fennel), by analogy with the dill fractal structure).

Wide S-blocks

To speed up the program realisation of ciphers, special tables are built. They combine substitution nodes and subsequent linear mixing operations. Such tables (e.g., for "Kalina-128") are represented as eight arrays of 256 64-bit words, and they differ for direct and reverse transformation. Table outputs are modulo 2 added in a definite order. Instead of having four substitution tables with a total size of 1 KB, we obtain two sets of eight tables each with a size of 16 KB, but with a speed higher by several-fold.

And if we try to fill in tables with random 64-digit numbers, and simply perform modulo 2 addition of the outputs? Then we will obtain an irreversible transformation (but we are not scared of irreversibility), for which each output bit will depend on all output ones and each input bit will affect all output ones. The values of the avalanche and strict avalanche criterion of such a substitution are close to RS indicators.

Multipliers

Currently, yet another one, all the more available resource has appeared, viz. multipliers. Their application is experiencing a boom owing to the increasing sizes of problems in digital signal processing (DSP). If fifty years ago a colonel had to go to the town of Zelenograd and resort to his connections and contacts to get hands on a 12x12 multiplier microcircuit at the price of a

motorbike, then presently nearly each microcontroller has a DSP hardcore with several multipliers. An average FPGA microcircuit has hundreds and even thousands of DSP units, each with up to four multipliers.

The belated view of multiplication as an expensive operation has engrained the practice of using the simple operations of tabular substitution, addition and shifting in cryptographic primitives.

Multipliers are combinatorial circuits, which extensively realise tables, addition and shifts, i.e. from the engineering standpoint, a multiplier is a powerful circuit ensuring both confusion and diffusion during transformation of input bits to output ones. Usually, the integer multiplier of $n*n$ digits has an output of $2n$ digits, with the standard value of n being within 8 to 32.

The diffusion properties of multiplier outputs are non-uniform: the most best mixed are the middle digits of the output word. Once the great Knuth, in the second volume of his "The Art of Computer Programming" described J. von Neumann's random number generator called the "middle of the square": A number was squared, its middle digits were taken, shifted to the right and squared again. We can do the same when expanding the number of digits after multiplication.

Here is a question: what is to be multiplied by what to achieve maximum diffusion? The most effective transform is believed to be an integral one (of the Fourier type), in which each output is formed by a weighted sum of all inputs. That is, each input is multiplied by a certain coefficient, and all the products are summed and form one output. For the next output, all the inputs are summed with multiplication by other coefficients, and so forth.

Let input A of our transform be represented as concatenation M of parts of A_m with n digits each (for instance, $M = 4$ and $n = 16$), then transform $A => B$ will have the form

$$B_m = \sum_{i=0}^{M-1} k_{m,i} A_i ,$$

where *k* are certain coefficients.

What is used as coefficients? An obvious conclusion offers itself: of course, the subkeys. However, the question remains: *M* inputs with *n* bits yield *M* outputs with 2*n* bits, so what do we do with the excess bits? During expansion of the number of digits after multiplication, we can follow J. von Neumann: isolate the middle *n* bits. Such a structure seems complicated, but for DSP blocks this is a typical "butterfly" used in FFT computations. By truncating the most significant and least significant digits our transform becomes nonlinear and irreversible, though we know that this is okay. In doing so, the values of the avalanche and strict avalanche criteria of such a transform become close to RS indicators.

Multipliers and wide blocks in round functions

Let us build a round function based on the above-described structures. Fig. 5 is an example of such a construction with a "butterfly" with sixteen multipliers 16*16 and four 32-bit modulo 2 adders 2^{32} and eight wide S blocks with random substitutions 8 × 64 with outputs combined by a 64-bit modulo 2 adder. The input is a 64-bit word, which is split into four 16-bit words, each of which is multiplied with four 16-bit subkeys (a total of 16 words or 256 bits). The products in regrouped order are added in four 32-bit adders. Each sum is truncated by eight digits on the right and left sides to form 16-bit words whose concatenation yields a 64-bit word sent to 8-bit inputs of eight S blocks. The 64-bit outputs of S blocks are modulo 2 added to form the round function output (called "MUL").

