

Article

Adding an Avalanche Effect to a Stream Cipher Suitable for IoT Devices

József Gáll, Pinar Gürgez and Géza Horváth * 

Faculty of Informatics, University of Debrecen, 26 Kassai Road, 4028 Debrecen, Hungary;
gall.jozsef@inf.unideb.hu (J.G.); pinar.gurgez@inf.unideb.hu (P.G.)

* Correspondence: horvath.geza@inf.unideb.hu

Abstract

In recent decades, a wide variety of Internet of Things (IoT) devices have been using encrypted communication. Hence, so-called light-weight cryptography has become especially important. The main advantage of stream ciphers is that their complexity, operation requirements, and memory usage are negligible compared to block ciphers. At the same time, these ciphers do not have the avalanche effect typical of block ciphers. The avalanche effect is the most important advantage of a block cipher over a stream cipher. A good block cipher will have an appropriate avalanche effect, whereas stream ciphers have no avalanche effect at all. Without this effect, stream ciphers can easily be broken by plaintext attacks. In this paper, we study a modified stream cipher and attempt to add an avalanche effect to the system. The original stream cipher at issue is a so-called “DH3 cryptosystem” (Dömösi and Horváth cryptosystem 3), which is particularly suitable for a variety of problems, e.g., for simple IoT devices. We are going to use the stream cipher in the Cipher Block Chaining (CBC) mode of operation. The CBC operational mode is very popular among block ciphers. With this technique, a DH3 stream cipher can be raised to the same level of security as a block cipher, while retaining the simplicity of its design.

Keywords: finite automata; stream cipher; avalanche effect; CBC mode; parameter estimation; goodness of fit



Academic Editor: Yazan Otoum

Received: 12 May 2025

Revised: 15 June 2025

Accepted: 18 June 2025

Published: 24 June 2025

Citation: Gáll, J.; Gürgez, P.; Horváth, G. Adding an Avalanche Effect to a Stream Cipher Suitable for IoT Devices. *Electronics* **2025**, *14*, 2546. <https://doi.org/10.3390/electronics14132546>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In contrast to the trend at the end of the 20th century, when encrypted communication was mostly between computers, in the 21st century, a wide variety of devices are now performing encrypted communication. In addition to mobile phones and tablets, smart devices such as smartwatches, bracelets, virtual reality glasses, and drones are becoming increasingly common. This trend and the devices associated with it are collectively called the Internet of Things (IoT) trend and devices. It has therefore never been more necessary to address the security of simple hardware components. These needs have given rise to so-called light-weight cryptography.

The stream cipher presented by Dömösi and Horváth in 2017 [1] uses a deterministic finite automaton, which is among the simplest computational tools. Their implementation requires only one operation, querying an element in a matrix by specifying its coordinates. The cryptographic architecture is easy to understand, requires very little and simple code, and achieves high-speed secure communication. It is very easy to load it on simple hardware devices, and it works perfectly on simple and cheap devices with minimal storage and computing power.

Let us note here that in previous works, some authors have introduced other cryptosystems that have also been based on the automata theory approach. First, a block cipher version was proposed in [2], and it was later referred to as the DH1 system (Dömösi and Horváth cryptosystem 1) in the literature. The system introduced [1] is a light-weight stream cipher, which is also based on the automata theory approach. We shall refer to this system as DH3. This paper studies an extended version of DH3, where instead of the common linear usage, we apply the CBC operational mode to the system.

The original DH3 novel stream cipher converts the plaintext into ciphertext by taking one byte of plaintext at a time. In contrast to the Vernam system [3], which is a hundred-year-old stream cipher and still the most popular one, it does not use the exclusive or XOR bitwise operation during encryption and decryption. The secret key of the apparatus is the transition matrix of a deterministic finite automaton that forms a Latin square. Thus, the key automaton is chosen randomly from a large set of automata with 256 states and 256 input signals. This algorithm provides a lot of options for choosing the key automaton; therefore, it is impossible to break the system using a brute-force approach. A typical “man in the middle” attack on the Vernam system is called a bit-flipping attack, where one can alter the content of the message without knowing the key. Bit-flipping attacks are impossible on the DH3 stream cipher. A detailed comparison of the DH3 system to the Vernam system is given in [4]. For a general background of automata theory, we refer the interested reader to the monograph in [5].

After introducing such a cryptosystem, we analyze the properties of the system, such as the avalanche effect, as well as the security requirements, including linear and differential cryptanalysis, NIST tests, or resistance regarding different type of attacks (e.g., side channel attacks), etc.

As to the block cipher DH1 mentioned above, it showed good numerical results with respect to the avalanche effect (see [6] for details), and note that this paper uses similar tools to study the same property in the extended system at issue. Furthermore, the DH1 system showed also satisfactory results in the NIST statistical tests (see [7] for further details).

Turning to the DH3 system, the ciphertext generated by the DH3 stream cipher passes the NIST test (presented in [8], 13th International Workshop on Non-Classical Models of Automata and Applications, 18–19 September 2023, Famagusta, short paper section, <https://ncma.emu.edu.tr/en/>, accessed on 15 June 2025), which includes a number of statistical tests developed by the U.S. National Institute of Standards and Technology to determine whether a particular data stream can be considered cryptographically random. Furthermore, due to the structure of this system, it is resistant to side-channel attacks as it performs the same operation in each step for both encryption and decryption.

