Pakistan Journal of Life and Social Sciences
www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2026-24.2.025

RESEARCH ARTICLE

Software-And-Hardware Module Development for Protecting Against Identification By Keyboard Handwriting

L.V. Cherkesova, I.A. Panfilov[1*], O.A. Safaryan, E.N. Klimova[2]

[1,2]Don State Technical University, 344003, Rostov-on-Don; Gagarin Square, Russia

ARTICLE INFO

Received: OCT 05, 2026

Accepted: OCT 26, 2026

Keywords

Information Security
Keyboard Handwriting
Behavioral Biometrics
Anonymization
Protection of Personal Data, Keyboard Dynamics
Hardware-Software Module, Arduino Microcontroller Platform

Corresponding Author:

safari_2006@mail.ru

ABSTRACT

Annotation. Purpose of the investigation is the development of hardware-and-software module that provides the user's protection against identification based on keyboard handwriting by anonymizing the temporal characteristics of keyboard input. Method. The solution to this problem based on software processing of keyboard input events using multithreaded architecture, buffering the sequence of keystrokes, and generating of independent output stream of characters with programmatically configurable time temporal delays. To verify proposed approach, the algorithm was implemented in Python programming language and its hardware was realized carried out on the Arduino microcontroller platform. Result. Experimental hardware-and-software module has been developed that makes it possible to separate the processes of registering key presses and displaying the entered characters. The proposed approach ensures the modification of temporal characteristics of the data flow output, reducing the possibility of identifying the user by his keyboard handwriting without compromising the correctness of text input. The software-and-hardware implementations of the developed algorithm are presented in the article. Conclusion. The proposed hardware-and-software module makes it possible to increase the level of protection of the user's behavioral biometric data by anonymizing keyboard input. The developed solution can serve as the basis for creating of specialized information security tools designed to counteract to biometric identification based on the dynamics of keystroke presses.

INTRODUCTION

One of the promising directions in modern behavioral biometrics is user identification based on keyboard handwriting, which is based on the analysis of the temporal characteristics of the text input process. In this case, the identification features used are the intervals between successive keystrokes, the duration of key hold, and other parameters of keyboard dynamics, which make it possible to establish the user's identity with high probability without using of additional authentication tools. The use of keyboard handwriting is applied in authentication systems, continuous user monitoring, banking services, corporate information systems, and other software complexes where identity confirmation is required during the work process. At the same time, the widespread adoption of such technologies gives arising to new problem: the possibility of unauthorized user identification based on the specific features of his typing style and his patterns examples and, as the result, breach of the confidentiality of behavioral biometric data.

Ensuring user's protection against identification based on keyboard typing patterns (handwriting) is actual task in information security. One of the promising approaches to solving this problem is the anonymization of temporal characteristics of keyboard input, which reduces possibility of

reconstructing user's individual typing style while maintaining the correct operation of software tools. The development of hardware-and-software tools that provide modification of temporal parameters of keyboard input, before the data is transmitting to the operating system, makes it possible to increase the level of protection for behavioral biometric data and reduce the likelihood of user identification based on keyboard dynamics.

PROBLEM STATEMENT

Currently, there are many software solutions that use methods of analyzing keyboard handwriting (typing) patterns for authentication and continuous user's identification. Such systems are used in information security, the banking sector, corporate information systems, and other areas where person identity verification based on behavioral biometric features is required. At the same time, significantly less attention is paid to developing means to protect users from such identification. Existing countermeasures are usually implemented using software tools that work at operating system level, which limits their scope of application and does not ensure universality across different computing platforms.

The aforementioned circumstances determined the feasibility of research conducting aimed at developing the hardware-and-software module to user protect against his identification based on keyboard handwriting by anonymizing the temporal characteristics of keyboard input. To achieve this goal, the following tasks were formulated and solved, within the framework of this investigation:

Conducting the analysis of existing methods for user's identification based on keyboard handwriting and different ways to protect their personal behavioral biometric data.

Developing the algorithm for keyboard input anonymizing, based on separating the processes of registering keystrokes and generating output stream of characters with altered temporal characteristics.

Implementing the developed algorithm as software module in the programing language Python and transferring it to the Arduino software-and-hardware platform [1, 2].

Assess and estimate the functionality and operability of developed hardware-and-software module and the possibility of using it to reduce the likelihood of user identification based on keyboard handwriting.

RESEARCH METHODS

The investigation is based on the application of methods for software processing of keyboard input events, analysis of the temporal characteristics of keyboard typing, and the development of hardware-and-software tools for the protecting personal behavioral biometric data. The proposed approach involves separating the registering keystrokes process of and the process of generating the output stream of characters, which makes it possible to control the temporal parameters of input independently.

Keyboard input processing algorithm involves the sequentially receiving events from input device, placing them in the buffer of fixed size, and then processing them with programmatically defined time delays. Events of three types are used as queue elements: character input, character deletion, and input completion.

The intervals between event outputs are changed using random delays within specified range, which makes it possible to disrupt the characteristic time dependencies that form the ground for user identification based on his keyboard handwriting personal features.

For verification of functionality and operability of the proposed approach, two implementations of the algorithm have been developed. The software implementation is written in Python programing language using multithreading processing and event queue. The hardware implementation is based on microcontroller platform Arduino with PS/2 keyboard connection, which ensures independent processing of keyboard input before it is transmitted to the computing system [1].