Why is such a function good? This is because it returns the values of the avalanche criterion and strict avalanche criterion, which are undistinguishable from a random substitution. This means that it is a true cryptographically secure round function.

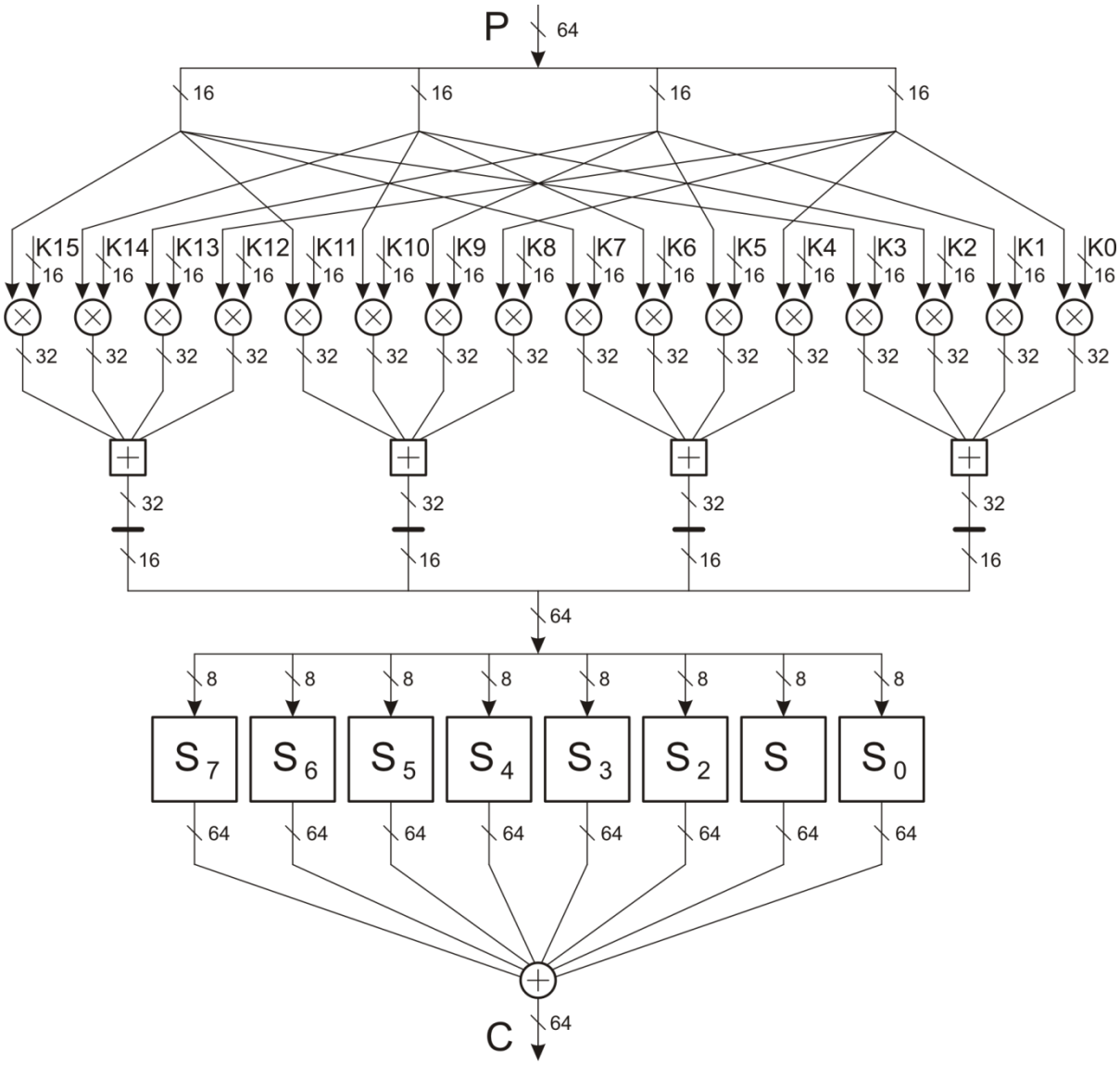


Fig. 5. 64-bit round function MUL with a "butterfly" and wide S blocks

A reduced version of the mini MUL round function splits the 16-bit input into four 4-bit words, computes the "butterfly" with sixteen multipliers built around sixteen 4-bit gates and four 8-bit adders. The four middle bits of the adder form a 16-bit input for four substitution nodes 4 × 16 whose outputs are modulo 2 added to form the function output.