The only disadvantage of DH3 compared to other advanced symmetric cryptosystems is that it does not have an avalanche effect as this feature has not been available in stream ciphers. That is why, in this paper, we aim to modify the DH3 cryptosystem by extending the system with a CBC mode in order to (hopefully) add an avalanche effect to it, which should greatly increase its security. Therefore, in this paper, we have two main goals: first, we introduce the extension of the DH3 system; second, we focus on the analysis of the desired avalanche effect property.

An avalanche effect is a desirable property of any cryptosystem that encrypts data. The main property here is that a small change in either the plaintext or the key should produce a significant change in the ciphertext (see e.g., [9]). This effect ensures that an attacker cannot easily predict a plaintext through a statistical analysis. For instance, if the alteration in a single bit/byte of the input results in change of only a single bit/byte of the desired output, then it would be easy to crack the ciphertext.

The avalanche effects in cryptography are classified into two types: plaintext avalanche and key avalanche effects. In this paper, we focus on and calculate the plaintext avalanche effect in a specific system, as described below.

Alongside the fact that the avalanche effect is used mostly to measure the security level of block ciphers and hash functions, it can also be used for stream ciphers as our stream cipher operates on a single character at a time and implements a form of feedback mechanism such that the plaintext is constantly changing.

This paper structures as follows: In the following, after recalling the basics of the original DH3 system, we will explain how to extend the DH3 cryptosystem with a CBC mode and offer some examples. We also provide a comparative analysis of our system to other light-weight ciphers regarding some performance metrics. Next, we turn to the investigation of whether the system has a good avalanche effect. For this, several statistical methods (parameter estimates, hypothesis tests) will be applied, and several test results will be presented that should demonstrate the system's avalanche performance.

2. The Original Stream Cipher

From now on we, follow the classical, commonly used notations of automata theory (for further remarks on basic notion, see, e.g., [5]). Suppose that $\mathcal{A} = (A, \Sigma, \delta)$ is an automaton that includes a non-empty and finite state set A .

Our automaton has a special form, satisfying the following conditions: $A = \Sigma$, where for every $a, b \in A$ ($a \neq b$) and $x, y \in \Sigma$ ($x \neq y$), we have $\delta(a, x) \neq \delta(b, x)$ and $\delta(a, x) \neq \delta(a, y)$. Thus, $\mathcal{A}$ is a permutation automaton (each row of the transition matrix forms a permutation of the state set). This is an essential property to ensure the unambiguity of the ciphertext for any plaintext. For security reasons, we assume, additionally, that all of the columns of the transition table also form a permutation of the state set.

Let $\mathcal{A}^{-1} = (A, \Sigma, \delta^{-1})$ be the inverse key automaton for which $\delta^{-1}(b, x) = a$ with $a, b \in A$ and $x \in \Sigma$ if and only if $\delta(a, x) = b$.

The transition function δ is extended to $\overrightarrow{\delta} : A \times \Sigma^+ \rightarrow A$, where $\overrightarrow{\delta}(a, x) = \delta(a, x)$, and $\overrightarrow{\delta}(a, x_1 x_2 \dots x_n) = b$ for every non-empty input word $x_1 x_2 \dots x_n \in \Sigma^+$, $n \geq 2$ such that $\delta(a, x_1) = a_1$, $\delta(a_1, x_2) = a_2$, ..., $\delta(a_{n-1}, x_n) = b$.

During encryption, the plaintext is read in sequentially, character by character, and the key automaton assigns a ciphertext character to each plaintext character. The corresponding ciphertext character will be the state into which the key automaton moves from the assigned state under the effect of the next pseudorandom string. The apparatus creates the ciphertext by linking these characters together. During decryption, the ciphertext is again read sequentially, character by character, and the inverse key automaton assigns to each ciphertext character the corresponding state, which is the same as the original plaintext character. The corresponding plaintext character will be the state into which the inverse key automaton moves from the assigned state under the effect of the mirror image of the next pseudorandom string. The apparatus recreates the plaintext by linking these characters together.

Let $p_1 \dots p_k \in A^+$ be a plaintext and let $r_1, \dots, r_k \in \Sigma^+$ be random strings generated by the pseudorandom number generator started by a seed r_0 . Let us denote the number of rounds by n , and for each pseudorandom string $r_1, \dots, r_k$, $|r_1| = n, \dots, |r_k| = n$ (so they are of the same length). The ciphertext will be $c_1 \dots c_k$ with $c_1 = \overrightarrow{\delta}(p_1, r_1), \dots, c_k = \overrightarrow{\delta}(p_k, r_k)$.

Let $c_1 \dots c_k \in A$ be a ciphertext and let $r_1, \dots, r_k \in \Sigma^+$ be the same random strings generated by the pseudorandom number generator started by a seed r_0 . Then, the decrypted plaintext will be $p_1, \dots, p_k$ with $p_1 = \overrightarrow{\delta^{-1}}(c_1, (r_1)^R), \dots, p_k = \overrightarrow{\delta^{-1}}(c_k, (r_k)^R)$.

3. Example for the Original Stream Cipher

In this example, we shall show the encryption and decryption of the original stream cipher.