The object of research is process of forming the user's keyboard handwriting when entering textual information. The subject of study is the methods of anonymizing the temporal characteristics of keyboard input, aimed at reducing the possibility of identifying the user based on behavioral biometric features.

The relevance of investigation is determined by the growing use of behavioral biometrics technologies in the modern information systems and the need to develop security tools that can ensure the confidentiality of keystroke dynamics without disrupting the user's usual interaction with the computer system [2].

Architecture of Hardware-And-Software Module Development

Elaboration of this architecture of hardware-and-software module for protection against identification on keyboard handwriting is based on the principle of separating registering keyboard input processes and generating the output stream of characters. This approach makes it possible to change the temporal characteristics of transmitted data without disrupting the characters sequence entered by user and usual scenario of interaction with computing system.

When module's architecture designing, following requirements were formulated: ensuring transparency of user's operation, maintaining the correctness of entered text, independence of keyboard event processing from application software, and possibility of software-and-hardware implementation of proposed algorithm.

To carry out these specified requirements, the keyboard input processing process was divided into several functionally independent stages. In the *first* stage, keyboard events are registered and the sequence of keystrokes is recorded. Subsequently, in the *second* stage, the received events are placed in the processing buffer, which makes it possible to eliminate strict temporal dependence between their registration and subsequent reproduction and playback. After this, for each event, the moment of its transmission to the output stream is determined using software-generated delay. At the *final* stage, the processed events are transmitted sequentially to the operating system or to the connected computing device [3].

Dividing the processing into separate functional blocks makes it possible to change the temporal timing characteristics of keyboard input without disrupting the user's workflow. At the same time, sequence of entered characters is fully preserved, while intervals between events differ from the original ones, which makes it difficult to form a stable biometric profile of the user based on the analysis of keyboard dynamics.

The structural diagram of the developed software-and-hardware module is shown in the Fig. 1.

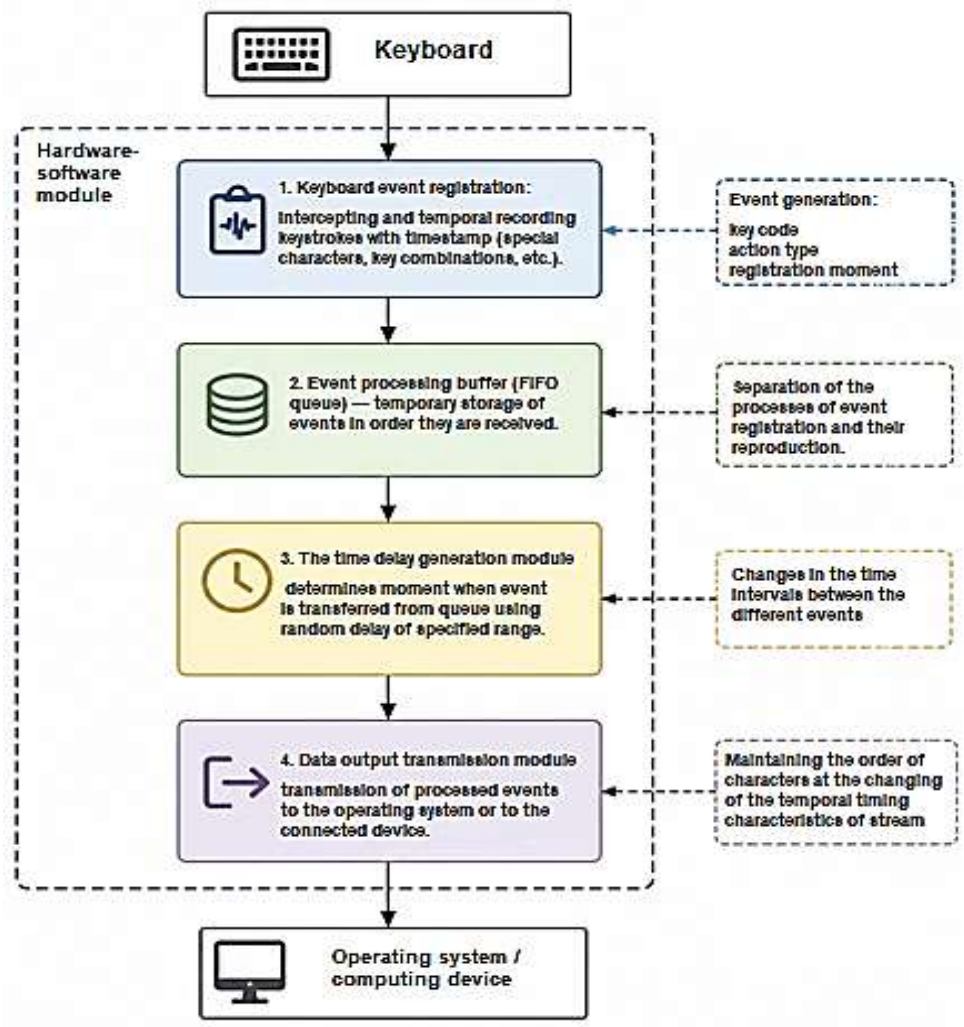


Fig. 1 – Hardware-software module architecture of for protection against identification based on keyboard handwriting

The developed architecture includes four main functional blocks: keyboard event registration module, event-processing buffer, time delay generation module, and data output transmission module. The interaction between the blocks ensures the sequential processing of each keyboard input event and the formation of output stream with modified temporal time characteristics.