Fig. 6 shows a graph of estimates of dependencies for a scaled-down version of the round function. Evidently, the mini-MUL is indistinguishable from a true RS 16 × 16.

Based on the behaviour of a 16-bit version of the mini-MUL, the logical

assumption is that the 64-bit version is not inferior. Unfortunately, this is hard to validate in the prequantum epoch.

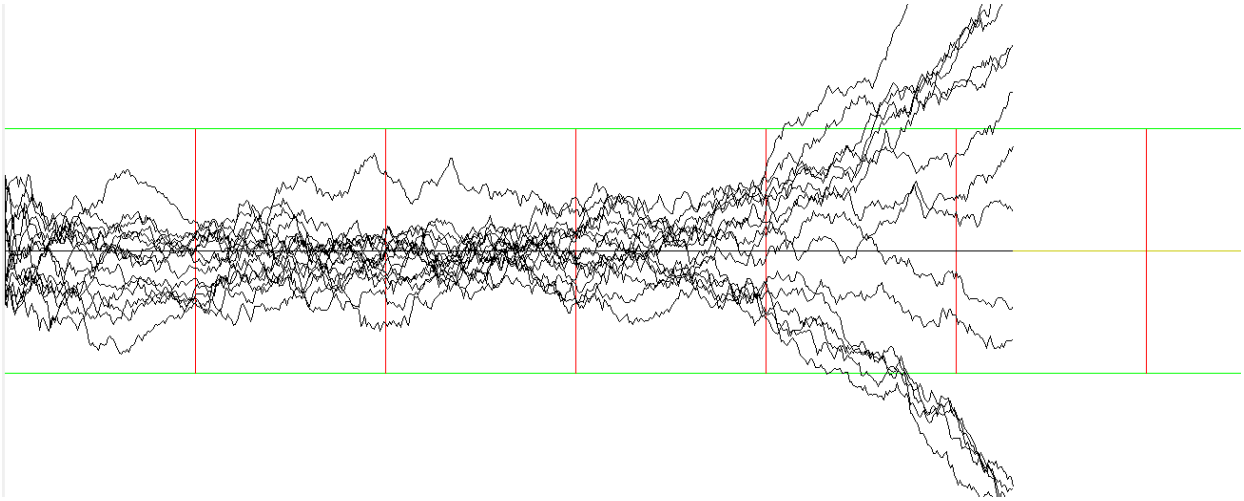


Fig. 6. Estimate of the probabilities of the dependence of a scaled-down version of a mini-MUL round function

For comparison, Fig. 7 shows the graphs of estimates of dependencies for a mini-AES [14] after the 3rd, 4th and 5th rounds. Clearly, the "tassel" tail wags considerably and it very slowly yields the estimates for a true RS of a mini-AES.