Let $\mathcal{A}$ be the permutation automaton that we use for encryption, and let $\mathcal{A}^{-1}$ be the inverse key automaton for decryption. They have a common state set and tape alphabet, and both of them contain numbers between 0 and 3 in our example. Further, $\mathcal{A} = (A, \Sigma, \delta)$, $\mathcal{A}^{-1} = (A, \Sigma, \delta^{-1})$, $A = \Sigma = \{0, 1, 2, 3\}$. The transition functions are now as follows.

as follows.

δ	A					δ^{-1}	A			
	0	1	2	3			0	1	2	3
0	1	2	3	0	and	0	3	0	1	2
1	2	0	1	3		1	1	2	0	3
2	3	1	0	2		2	2	1	3	0
3	0	3	2	1		3	0	3	2	1

Suppose we receive the following pseudorandom numbers and plaintext:

pseudorandom numbers: 0 1 0 1 2 1 3 3
 plaintext: 0 1 2 3 1 0 3 0.

Then, the ciphertext that we receive is as follows:

ciphertext: 1 0 3 3 1 2 1 0.

During encryption, we calculate the ciphertext with the following formulas:

$$\begin{aligned} c_1 &= \delta(p_1, r_1) = \delta(0, 0) = 1, \\ c_2 &= \delta(p_2, r_2) = \delta(1, 1) = 0, \\ c_3 &= \delta(p_3, r_3) = \delta(2, 0) = 3, \\ &\dots \end{aligned}$$

During decryption, we use the inverse transition matrix to calculate the following:

$$\begin{aligned} p_1 &= \delta^{-1}(c_1, r_1) = \delta^{-1}(1, 0) = 0, \\ p_2 &= \delta^{-1}(c_2, r_2) = \delta^{-1}(0, 1) = 1, \\ p_3 &= \delta^{-1}(c_3, r_3) = \delta^{-1}(3, 0) = 2, \\ &\dots \end{aligned}$$

It is also possible to calculate each ciphertext with more rounds. In the following example, we show the calculations with two rounds. First, we calculate c'_1 , then c_1 , then c'_2 , then c_2 , etc. For this,

pseudorandom numbers: 3 2 2 1 0 1 3 3 1 1 2 1 2 3 0 2
 plaintext: 0 1 2 3 1 0 3 0
 ciphertext: 3 0 3 3 2 3 2 1

During encryption, we calculate

$$\begin{aligned} c'_1 &= \delta(p_1, r_1) = \delta(0, 3) = 0 \text{ and } c_1 = \delta(c'_1, r_2) = \delta(0, 2) = 3, \\ c'_2 &= \delta(p_2, r_3) = \delta(1, 2) = 1 \text{ and } c_2 = \delta(c'_2, r_4) = \delta(1, 1) = 0, \\ c'_3 &= \delta(p_3, r_5) = \delta(2, 0) = 3 \text{ and } c_3 = \delta(c'_3, r_6) = \delta(3, 1) = 3, \\ &\dots \end{aligned}$$

During decryption, we use the inverse transition matrix and the pseudorandom numbers in reverse order to calculate

$$\begin{aligned} c'_1 &= \delta^{-1}(c_1, r_2) = \delta^{-1}(3, 2) = 0 \text{ and } p_1 = \delta^{-1}(c'_1, r_1) = \delta^{-1}(0, 3) = 0, \\ c'_2 &= \delta^{-1}(c_2, r_4) = \delta^{-1}(0, 1) = 1 \text{ and } p_2 = \delta^{-1}(c'_2, r_3) = \delta^{-1}(1, 2) = 1, \\ c'_3 &= \delta^{-1}(c_3, r_6) = \delta^{-1}(3, 1) = 3 \text{ and } p_3 = \delta^{-1}(c'_3, r_5) = \delta^{-1}(3, 0) = 2, \\ &\dots \end{aligned}$$

Finally, we note that the original stream cipher works with one-byte-long states. The size of the transition matrix is 256×256 , and it uses eight rounds during encryption and decryption.

4. Applying the CBC Mode

The cipher block chaining mode of operation was invented by Ehrtam, Meyer, Smith, and Tuchman in 1976 (see e.g., [10,11]). In our modified cryptosystem, we use the CBC mode of operation. Encryption is processed character by character, which means byte by byte in our case, and we use the exclusive or XOR bitwise operation. We calculate the XOR value of each byte of plaintext and the previous byte of the ciphertext and then the result will be encrypted. Thus, each ciphertext byte depends on all plaintext bytes processed up to that point. To make each message unique, the initialization vector must be used starting from the first byte. This way, each ciphertext character depends on all plaintext characters processed up to that point. Thus, changing one byte of the plaintext would imply only the change to the ciphertext bytes starting from the same position (rank) but not the previous ones. By applying two rounds, all ciphertext bytes will be the subject of a change, no matter where (in what position) the plaintext was modified. Hence, in order to have an appropriate avalanche effect, we have to repeat this procedure twice. At least two loops are necessary, and for the second loop, we use the last byte of the ciphertext of the first loop.

Let us use the following notation.

The plaintext: $p_1 p_2 \dots p_k$; the pseudorandom strings: $r_1, r_2, \dots, r_{2k}$;

The ciphertext: $c_1 c_2 \dots c_k$;

The initialization vector: IV;

The key automaton: $\mathcal{A} = (A, \Sigma, \delta)$;

The inverse key automaton for decryption: $\mathcal{A}^{-1} = (A, \Sigma, \delta^{-1})$.

