Research Article
Open Access

Secured recorder box (SEREBO) based on block chain technology for immutable data management and notarization

Abstract

Several surveys suggest that as many as 33% of scientists have personal knowledge of a colleague who fabricated or falsified research data. This indicates the need of a system that can aid the assurance that research data is not modified. Block chain technology ensures data authenticity as recorded data is not mutable. In this study, a command-line data recorder and notary service based on blockchain, SEcured REcorder BOx (SEREBO) is presented. SEREBO can help individual scientists or research teams to prove data authenticity after logging data files into the system, and to provide traceable notarization records. Hence, SEREBO a potentially important tool for auditing research data against modifications, and auditing notarization events against backdating or postdating. SEREBO is available for forking at https://github.com/mauriceling/serebo under GNU General Public License version 3 for non-commercial or academic use only.

Volume 7 Issue 6 - 2018

Maurice HT Ling[1,2]
[1]Colossus Technologies LLP, Republic of Singapore, Singapore
[2]HOHY PTE LTD, Republic of Singapore, Singapore

Correspondence: Maurice HT Ling, Colossus Technologies LLP, HOHY PTE LTD, Colossus Technologies LLP. 8 Burns Road, Trivex, Singapore 369977, Republic of Singapore, Tel +65-96669233, Email mawriceling@colossus-tech.com

Received: October 10, 2018 | **Published:** November 20, 2018

Problem scenario and introduction

Consider this question–given a document, such as a letter, how to ensure that the document in question is authentic in terms of both author and date? The usual method is to accept the date on the document and signature of the author (also known as notarization) as proof. Notarization and dating are crucial to render the documents admissible in the court of law.[1] However, the weakest link in this process is the notary–the person signing and dating the document. Firstly, is it possible that the date is incorrect? Kwall and Duhl[2] suggest that backdating a document can either be legitimate or fraudulent, and recommend that the context for the need for backdating should be disclosed. To provide credibility in dating, an independent date stamp is used; such as, the postmark by postal service.[3] Secondly, the authenticity of the signature can be in question due to potential forgery. Recently in Singapore, a 26-year old man had been sentenced to 10 months imprisonment for forging his father's signature for the purpose of bank withdrawals using cheques.[4]

Previously, NotaLogger [5] had been implemented as a service to record the date and time of notarization events. Although this may be sufficient to strengthen notarization of research notebooks, it is inadequate to deal with the problem with data file modifications. It is common in current biological research to generate large numbers of data files from machine readings, images, and intermediates of analytical steps. The usual practice is to adopt various naming systems and filing methods to manage this volume of research results. These methods do not prevent both mislabelling of data files or modification of data. For example, I had generated an Excel file, sampleTimeSeries.xlsx, on 15th May 2015. Three years later, on 10th June 2018, how can I prove that sampleTimeSeries.xlsx had not been modified since 15th May 2015? If I had the intention to change the file on 10th December 2016, I can safely set my computer clock back to 15th May 2015, modify the file and record it as 15th May 2015.

This may not be a hypothetical situation as 33.3% of survey respondents reported personal knowledge of a colleague who fabricated or falsified research data, or who altered or modified research data.[6] This is supported by another survey reporting 29% of the doctoral candidates in Sweden had heard about fabricated data.[7] Similarly, 29.1% of respondents from Croatia observed fellow scientists engaging in data falsification.[8] A suggestion is to use a checksum, such as md5sum generated by MD5.[9] However, checksum can only be used to check for differences between 2 files, which may suggest data corruption, but it does not ensure one of them to be the original file. On the other hand, blockchain technology can be used to prove data integrity between different periods of time.[10] Ichikawa et al.,[11] reported the implementation of a tamper-resistant clinical record management system using blockchain technology.

The fundamental property of blockchain technology is immutability[12]–when data is packed into a blockchain, any modifications to the data is easily detected. The source of this property lies in its use of hash functions. When the first piece of data (D_1) is processed into a blockchain, a hash of the data (H_1) is generated from D_1. Subsequently, when the second piece of data (D_2) is added into the blockchain, the hash at this step (H_2) is generated using a combination of both H_1 and D_{21}.[13,14] This proceeds on to D_N and H_N. Due to the repeated inclusion of the previous hash in the subsequent hash generation (H_{N-1} is used in the generation of H_N), any modifications of preceding data ($D_{1\ to\ N-1}$) can be detected by an audit of hashes, which is usually the regeneration of hashes and compare with the hashes in the blockchain. Data immutability also ensures immutable audit trail and data provenance.[15] In the case of research data, periodic logging of newly generated data files into blockchain can be an important step to prove authenticity.

This immutable property enables blockchain as a suitable mechanism to ensure digital authenticity. For example, Chowdhury et al.,[16] proposed to use blockchain as a means to ensure integrity and

authenticity in medical records and Kleinaki et al.,[17] proposed to use blockchain to ensure integrity and non-repudiation of records. Here, a command-line data recorder based on blockchain, SEcured REcorder BOx (SEREBO), is presented. The term "secured" refers to security against amendments or modifications rather than security against data theft. SEREBO consists of two components for secured data logging – SEREBO Black Box and SEREBO Notary. Several usage scenarios pertaining to bioinformatics will be illustrated before presenting a set of procedures to audit SEREBO.