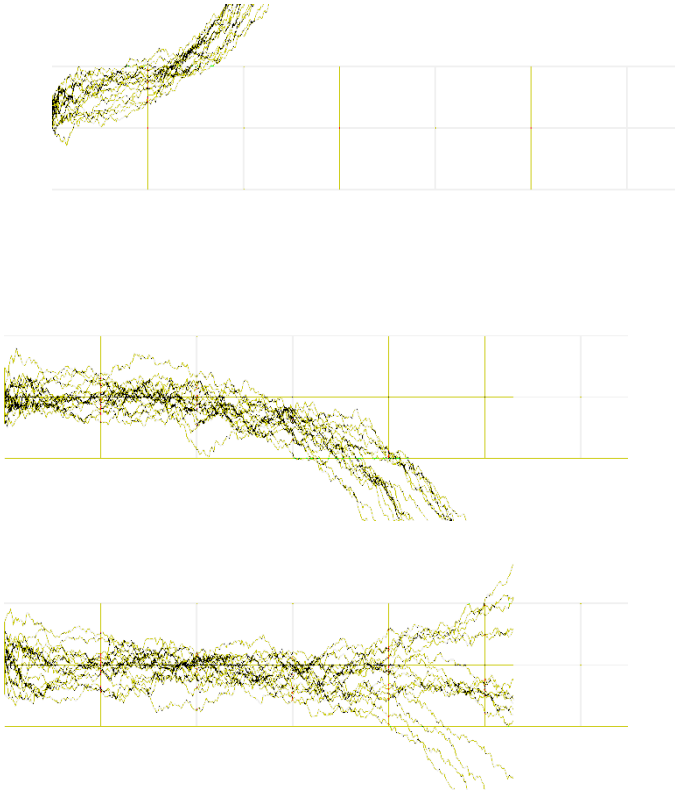


Fig. 7. Estimate of the probability of dependencies for the 3rd, 4th and 5th rounds of a mini-AES

Feistel and SPN

The author has never mentioned the SPN network used recently to build ciphers because he has a penchant for Feistel, and there is a reason for this.

Let us run a simple computational experiment: we take a small Feistel network and SPN (e.g., mini-GOST and mini-AES) and, by searching through all the input vectors, we will compute the estimates of the probability of "1"s at all outputs.

For Feistel, these probabilities will be strictly equal to 0.5, whereas for SPN, they will differ slightly from 0.5.

Now we will use a prohibited technique: create a defect in S blocks – shift the probabilities to zeros by 10 % (the blocks will lose their properties and become irreversible). Then we compute the estimates once more and see that, for Feistel, they again are equal to 0.5, whereas SPN has shifted to zeros, and almost all have become less than 0.5.

Hence, the SPN structure is prone to pass internal defects to the output, whereas the Feistel network smoothens these defects and pulls the probabilities of outputs to 0.5. So, I would be more cautious with SPN.

Moreover, for instance, neither AES nor "Kuznyechik", as algebraic ciphers, have no single drop of randomness in their structures. In the author's opinion, a cipher can be brought to an ideal RS only by many incidents included in S blocks, differing for different rounds. If this is about a high degree of secrecy, then there is no point in saving on memory. How difficult it is, by applying the same deterministic substitution hundreds of times for encrypting one block, to prove cipher security against attacks, which exploit this property of multiple repetitions of operations from round to round.

And now the cipher!

Cryptography, at any rate for symmetric block ciphers, is more of an engineering science than a strictly mathematical one, and mathematics here is far-fetched. The overall picture for the past fifty years is as follows: crowds of

cipher developers are competing with one another and empirically creating schemes from simple primitives co-opted owing to Claude Shannon: addition with a key, confusion and diffusion, repeated invariably and many times in a hope that the result will be very secure. All are attempting to make a cipher simple, fast and secure. The opposition is also trying very hard in inventing attacks on ciphers. The attacks are escaping reality even more, but are helping to compare the security of various ciphers.

To date, neither known block cipher has a strict proof of security. It is is estimated only by its being secure against all known attacks and by the years of a cipher existence without being compromised.

That being so, we also want to train for a while.

Let us take a complete 64-bit MUL function, assuming it to be a cryptographically secure round function. We substitute it into a 128-bit Luby-Rackoff or Lai-Massey structure and obtain a cipher, also an MUL, with provable security. Since it is known that cryptographically secure round functions in a strong pseudorandom permutation should be different, we apply functions with different tables, and naturally, with different subkeys. The Lai-Massey structure needs a total of 48 Kbytes of tables and 768 bits (48 16-bit words) of subkeys. Pipelining the cipher on an FPGA will require 48 multipliers 16*16, 24 memory units with 256 64-bit words (a total of 48 Kbytes), twelve 4-bit input 32-bit adders and eleven modulo 2 adders (nine 64-bit ones and two 32-bit ones). Pipelining initialisation will need (with account of encrypting occasions) about 60 clock cycles, and then one clock cycle for encrypting one block. With a standard clock frequency of 200 MHz, the encryption speed will exceed 25 Gbits/s or 3.2 GB/s. The received MUL cipher is good!

Wikipedia: "The advantages of a mule are big stamina, strength and a lifespan of up to 40 years."

Dessert

If we place strict constraints on cipher application:

– Only one message is encrypted with one key;
– Only one mode is used – running key ciphering with a fixed synchrosignal;
– The message length does not exceed 64 Kbytes,

then we will manage to reduce the width of a cipher with provable security because the opposition will have a limited-length cipher text. The opposition could have easily broken a small-width substitution if it had the plaintext and the secret text. But if the opposition already has them, there is no point in breaking the substitution because another key will be used for the next message.