The proposed architecture is universal and allows for various implementation options. In the framework of this study, the algorithm was implemented using Python language software, and also transferred to the Arduino microcontroller platform, which made it possible to implement hardware processing of keyboard input independently from the software of the target computing system [2, 4].

DISCUSSION OF THE RESULTS

Registration of events of keyboard input. The *first* stage of developed software-and-hardware module operation is the registration of events of keyboard input. At this stage, all user actions related to pressing functional and character keys are intercepted without making any changes to their sequence or content. The main task of this stage is to generate the stream of events, which is used subsequently to anonymize the temporal timing characteristics of the keyboard handwriting.

Each registered event contains the minimal set of data necessary for its subsequent processing: code of pressed key, type of event, and moment of its registration. This obtained information is not

transmitted directly to operating system but is sent to processing buffer, where it is stored until output data stream is formed.

Using the separate stage of event registration makes it possible to separate completely the process of receiving keyboard input and the process of its subsequent transmission. This separation is key element of the proposed algorithm, as it is this separation precisely enables the possibility of changing the time intervals between events without disrupting the order of entered characters. The general algorithm for registering and subsequent processing of keyboard events is presented in the Fig. 2.

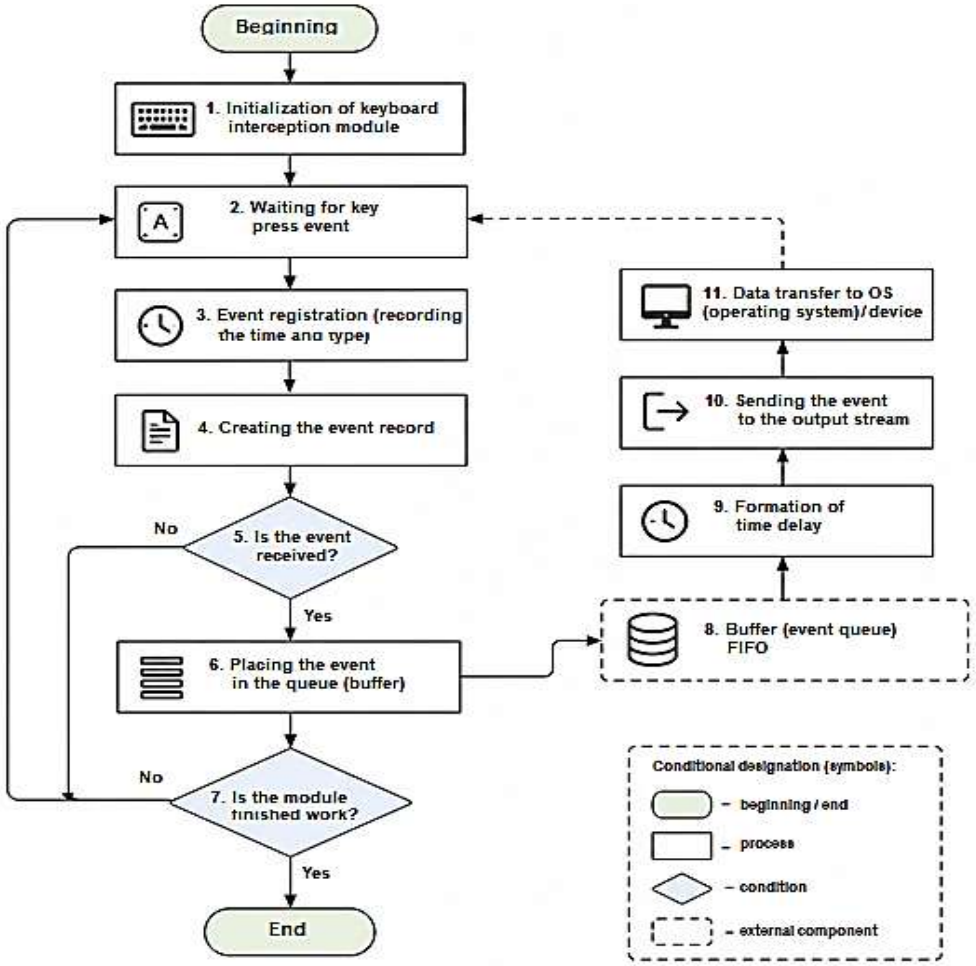


Fig. 2 – Algorithm for registering and processing keyboard input events

Keyboard event registration (logging) in software implementation is carried out using specialized Python library that provides real-time information about key presses. When the event occurs, corresponding record is generated, which is placed in processing queue and awaits further execution of anonymization algorithm.

Organization of Keyboard Event Registration and Processing

Registering of the keyboard input is the *first* stage of developed software module's operation. To obtain the information about user's actions, the ***msvcrt*** library is used; it provides direct reading of keystrokes in Windows operating system without the necessity to wait for the input string to be completed. This approach makes it possible to record each keyboard event and to determine the time intervals between consecutive keystrokes.