Consider first the encryption (which is also shown in Figures 1 and 2). The first byte of the ciphertext (c'_1) is obtained by applying the exclusive OR operation bitwise to the first byte of the plaintext (p_1), which gives the inner state of the transition function of the key automaton using input string r_1 from the pseudorandom string (see Figure 1). In the following steps, the last ciphertext byte plays the role of the initial vector of the first step; that is, more formally, we have for the first loop:

$$c'_1 = \overrightarrow{\delta(\text{IV} \oplus p_1, r_1)}, c'_i = \overrightarrow{\delta(c'_{i-1} \oplus p_i, r_i)}, \quad i = 2, \dots, k.$$

As concerns the second loop, we use the same procedure (with the same automaton) such that the ciphertext bytes obtained in the first loop play the role of the inner states. For the first byte, instead of the original initial vector, we use the last byte given by the first loop in the OR operation (see Figure 2).

So, formally, we have the following:

$$c_1 = \overrightarrow{\delta(c'_k \oplus c'_1, r_{k+1})}, c_i = \overrightarrow{\delta(c'_{i-1} \oplus c'_i, r_{k+i})}, \quad i = 2, \dots, k.$$

During decryption we use the inverse key automaton (Figures 3 and 4 show the whole process). First, we have to calculate c'_k ; then, we have to calculate $c'_1, \dots, c'_{k-1}$ before we can calculate the plaintext $p_1 \dots p_k$.

For the decryption steps, however, we use the bytes of the pseudorandom strings in the opposite order (see Figure 3).

$$\text{First loop: } c'_k = \overrightarrow{\delta^{-1}(c_k, (r_{2k})^R)} \oplus c_{k-1}, c'_1 = \overrightarrow{\delta^{-1}(c_1, (r_{k+1})^R)} \oplus c'_k, \\ c'_i = \overrightarrow{\delta^{-1}(c_i, (r_{k+i})^R)} \oplus c'_{i-1}, \quad i = 2, \dots, k-1,$$

Second loop: $p_1 = \overrightarrow{\delta^{-1}(c'_1, (r_1)^R)} \oplus IV$, $p_i = \overrightarrow{\delta^{-1}(c'_i, (r_i)^R)} \oplus c'_{i-1}$, $i = 2, \dots, k$.

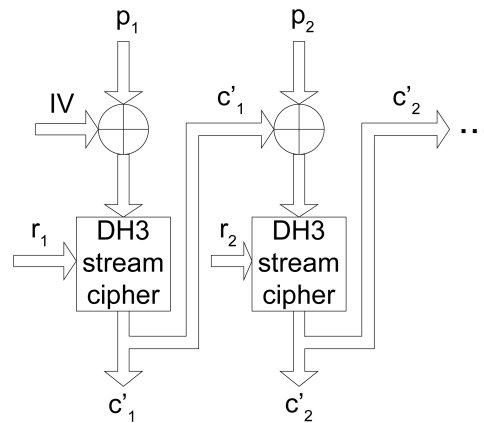


Figure 1. First loop of encryption procedure, using the CBC mode.

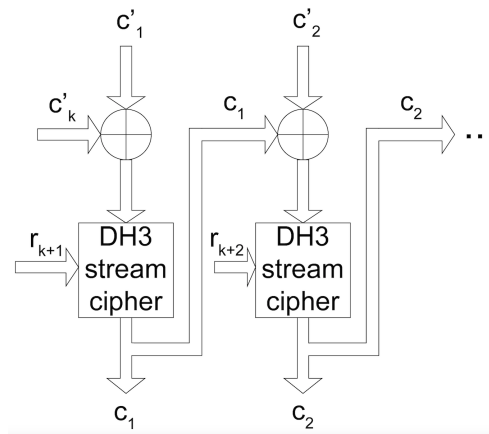


Figure 2. Second loop of encryption procedure, using the CBC mode.

Note that in the second loop (see Figure 4), the initial vector is used again for the first step. The inner states in the second loop are given by the bytes resulting from the first loop starting from (c'_1) .

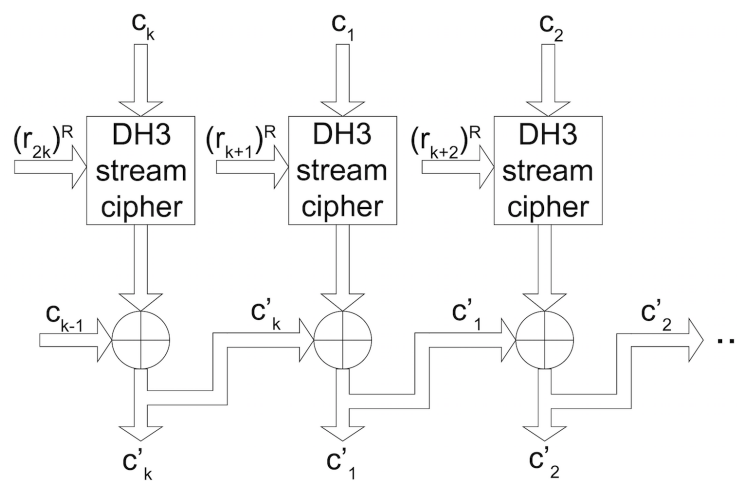


Figure 3. First loop of decryption procedure, using the CBC mode.