Shifting cipher security from the cipher width to the key length will need a big number of cryptographically secure round functions depending on the key.

Let us take a 16-bit-width Lai-Massey structure. The round function is an 8 bit one. Do we know any provable cryptographically secure round functions? Yes, for instance, this is the Even-Mansour structure. With an 8-bit width, it should include a random 8×8 bit permutation (RP) and two 8-bit keys with modulo 2 addition with the permutation input and output. The security of such a function is not high – 2^8. If each round function will use its unique RP and its own keys, then the relationship between the round functions will be excluded and the degree of security will be proportional to the number of rounds. This means that if we want 128-bit key security, then we will need 16 rounds with a 256-bit total length of subkeys. Hence, we obtain the structure of the basic transformation of the "Dessert" cipher shown in Fig. 8.

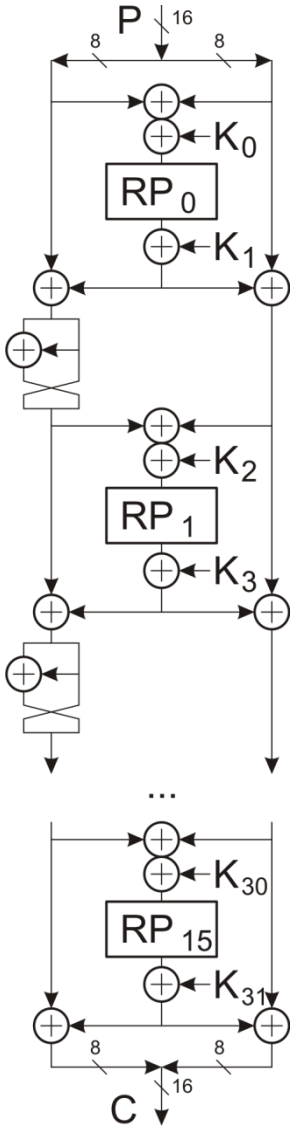


Fig. 8. Structure of the "Dessert" cipher

RP_0.- RP_{15} are sixteen random 256-byte permutations, a total of 4,096 bytes; K_0 - K_{31} are thirty-two 8-bit keys, a total of 256 bits.

If we want to increase key security to 256 bits, we increase the number of rounds to thirty-two and the size of subkeys to 64 (512 bits).

The structure of this lightweight cipher is very simple and the program for the 8-bit controller contains two dozens of lines. Pipelining at the clock frequency of 200 MHz yields an encryption speed of over 3 Gbits/s.

Conclusion. Let us briefly summarise all the aforementioned:

- Instead of testing random sequences for randomness, we need a confirmation of the hypothesis of a truly random generator and if the hypothesis will be confirmed, then filtering and discarding random sequences is a grave error.
- One need not search for and discard "weak keys" – the laws of statistics will rescue us.
- Bruce Schneier recommends using the single encryption mode – running key ciphering because under definite conditions it can replace several other modes and manage without transmission of the initialisation vector.
- It is advisable to make the archiving procedure obligatory and automatic during running key ciphering.
- In a system with encryption of keys they can be not encrypted, but rather, by generating random sequences in the Generation centre, assume them to be encrypted keys. Such virtual encryption of keys slackens demands to Centre security.
- Moreover, instead of keys, virtually expanded subkeys can be generated as random sequences and with the assumption that they are encrypted subkeys.
- The iterated logarithm law (a variant of the law of great numbers) is violated for any substitution, though no earlier than for $n > 2^n$ for an RS of the order of n. If the substitution meets these conditions, then it is undistinguishable from an RS with high probability.
- Decomposition of a random substitution with the help of a strong pseudorandom permutation comprising random substitutions with a half-width size enables building an approximation to an RS of large size with a certain number of small RS realised as tables.