After registering the next event, the object is formed and created that contains event type, entered character (if it is present), registration moment, and time interval relative to the previous keystroke. Received data is placed into FIFO queue, which ensures sequential processing of events in the order they are received. Using the queue allows to separate the processes of input registration and character stream output generation, eliminating the mutual influence and interconnection of these operations.

The processing of these accumulated events are implementing by separate worker thread. For each event, the adaptive output delay is calculated, taking into account the current typing speed and the degree of queue filling. After the delay is calculated, the character is displayed in the console, and the event information is saved to the registration log for subsequent analysis. Thanks to the use of multithreaded architecture, the input of new characters is not blocked while waiting for output delays, which ensures stable operation of the software module even with high typing intensity [4].

To illustrate the software implementation of the keyboard event handling mechanism, Fig. 3 shows the fragment of the source code that implements the processing of registered events in the separate execution thread.

```
def process_event(self, event: KeyEvent):

    delay = self.calculator.calculate(
        self.event_queue.qsize()
    )

    time.sleep(delay)

    if event.event_type == "char":

        print(
            event.value,
            end="",
            flush=True
        )

    elif event.event_type == "backspace":

        print(
            "\b \b",
            end="",
            flush=True
        )
```

```
def process_event(self, event: KeyEvent):

    delay = self.calculator.calculate(
        self.event_queue.qsize()
    )

    time.sleep(delay)

    if event.event_type == "char":

        print(
            event.value,
            end="",
            flush=True
        )

    elif event.event_type == "backspace":

        print(
            "\b \b",
            end="",
            flush=True
        )
```

Fig. 3 – Fragment of the software implementation for processing keyboard events

The presented code fragment demonstrates the sequence of processing keyboard events in the separate thread. For each registered event, the adaptive delay value is determined, after which the character is output to the console. Separating of the processes of registration and handling events across the independent threads ensures the stable operation of the software module regardless of the intensity of user input. Interaction structure between the main components of software module is shown in the Fig. 4.

Implementation of Adaptive Delay Algorithm

The main task of the developed software module is to modify the timing temporal characteristics of character output in order to reduce the possibility of identifying the user based on the features of their keyboard typing style – handwriting.

To solve this problem, the adaptive delay algorithm has been developed that dynamically adjusts the output time for each registered event depending on the current text input speed [5].

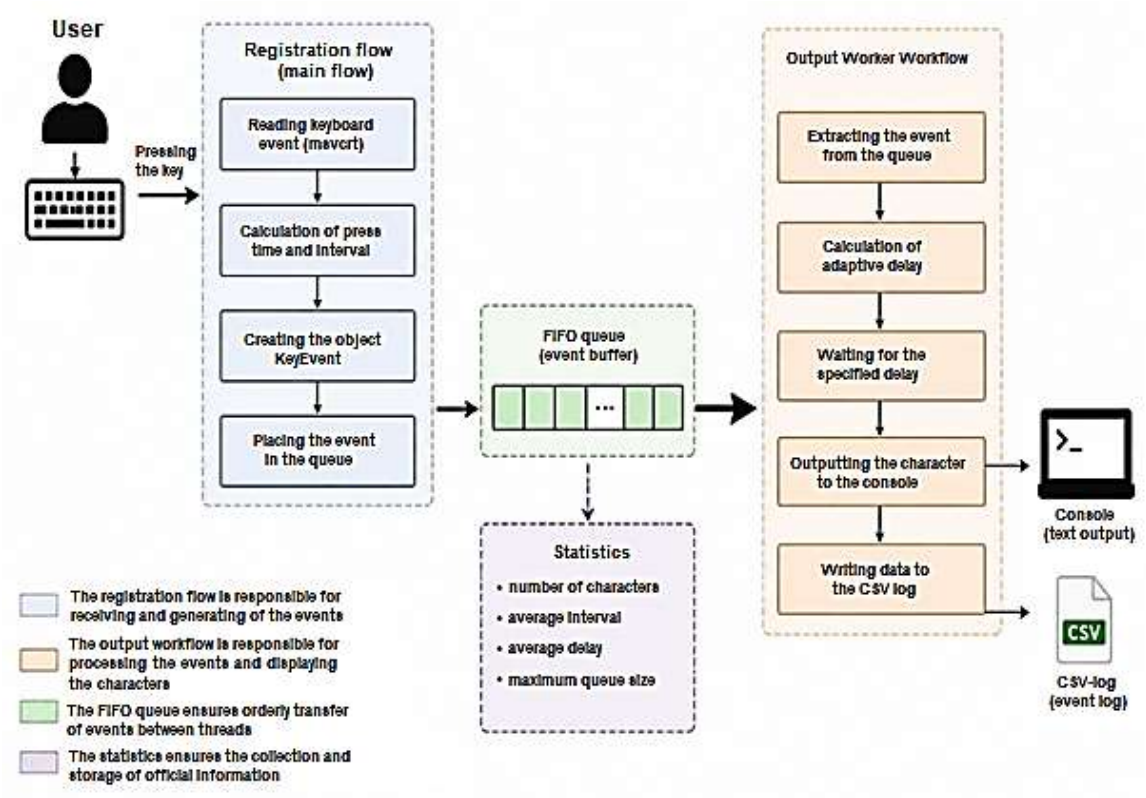


Fig. 4 – Scheme for registering and processing keyboard events

Operation of algorithm begins by determining the time interval between two consecutive keystrokes. After registering next event, calculated interval value is stored in the buffer of latest observations of fixed length. Using sliding window makes it possible to smooth out random fluctuations in typing speed and obtain the stable estimation of user's current input rate. Average value of intervals is calculated using the following expression:

$$\overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i, \quad (1)$$

where: $\overline{T}$ – the average interval between keystrokes; T_i – the interval between two consecutive keyboard events;

n – the number of elements stored in the buffer of the latest observations [5].