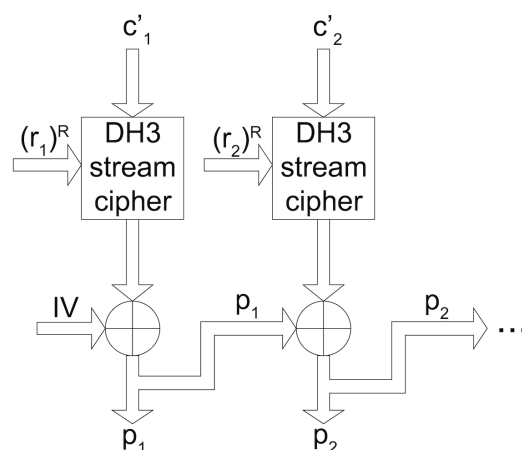


Figure 4. Second loop of decryption, using the CBC mode.

Remarks on Security Implications of the CBC Mode in Light-weight and IoT Environments

The adoption of the CBC mode in the proposed stream cipher architecture serves a specific purpose: to enhance diffusion properties and support the avalanche effect. While CBC remains a widely accepted and well-studied mode of operation in symmetric cryptography, its integration into light-weight cryptographic designs—particularly those targeting constrained IoT environments—necessitates a more nuanced discussion of the potential security implications and implementation-related vulnerabilities. One notable consideration is the sequential nature of the CBC mode, which inherently prevents the parallel processing of encryption operations. In performance-sensitive applications running on low-power or resource-constrained devices, this characteristic may introduce measurable latency or energy inefficiencies. Although the performance trade-off is often negligible in general-purpose systems, it becomes a critical factor in embedded or battery-operated environments.

Another significant concern lies in the management of initialization vectors (IVs). The CBC mode requires the use of a unique, unpredictable IV for each encryption session to maintain semantic security. In light-weight environments, where high-quality entropy sources may be unavailable or restricted, securely generating and storing IVs becomes a non-trivial challenge. Reuse or poor randomness in IVs can lead to ciphertext patterns that may be exploited by adversaries to infer relationships between plaintexts or, in the worst case, recover partial plaintext information (see [12,13]).

Additionally, the CBC mode has been shown to be susceptible to padding oracle attacks, especially in systems where decryption errors or timing variations are observable by an attacker. Such side channels are more likely in IoT deployments that may not implement robust error-handling routines or constant-time execution paths. This introduces a potential attack vector that can undermine the confidentiality guarantees of the cipher unless explicitly mitigated through secure padding schemes and constant-time implementations (see [14]).

In summary, while the CBC mode contributes positively to the avalanche behavior of the proposed cipher, its use in light-weight and IoT scenarios introduces specific security and performance considerations that warrant careful implementation and mitigation strategies.

There are several studies on this issue. Here, we refer to the great work of Dworkin [12], Vaudenay et al. [14] and Thakor et al. [13].

5. Example for the Novel Cipher

In this example, we show the encryption and decryption of the novel cipher. We use one round, but two loops, for the appropriate avalanche effect.

Let $\mathcal{A}$ be the permutation automaton that we use for encryption, and let $\mathcal{A}^{-1}$ be the inverse key automaton for decryption. They have a common state set and tape alphabet, and both of them contain numbers between 0 and 3 in this example. Further, $\mathcal{A} = (A, \Sigma, \delta)$, $\mathcal{A}^{-1} = (A, \Sigma, \delta^{-1})$, $A = \Sigma = \{0, 1, 2, 3\}$. The transition functions and the XOR function are as follows:

function are as follows.

δ					δ^{-1}					$\oplus$					
A					A										
δ	0	1	2	3	δ^{-1}	0	1	2	3	$\oplus$	0	1	2	3	
0	1	2	3	0	0	3	0	1	2	&	0	0	1	2	3
1	2	0	1	3	1	1	2	0	3		1	1	0	3	2
2	3	1	0	2	2	2	1	3	0		2	2	3	0	1
3	0	3	2	1	3	0	3	2	1		3	3	2	1	0

Suppose we receive the initial vector, pseudorandom numbers, and plaintext given below, and hence the ciphertext that can be seen below:

Initial vector (IV): 2
Pseudorandom numbers: 3 1 0 1 2 2 1 3
Plaintext: 0 1 2 2
Ciphertext: 3 3 0 2

During encryption, we calculate the ciphertext with the following formulas:

$$\begin{aligned}
c'_1 &= \delta(IV \oplus p_1, r_1) = \delta(2, 3) = 2, \\
c'_2 &= \delta(c'_1 \oplus p_2, r_2) = \delta(3, 1) = 3, \\
c'_3 &= \delta(c'_2 \oplus p_3, r_3) = \delta(1, 0) = 2, \\
c'_4 &= \delta(c'_3 \oplus p_4, r_4) = \delta(0, 1) = 2, \\
c_1 &= \delta(c'_4 \oplus c'_1, r_5) = \delta(0, 2) = 3, \\
c_2 &= \delta(c_1 \oplus c'_2, r_6) = \delta(0, 2) = 3, \\
c_3 &= \delta(c_2 \oplus c'_3, r_7) = \delta(1, 1) = 0, \\
c_4 &= \delta(c_3 \oplus c'_4, r_8) = \delta(2, 3) = 2.
\end{aligned}$$