- Such a large strong pseudorandom permutation can be inserted into the Even-Mansour cipher with provable security (the "Kpin" cipher).
- "Wide" S blocks filled in with random numbers whose outputs are modulo 2 added yield an irreversible transformation when each output bit depends on all input ones and each input bit affects all output ones. The values of the avalanche and strict avalanche criterion of such a substitution are close to RS indicators.
- The multiplier is a powerful circuit providing both confusion and diffusion when input bits are transformed into output bits. The most effective transform is the well-known "butterfly" FFT, in which each output is formed by the sum of all inputs weighted by subkeys. The values of the avalanche and strict avalanche criterion of such a substitution are close to RS indicators.
- The MUL round function built of a "butterfly" and wide S blocks returns the values of the avalanche and strict avalanche criterion indistinguishable from a random substitution. Hence, this is a true cryptographically secure round function. The downscaled version of the round function, by estimates of dependence probabilities, does not differ from an RS of respective width.
- In the case of SPN, all is not as good as it appears to be. They pass to the outputs the possible defects of the round function.
- A wonderful cipher can be built by three-fold application of the MUL round function in the Lai-Massey network with provable security against attacks with matching input and output texts. With pipelining, the cipher speed exceeds 25 Gbits/s.
- The 16-bit "Dessert" cipher is paradoxical.

Virtually each of these fifteen statements is a paradox or a challenge, i.e. has a sentence in variance with common belief. Therefore, the host of ideas (if

not all) given above can be seen, mildly speaking, as controversial.

Altschuller's ice-breaker, even if it were granted an Inventor's Certificate, failed to find any practical application because of the inaccurate problem statement. The ice-breaker need not cut the ice with a sharp blade, but should rather crush it by its weight to open a waterway for the convoy of ships following it.

Genrikh Saulovich pointed out on many occasions that the purpose of the book "Algorithm of Invention" was not to provide a ready-made solution, but rather to stir up the inventor and free his eyes from the blunders of well-entrenched beliefs.

References

1. Shannon C. Works in Information Theory and Cybernetics, M., Foreign Literature Publishers, 1963. P. 333-369 [In Russian].
2. Bruce Schneier. Applied Cryptography. Protocols, Algorithms, Source Texts in C. M.: "Triumf", 2002. 816 p.: ill. Print run 3,000 copies, ISBN 5-89392-055-4 [In Russian].
3. Ventsel Ye.S. Theory of Probability: College Textbook. 6th stereotype ed. M.: Vyssh. Shk. Publishers. 1999. 576 p.
4. Koriakov I.V. Designing a generator of truly random numbers for cryptography applications. LLC NVF Crypton. URL: http://www.crypton.ua/images/noiser_2012.pdf
5. Stephenson Neal. Cryptonomicon I-II. Avon Books, 1999.
6. Investigating the cryptographic properties of nonlinear nodes for substitution of downscaled versions of certain ciphers / V.I. Dolgov, A.A. Kuznetsov, I.V. Lisitskaya, R.V. Sergienko, O.I. Oleshko. *Prikladnaya Elektronika* [Applied Radio Electronics], 2009. V. 8. No. 3.
7. Ferguson Niels, Schneier Bruce. Practical Cryptography. [Trans. into Russian]. M., Williams Publishing House, 2004. 432 p.

8. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Special Publication 800–22. Revision 1a. National Institute of Standards and Technology Gaithersburg, MD 0899–8930. Revised: April 2010.
9. Luby M. and Rackoff C. How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing, April 1988. Vol. 17. No. 2. P. 373-386.
10. Lai X., Massey J. L. A proposal for a new block encryption standard. Advances in Cryptology - EUROCRYPT'90, LNCS 473, Springer-Verlag, 1991. P. 389-404.
11. Even S., Mansour Y. A Construction of a Cipher from a Single Pseudorandom Permutation. In: Imai H., Rivest R.L., Matsumoto T. (eds.): ASIACRYPT 1991. LNCS, Vol. 739. Springer, Heidelberg (1993). P. 210–224.
12. Pascale S. The degrees of completeness, of avalanche effect, and of strict avalanche criterion for MARS, RC6, Rijndael, Serpent, and Twofish with reduced number of rounds. [Text] / S. Pascale // Siemens AG, ZT IK 3. April 3, 2000.
13. Raphael Chung-Wei Phan. Mini Advanced Encryption Standard (Mini-AES): A testbed for Cryptanalysis Students / Raphael Chung-Wei Phan // Cryptologia. October 2002. XXVI(4). P. 283-306.