Based on the calculated average interval, the base value of the character output delay is determined:

$$D = \min(D_{max}, \max(D_{min}, -k\overline{T}), \quad (2)$$

where: D – calculated delay; D_{min} – minimal permissible delay; D_{max} – maximal permissible delay; k – scaling factor (coefficient) determining the algorithm sensitivity to the changes in printing speed.

From expression (2), it follows that as typing speed increases, the average interval between keystrokes decreases, which results in calculated delay increasing for displaying characters. When

input speed decreases, delay automatically decreases, ensuring comfortable user work and preventing text display excessive slowdown.

To increase the algorithm's resistance to statistical analysis, the random component of the delay is additionally introduced. The final value is determined by the expression:

$$D_f = D + \Delta, \quad (3)$$

where: D_f – final delay of the output character displaying;

Δ – random deviation generated according to the normal distribution law [5].

Using of random component helps to eliminate the emergence of constant temporal regularities and patterns, and creates the more natural structure for characters output stream, which significantly complicates the recovery of user's individual keyboard handwriting characteristics. In cases of high input intensity, the size of queue of unprocessed events is additionally monitoring. If the number of accumulated elements exceeds the set threshold value, delay value is reduced automatically. This mechanism prevents overflowing of event buffer and ensures the stable operation of software module regardless of text typing speed.

After calculating the final result value, it is limited by the minimum and maximum permissible values, which prevents the occurrence of excessively large or excessively small delays and ensures the stability of the algorithm operation under various functional modes.

The program fragment listing of implementing the adaptive delay algorithm is presented in the Fig. 5. The code fragment demonstrates the calculation of adaptive delay based on the average interval between consecutive keystrokes, its adjustment and correcting taking into account the load on the event processing queue, and the addition of random component. This implementation ensures the dynamic adjustment of timing characteristics of keyboard input while maintaining the correct sequence of character output.

```
def calculate(self, queue_size: int):
    avg = self.average_interval()
    delay = MAX_DELAY - avg * 0.5
    if queue_size > 10:
        delay -= 0.02
    if queue_size > 20:
        delay -= 0.02
    delay = max(MIN_DELAY, min(MAX_DELAY, delay))
    delay = random.gauss(delay, JITTER_SIGMA)
    delay = max(MIN_DELAY, min(MAX_DELAY, delay))
    return delay
```

Fig. 5 – Fragment of software implementation of adaptive delay algorithm

For visual representation of described algorithm operations sequence, its flowchart is shown in Fig. 6.

Logging and Analysis of Keyboard Input Parameters.

To evaluate the performance of developed software module, the subsystem for registering keyboard input parameters was implemented. During program execution, the original and calculated time characteristics are saved automatically for each registered event. The obtained data are written to the CSV file, which allows for their subsequent analysis using standard data processing tools, in particular, the Microsoft Excel spreadsheet processor [6].

The registration log records the type of processed event, the moment when the key press is registered, the time interval relative to the previous event, the delay value calculated by the algorithm, as well as the moment of the actual character output. The generated registration log fragment is presented in Table 1.