During decryption, we first have to calculate c'_4 ; then, we can calculate $c'_1, c'_2, c'_3, p_1, p_2, p_3, p_4$. We have

$$\begin{aligned}
c'_4 &= \delta^{-1}(c_4, r_8) \oplus c_3 = \delta^{-1}(2, 3) \oplus 0 = 2, \\
c'_1 &= \delta^{-1}(c_1, r_5) \oplus c'_4 = \delta^{-1}(3, 2) \oplus 2 = 2, \\
c'_2 &= \delta^{-1}(c_2, r_6) \oplus c_1 = \delta^{-1}(3, 2) \oplus 3 = 3, \\
c'_3 &= \delta^{-1}(c_3, r_7) \oplus c_2 = \delta^{-1}(0, 1) \oplus 3 = 2, \\
p_1 &= \delta^{-1}(c'_1, r_1) \oplus IV = \delta^{-1}(2, 3) \oplus 2 = 0, \\
p_2 &= \delta^{-1}(c'_2, r_2) \oplus c'_1 = \delta^{-1}(3, 1) \oplus 2 = 1, \\
p_3 &= \delta^{-1}(c'_3, r_3) \oplus c'_2 = \delta^{-1}(2, 0) \oplus 3 = 2, \\
p_4 &= \delta^{-1}(c'_4, r_4) \oplus c'_3 = \delta^{-1}(2, 1) \oplus 2 = 2.
\end{aligned}$$

6. Experimental Results

In this section, we turn to the analysis of the avalanche effect of our extended DH3 system. The avalanche effects in cryptography are classified into two types: plaintext avalanche and key avalanche effects. In what follows, we shall calculate the plaintext avalanche effect in our case.

Different cryptographic algorithms have various avalanche effects [15], and they are often categorized as “good” if the value of the measure of the avalanche effect ranges between 45 and 60%; furthermore, they are said to be “very good” if the value of the

measure of the avalanche effect is 50% (see, e.g., [16]). One can here employ the following simple formula to measure the avalanche effect:

$$\text{Avalanche effect (AE)} = \frac{\Sigma \text{ bit (or byte) change}}{\Sigma \text{ total bit (or byte)}} \times 100 \, \%.$$

In order to test the avalanche effect statistically, we have generated a sample in the following way. Given a plaintext, we have changed a single bit in a text of 128 bits and compared the two texts; that is, we calculated the number of bits that remain the same in the new ciphertext. We have generated two different samples: in the first case, we used two encryption rounds for each sample element, whereas in the second case, we applied four rounds of encryption. In what follows, we shall refer to these samples as the two-round sample and the four-round sample, or as two-round and the four-round cases.

So, each element of the sample is an integer showing the number of bits that remained unchanged in the text after changing a single bit. Obviously, any element may take a value between 0 and 128. We have generated a sample of size 10,000 in both cases.

In the case of an ideal avalanche effect, the plaintext and the ciphertext have seemingly no relationship; i.e., the ciphertext looks like an independent text. Thus, we will consider the binomial distribution with parameters $m = 128$ and $p = 0.5$ to be a reference distribution. In the case of an ideal avalanche effect, we would obtain a sample in our experiment that would show a good fit to this distribution; that is, it would be very similar to the reference. (Note that this reference distribution is obtained when generating 128 random bits uniformly and then taking their sum.) Hence, in what follows, we shall refer to this distribution at issue as the “theoretical distribution” in our experiment. Note that a similar statistical approach for testing the avalanche effect to that which we present here was applied for a different setup in [6].

Some of the main characteristics of the two samples are shown by the descriptive statistics in Table 1.

Table 1. Main descriptive statistics of the samples.

Sample	Minimum	Lower Quartile	Median	Mean	Upper Quartile	Maximum
Two rounds	43.0	60.0	64.0	64.8	68.0	85.0
Four rounds	44.0	60.0	64.0	64.1	68.0	85.0

One can see that in both cases, the statistics show values that are fairly close to those of the theoretical example.

Next, we calculated some obvious point estimations to compare the empirical distribution with the theoretical one. First, we simply took the relative frequency distribution of the samples; i.e., we calculated the point estimate of the probability of each value between 0 and 128. Figure 5 shows the relative frequency distribution together with the theoretical probability distribution of both samples. We can see again that both samples provide a relative frequency distribution that is fairly close to the theoretical one.

Now, let us turn to the differences (in other words, the errors) between the empirical and the theoretical distributions in both samples. The maximum of the absolute difference—where the maximum is taken for all possible values—gave a value of 0.008547974. This, again, is a nice result, showing that the two distributions are very close to one another. To underline this, note that the corresponding maximal margin of error values for confidence levels 95% and 99% for such estimations are 0.00979982 and 0.01287915, respectively.