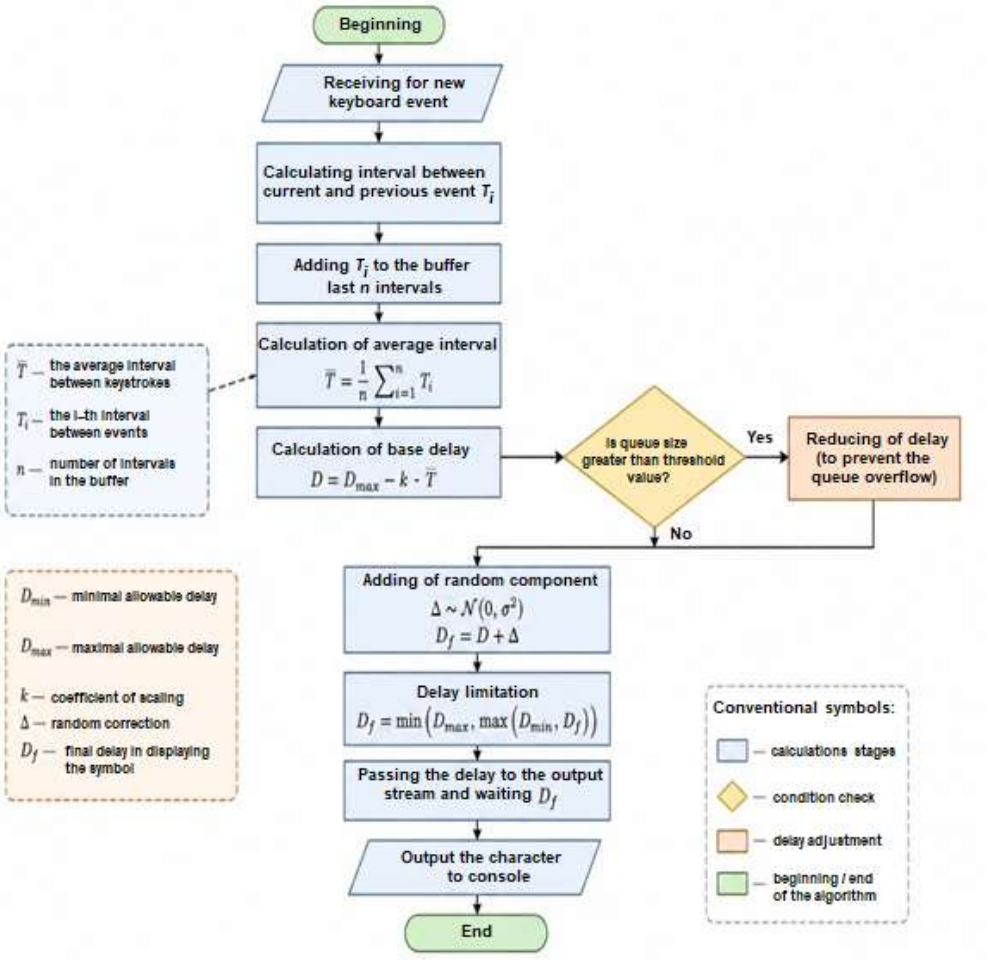


Fig. 6 – Algorithm for calculating adaptive delay

Table 1 – The keyboard input event log fragment

Key	***Press Time***	***Interval***	***Delay***	***Output Time***
v	1,783E+09	0	0,134785	1782666729
f	1,783E+09	0,29385	0,052096	1782666730
l	1,783E+09	0,091143	0,059834	1782666730
b	1,783E+09	0,226013	0,04	1782666730
y	1,783E+09	0,171521	0,049848	1782666730

f	1,783E+09	0,129177	0,059417	1782666730
	1,783E+09	0,065396	0,112871	1782666731
g	1,783E+09	0,139343	0,093749	1782666731
h	1,783E+09	0,153405	0,125034	1782666731

Based on received log, the results of algorithm's operation were visualized. To analyze the system's behavior, the graph was constructed demonstrating the change in the delay value during the sequential input of characters. The ordinal number of recorded event is plotted on *x-axis*, and calculated value of character output delay is plotted on the *y-axis*. The resulting graph makes it possible to assess the nature of the delay change during the algorithm's operation. The absence of constant delay value indicates the dynamic change in the timing characteristics of the output. At the same time, the observed fluctuations correspond to changes in the user's input speed and the influence of the random component generated by the pseudo-random number generator [7]. The results of the graph construction are presented in the Fig. 7.



Fig. 7 – Change of the value of adaptive delay when entering the text

Additionally, the analysis was performed to examine the relationship and dependencies between the intervals in the successive keystrokes and the delay that were calculated by the algorithm [8].

For this purpose, the dotted scatter plot was constructed, reflecting the distribution of the experimental data. The obtained results confirm the existence of the relationship and the dependency between the text input speed and the temporal delay by time for character output, which indicates the correct operation of the developed adaptive algorithm. The corresponding diagram is presented in the Fig. 8.

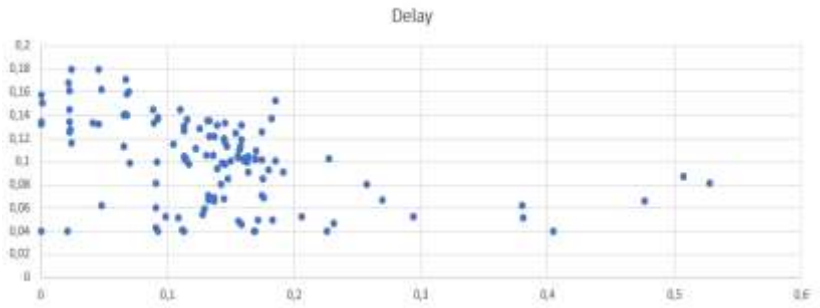


Fig. 8 – The dependence of calculated delay on the interval between consecutive key presses.

Experimental Assessment of the Developed Software Module Effectiveness.

The developed software module was tested on Windows operating system on personal computer using standard keyboard. During the experimental study, text information of varying lengths and typing speeds was entered repeatedly. The purpose of the experiment was to verify the functionality of the adaptive delay algorithm, assess the stability of the software module's operation, and analyze of changes in the time characteristics of character output.

During the experiments, the software module operated in console mode, with automatic displaying of the entered characters and subsequent generation of statistical information about the results of keyboard event processing. The example of the operation of the developed software module is shown in Fig. 9.



Fig. 9 – The result of the operation of software module's in console mode

As can be seen from the provided example, after the input is completed, the developed program automatically generates the summary experiment statistics. It includes the number of processed events, the average values of the intervals between keystrokes and the calculated delays, the maximal size of the processing queue, as well as information about saving the log to the file of CSV format. The obtained data are used for subsequent analysis of the effectiveness of the developed algorithm.

During the experiments, the temporal time parameters of each keyboard input event were recorded automatically, including the moment when the keystroke was registered, the interval between consecutive events, the calculated delay value, and the time of the actual character output. The obtained data were saved in the registration logbook, after which they were processed using Microsoft Excel.

Analysis of experimental data showed that the developed algorithm ensures the dynamic changing and adjustment of delay value depending on the current text input rate. When the typing speed increased, the delay was adjusted automatically and corrected, which led to the change in the temporal structure of the characters output stream. When the input intensity decreased, the delay was reduced, ensuring the comfortable interaction between the user and the developed software [9].

The additional introduction of random component into the calculated delay, made it possible to eliminate the formation of fixed temporal patterns between consecutive events. In the result, the generated sequence of time intervals acquires the stochastic character, which makes it difficult to apply the methods of statistical analysis to recover the individual features of the user's keyboard handwriting. During the tests conducted, no cases of disruption of the character output sequence, loss of keyboard input events, or queue overflow were observed. Using of multithreaded architecture made it possible to separate the processes of event registration and their display, ensuring the stable operation of the software module at various speeds of text input. The obtained results confirm the effectiveness of the developed algorithm and the possibility of its using as software tool for preliminary anonymization of keyboard input temporal characteristics.

Analysis of obtained graphical dependencies shows that the developed algorithm does not form and generate the fixed values of delay but change and adjusts them in accordance with the user's current typing speed. At the same time, the use of moving average values eliminates the abrupt changes in delay, ensuring smooth adaptation of the algorithm to the changes in user's typing speed [10].

The additional introduction of the random component leads to the violation of the strict periodicity of the time intervals between the displayed characters. In result, the statistical characteristics of the output sequence differ from those of the user's original keyboard handwriting, which reduces the possibility of identifying it using methods based on the analysis of typing dynamics.

The conducted experiments demonstrated the stable operation of the developed software module at the various input speeds. No losses of keyboard event or disruptions to the sequence of character output were observed during the tests, which confirms the correctness of the selected multithreaded architecture.

Prospects for the Hardware Implementation of the Developed Algorithm

The developed software module is experimental implementation of algorithm for adaptive changes in the temporal characteristics of keyboard input. Using Python language made it possible to debug the algorithm, conduct experimental studies, and evaluate its effectiveness under various operating modes. The obtained results confirmed the correctness of the developed method functioning and the possibility of its practical application to reduce the informativeness of the user's keyboard handwriting. Next stage in the work development is to transfer the developed algorithm into the hardware platform of the Arduino microcontroller with support for USB interface – Human Interface Device (HID) [3]. Unlike the software implementation that functioning within the operating system, the hardware module can be connected between the keyboard and the personal computer and process the keyboard input events in real time without making any changes to the end user's software [11]. The proposed architecture of the hardware implementation is shown in the Fig. 10.