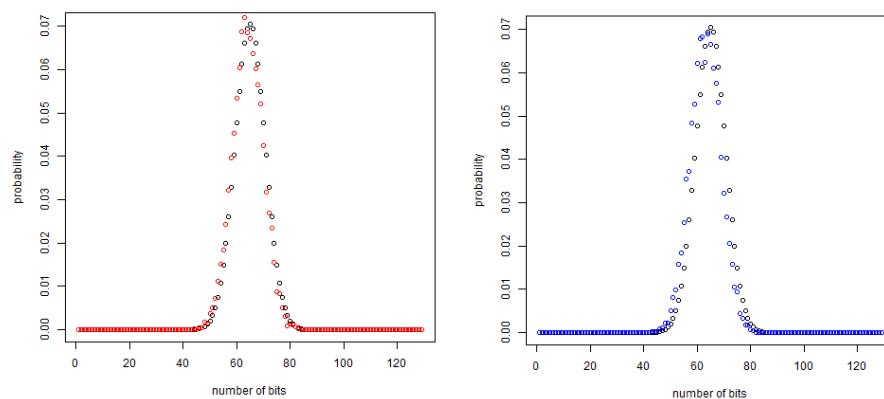


Figure 5. The relative frequency distributions of the samples, with the two-rounds case in red (**left**) and the four-rounds case in blue (**right**), together with the reference binomial (128, 0.5) distribution (black).

Second, we have also considered the direct estimation of the parameters of the binomial distribution; i.e., we basically fitted a binomial distribution to the sample. Now, if we assume that both parameters are unknown, then the method gives the following parameter estimations:

$$\hat{p}_{(2)} = 0.4999426, \quad \hat{m}_{(2)} = 128.1787$$

for the two-round case, and

$$\hat{p}_{(4)} = 0.497767, \quad \hat{m}_{(4)} = 128.7679$$

for the four-round case.

However, we know that $m = 128$; hence, the estimates of the other parameters change to $\hat{m}_{(2),2} = 0.5006406$ and $\hat{m}_{(4),2} = 0.5007531$, respectively, which show a fairly small error ($\leq 10^{-3}$). All in all, we may conclude that the empirical distribution given by the sample is fairly close to the theoretical distribution; hence, the system has a nice avalanche effect.

Finally we have run several goodness-of-fit tests. For this, in both cases, we considered 20 subsamples of size 500. The range of the sample within the interval $[0, 128]$ of the possible values was cut into eight classes (subintervals). Then, the chi-square goodness-of-fit test was applied to each sample ($df = 7$). We set the significance level to 1%. In this case, 11 and 12 samples (out of 20) showed no significant difference from the theoretical distribution in the two-round and four-round cases, respectively. (Note that a sample size of 500 is sufficiently large to identify minor significant differences.)

7. Performance Comparison with Light-Weight Ciphers in the CBC Mode

To assess the practical feasibility of the proposed cipher when deployed in the CBC mode within resource-constrained environments, a comparative evaluation was conducted against representative light-weight block ciphers: PRESENT, Simon, Speck, and software-based AES-128. While no direct benchmarking was performed for this study, the comparison is informed by the cipher's algorithmic structure and published performance metrics for light-weight cryptosystems on similar microcontroller platforms. Furthermore, the estimates are grounded in implementation experience, including the NIST SP 800-22 randomness evaluations conducted on standard consumer hardware. Several papers deal with the systems mentioned above; here, we refer to [17] for a general description of AES, [18] for the case of Present, and [19] for the case of Simon and Speck, whereas [20] provides valuable notes on the comparison of these systems.

Considering the estimated processing speed, due to its structure (featuring XOR, modular addition, and bitwise shift operations within a reduced number of rounds), the encryption throughput of the proposed cipher is estimated at approximately 20–22 Kbps on platforms such as the ARM Cortex-M series. This estimation is based on the simplicity of the round function and comparisons with light-weight block ciphers of similar complexity. It is consistent with reported figures for Simon and Speck, and notably, it outperforms PRESENT and software-only implementations of AES, which suffer from higher computational overhead due to more complex transformations (see [18]).

Regarding to the memory utilization, the algorithm’s design avoids large S-boxes and extensive lookup tables, resulting in a compact memory footprint. On platforms similar to the test machine previously used for NIST randomness testing (specifically, a 2019 Asus VivoBook with Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz and 8 GB RAM, Intel, Santa Clara, CA, USA), the compiled implementation is expected to require approximately 2.7 KB of flash memory and less than 100 bytes of RAM. These figures place it in line with Simon and Speck and substantially below AES-128, which typically requires over 10 KB of flash memory in software implementations.

Turning to energy efficiency issues, we note that thanks to the light-weight operations and the absence of complex transformations, energy consumption per encrypted byte is projected at approximately 0.9–1.0 μJ , assuming typical microcontroller profiles. This estimate is consistent with energy-efficient designs like that of Speck and Simon and reflects a notable improvement over AES (see [19]), which incurs higher energy costs due to its computational complexity (see also [18]). As to implementation simplicity, the cipher employs exclusively arithmetic and logical operations with a fixed control flow, avoiding data-dependent branches or cryptographic tables. This simplifies both hardware and software implementation, facilitates portability across embedded platforms, and reduces susceptibility to timing-based side-channel attacks. In comparison to PRESENT, which uses many rounds, and AES, which involves complex non-linear transformations, the proposed cipher offers a favorable balance between security and implementation cost.

We provide a summary of the above notes in Table 2.

Table 2. Estimated comparative summary.

Cipher	Throughput (Kbps)	Flash Memory (KB)	RAM Usage (Bytes)	Energy ($\mu\text{J}/\text{Byte}$)	Implementation Complexity
Proposed Cipher (Est.)	20–22	2.7	<100	0.9	Low
PRESENT	12–15	2.5	100	1.1	Low
Simon	15–20	2.5	100	0.95	Moderate
Speck	20–25	3.0	100	0.90	Moderate
AES-128 (Software)	10–15	>10	>200	>1.4	High

All in all, the estimated resource profile of the proposed cipher in the CBC mode suggests strong suitability for resource-constrained environments such as low-power IoT devices. With minimal memory requirements, low energy consumption, and a structurally simple design conducive to secure implementation, the cipher compares favorably with prominent light-weight alternatives.