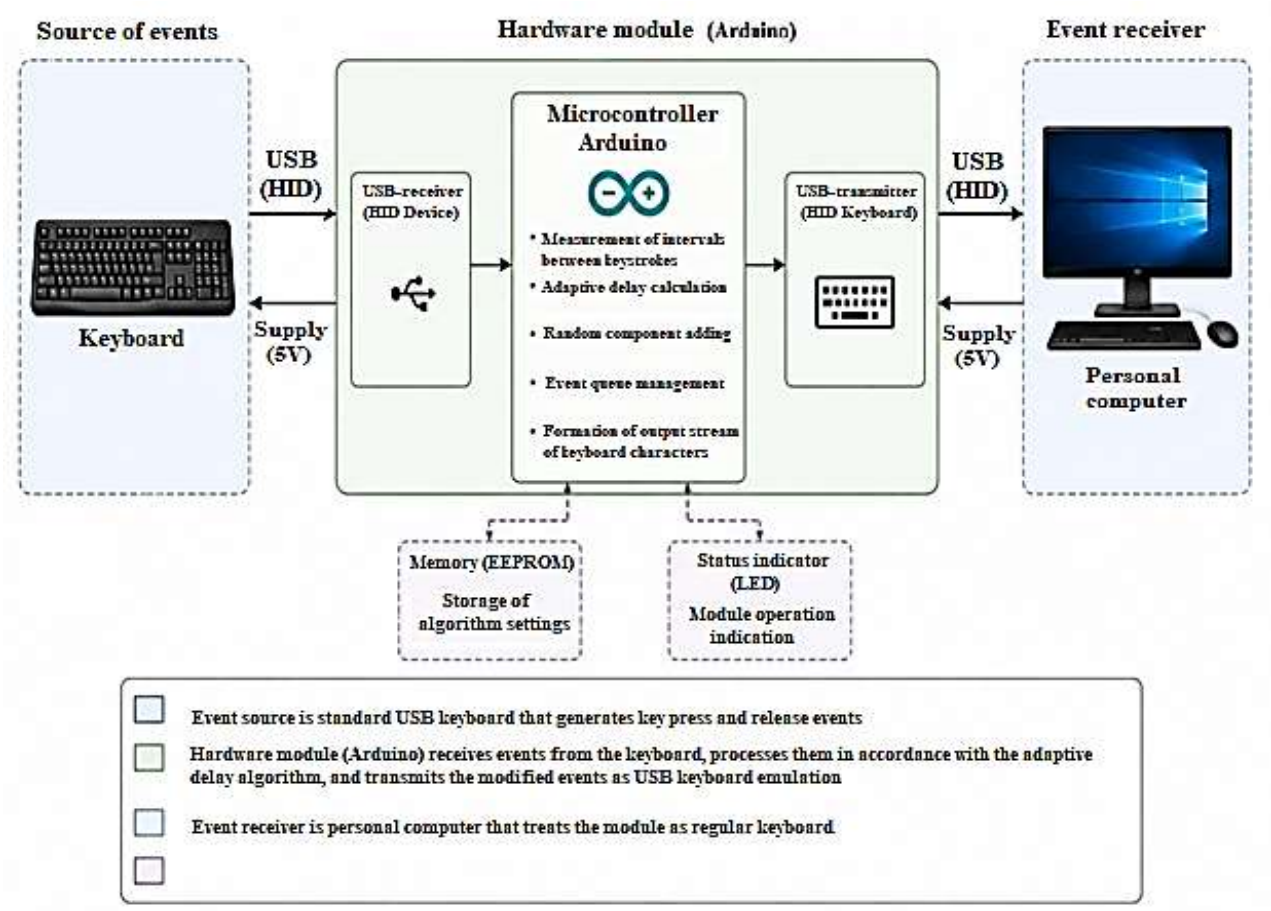


Fig. 10 – Structural diagram of the software-and-hardware module of hardware implementation

In hardware implementation, the overall structure of the developed algorithm is preserved. After receiving the next event of keyboard input, the microcontroller calculates time interval between consecutive keystrokes, determines the value of adaptive delay taking into account the current input speed and the random component, and then transmits the modified event to the operating system by emulating standard USB keyboard. Thanks to this, the use of developed method becomes completely transparent to the software being used and does not require the installation of additional drivers or specialized applications.

Using the Arduino microcontroller makes it possible to implement the standalone device that modifies the temporal timing characteristics of keyboard input regardless of the software platform [12].

It expands the scope of application of the developed algorithm and creates the prerequisites for using the software-and-hardware module as part of specialized information protection tools and systems for enhancing the confidentiality of user interaction with personal computer system.

CONCLUSION

In result of the research, the software module has been developed, designed to modify the temporal characteristics of keyboard input in order to reduce the possibility of identifying the user based on the features of their keyboard typing style – keyboard handwriting. The architecture of software-and-hardware module is proposed, which ensures the registration of keyboard events,

their processing using adaptive delay algorithm, and the subsequent generation of output stream of characters [10 – 12].

The implemented algorithm dynamically change and adjusts the delay value depending on the current text input rate and also uses the random component to form and generate the unpredictable temporal structure of event outputs. The experimental studies conducted confirmed the correct operation of the developed algorithm, the stable performance of the multithreaded architecture, and the absence of event loss under various text input modes. Analysis of the registered and recorded data and the constructed graphical dependencies showed that the use of adaptive delay makes it possible to change the dynamic characteristics of keyboard input without disrupting the sequence of character display. The obtained results confirm the possibility of practical use of developed algorithm as part of software tools for information protection and systems ensuring the confidentiality of user interaction with his personal computer.

The developed software implementation can serve as the basis for creating the standalone hardware device based on Arduino microcontroller, operating in USB HID mode and providing modification of the temporal characteristics of keyboard input in real time without the need to install additional software.

BIBLIOGRAPHY LIST

Python. Python Language Reference. Python Software Foundation.

Arduino IDE. Documentation for the Arduino development environment.

USB Implementers Forum. Universal Serial Bus Device Class Definition for Human Interface Devices (HID). Proceedings of Interface Forum. 2017. Vol. 2. No. 3. Pp. 128–133.

Osin A.V., Murashko Yu. Review of User Identification Methods Based on Digital Fingerprints // Proceedings of Educational Institutions of Communications. 2023. Vol. 9. No. 5. Pp. 91–111. DOI: 10.31854/1813-324X-2023-9-5-91-111.

Zegzhda P.D., Ivashko A.M. Fundamentals of Information System Security. Moscow: Hotline – Telecom, 2018.

Gorbunov A.V., Kuznetsov A.A. Behavioral Biometrics as Means of User Authentication // Information Technologies. 2021. No. 9. Pp. 34–41.

Monrose F., Rubin A.D. Keystroke Dynamics Authentication // Proceedings of the 4th ACM Conference on Computer and Network Security. 2017.

Monrose F., Rubin A.D. Keystroke Dynamics as Biometric Authentication Method // Computer Systems of the Future Generation. 2020.

Pisani P. H., Lorena A. K. Systematic Review of Research in the Field of Key Press Dynamics // Journal of Brazilian Computer Society. 2013. Vol. 19. Pp. 573–587. DOI: 10.1007/s13173-013-0117-7.

Te P.S., Theo A.B., Yue S. Review of Biometric Methods Based on Keystroke Dynamics // The Scientific World Journal. 2013. Article No. 408280. DOI: 10.1155/2013/408280.

Shadman R. et al. Keystroke Dynamics: Concepts, Methods, and Applications. 2023.

Teotia T., Sharma M., at all. Comprehensive Study of the Machine Learning Approaches for Keystroke Dynamics Authentication // International Journal of Biometrics. 2023. Vol. 15. No. 5. Pp. 560–586. DOI: 10.1504/IJBM.2023.133148.