8. Conclusions

In this paper, we introduced a light-weight stream cipher in the CBC mode and demonstrated that this extension leads to an appropriate avalanche effect. Such ciphers are particularly suitable for a variety of IoT devices due to their relatively simple structure.

We analyzed the strength of the this system's avalanche effect via different statistical estimations and tests using large generated random samples, where the main focus was on the number of bits that remained the same in the new ciphertext after changing a single bit. Parameter estimates, as well as goodness-of-fit results, all show that the performance of the system is fairly close to an ideal avalanche effect.

Based on the results of this research, it can be concluded that the security level of the Dömösi and Horváth stream cipher in the CBC mode can be classified as good, as proven by its plaintext avalanche effect, which means that it can be used to improve system security.

We end by noting that beyond the avalanche issues, one could further analyze the properties of the system as laid out in the introduction. In particular, a complex security analysis—including tools like NIST tests, diehard tests, linear and differential cryptanalysis, etc.—could be the focus of further research, but this was outside of our present scope.

Author Contributions: Conceptualization, P.G. and G.H.; methodology, J.G. and G.H.; software, P.G.; statistical analysis, J.G.; writing, J.G., P.G. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dömösi, P.; Horváth, G. A Novel Stream Cipher Based on Deterministic Finite Automaton. In Proceedings of the Ninth Workshop on Non-Classical Models of Automata and Applications (NCMA 2017), Prague, Czech Republic, 17–18 August 2017; Short Papers; pp. 11–16.
2. Dömösi, P.; Horváth, G. A novel cryptosystem based on abstract automata and Latin cubes. *Stud. Sci. Math. Hung.* **2015**, *52*, 221–232. [CrossRef]
3. Vernam, G.S. Secret Signaling System. U.S. Patent no. US1310719, 22 July 1919.
4. Dömösi, P.; Horváth, G. Advantages of an Automata-Based Stream Cipher. *Surikaiseikikenkyusho Kokyuroku/Rims Kokyuroku* **2024**, *2291*, 1–10.
5. Dömösi, P.; Nehaniv, C.L. *Algebraic Theory of Automata Networks. An Introduction*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 2005; 258p.
6. Dömösi, P.; Gáll, J.; Horváth, G.; Tihanyi, N. Statistical Analysis of DH1 Cryptosystem. *Acta Cybern.* **2017**, *23*, 371–378. [CrossRef]
7. Dömösi, P.; Gáll, J.; Horváth, G.; Tihanyi, N. Some Remarks and Tests on the DH1 Cryptosystem Based on Automata Compositions. *Inform.-J. Comput. Inform.* **2019**, *43*, 199–207. [CrossRef]
8. Chebeb, S.; Horváth, G. *Automata Based Stream Cipher and Pseudorandom Number Generators*; working paper; University of Debrecen: Debrecen, Hungary, 2023.
9. Stallings, W. *Cryptography and Network Security: Principles and Practice*, 5th ed.; Prentice Hall Publications: Upper Saddle River, NJ, USA, 2011.
10. Ehrtam, W.F.; Meyer, C.H.W.; Smith, J.L.; Tuchman, W.L. Message Verification and Transmission Error Detection by Block Chaining. U.S. Patent no. US4074066, 26 April 1976.
11. Patel, D.R. *Information Security: Theory and Practice*; Prentice Hall Publications: Upper Saddle River, NJ, USA, 2008.
12. Dworkin, M. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. In *NIST Special Publication 800-38A*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2019. Available online: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf> (accessed on 15 June 2025).
13. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R.A. Lightweight cryptography for IoT: A state-of-the-art. *arXiv* **2020**, arXiv:2006.13813
14. Vaudenay, S. Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS. In *Advances in Cryptology—EUROCRYPT 2002*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2332, pp. 534–546. Available online: https://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf (accessed on 15 June 2025).
15. Ramanujam, S.; Karuppiyah, M. Designing an algorithm with high Avalanche Effect. *IJCSNS Int. J. Comput. Sci. Netw. Secur.* **2011**, *11*, 106–111.

16. Astuti, N.; Arfiani, I.; Aribowo, E. Analysis of the security level of modified CBC algorithm cryptography using avalanche effect. In Proceedings of the IOP Conference Series Materials Science and Engineering, Aceh, Indonesia, 9–10 October 2018; IOP Publishing: Bristol, UK, 2019; Volume 674, p. 012056.
17. Pub, N.F. 197: Advanced encryption standard (AES). *Fed. Inf. Process. Stand. Publ.* **2008**, *197*, 0311.
18. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Yannick, V.; Vikkelse, C. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES, Vienna, Austria, 10–13 September 2007; Lecture Notes in Computer Science; Volume 4727. pp. 450–466. ISBN 978-3-540-74734-5. [[CrossRef](#)]
19. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *Iacr Cryptol. Eprint Arch.* **2013**, *2013*, 404–449.
20. El-hajj, M.; Mousawi, H.; Fadlallah, A. Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform. *Future Internet* **2023**, *15*, 54. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.