Architecture and implementation of SEREBO black box and SEREBO notary

SEREBO Black Box is inspired by the black boxes (cockpit voice recorder and flight data recorder) in airliners.[18] The intended purpose is to track and audit research records under the following premise– Given a set of data files, is there a system to log and verify that these files had not been changed or edited since its supposed creation? SEREBO Black Box aims to address this issue by using several approaches. Firstly, the data files can be used to generate a file hash. It is very likely that an edit in the file will result in a different hash. Hence, if a file generates the same hash across two different points in time, it can be safely assumed that the file had not been edited during this time span. Secondly, the file hash must be securely recorded with amendment protection. SEREBO Black Box records the hash and registers the hash into a blockchain. The main concept of blockchain is that the hash of previous (parent) block is concatenated with the data (file hash in this case) of the current block to generate a hash for the current block. Hence, as the blockchain grows, any amendments in earlier blocks can be easily detected – only amendments to the latest block cannot be detected. This property makes the data in blockchain immutable once it is locked within a chain.[12]

SEREBO Black Box is is implemented as a command-line tool using Python 3 and Python-Fire module (https://github.com/google/python-fire), which aims to simplify the implementation of command-line interface in Python 3. The black box is implemented as a SQLite database consisting of 7 tables:

a. Metadata table stores information about the SEREBO Black Box. At creation using init command (see Appendix A for list of commands), date time stamp of creation and a randomly generated 512-character string to represent the identity of the SEREBO Black Box will be recorded.

b. Notary table stores registration record(s) between SEREBO Black Box to one or more SEREBO Notary(ies), registered using register command.

c. System data table stores data and test hashes of current platform using sysrecord command, which is used to provide a data baseline and for future checking for potential differences in processing outcomes due to different platforms.

d. Data log table stores the actual data to be logged and its corresponding hash.

e. Block chain table is the backbone of SEREBO Black Box, where each record/tuple represents a block in the block chain.

f. Event log table stores audit trail of the data logging event when a piece of data is stored in datalog table.

g. Event log datamap table provides the second audit trail of data hash and block chain hashes when a piece of data is stored in datalog table.

When SEREBO Black Box is installed, it should be initialized using init command and record the baseline data using sysrecord command before actual use. Inputs into SEREBO Black Box can be either a file or string. When the input is a string, the data is the actual input string and the description can be used to provide additional information about the string. When a file is given, a series of 12 hashes is generated from the file[19] to reduce the possibility of hash collision.[20–22] The series of hashes is then concatenated to form the data. The absolute and relative file path are added to the description. Hence, both file and string input are reduced to a data component and a description component (Figure 1), which is then used to generate a series of 12 hashes, called Data Hash, after the inclusion of a date time stamp for the input event. The data, description, date time stamp, and Data Hash are recorded in the datalog table.

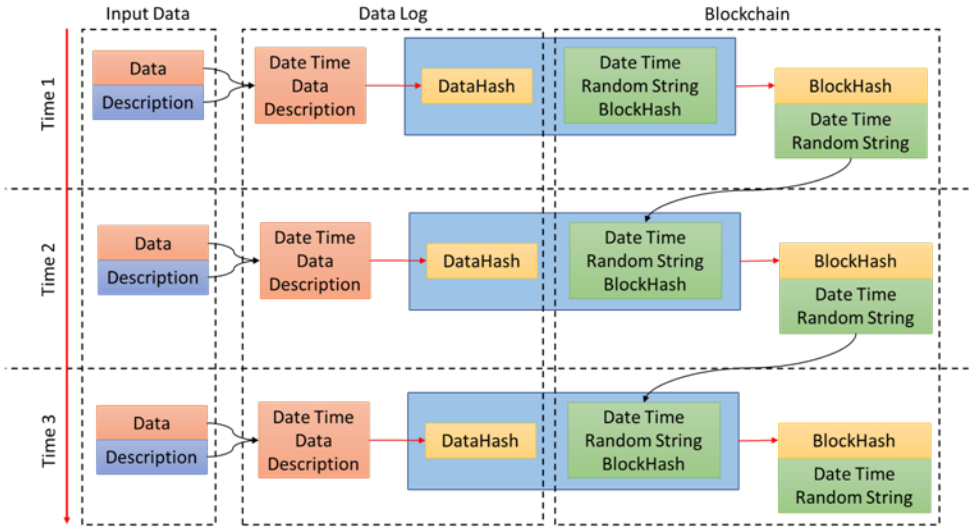

Figure 1 Overview of operations in SEREBO black box.

The latest block in blockchain table is identified and used as the parental block. Three attributes/values from the parental block are extracted–date time stamp (known as parental Date Time stamp), random string (a 32-character random string, known as parental Random String), and block hash (known as parental BlockHash). These 3 parental attributes will be combined with DataHash to generate a BlockHash (known as current BlockHash). A new 32-character Random String (known as current Random String) will be generated and a new block is generated (and will be used as parental block for the next succeeding block), which consists of the following attributes:

1. Date time stamp of the data entry event (identical to date time stamp in the datalog table)
2. Current Random String
3. Current Block Hash
4. Parental block ID
5. Parental Date Time stamp
6. Parental Random String
7. Parental Block Hash
8. Data (which is Data Hash)

Finally, the description and date time stamp are logged in eventlog table. This is followed by logging current Block Hash, parental Block Hash, and Data Hash are logged in eventlog_datamap table.

SEREBO Black Box alone is insufficiently secured from modifications by the fact that it is a standalone unit. An easy method to increase the level of security is to have SEREBO Black Boxes periodically notarized by one or more independent notary/notaries. SEREBO Notary is a web application built on Web2Py framework[23] and exposes a set of XMLRPC web services to provide an independent agent/platform as a notary service. The architecture of SEREBO Notary is based on a previous work, Nota Logger.[5]

To use SEREBO Notary services, the SEREBO Black Box must be registered to at least one SEREBO Notary using register command, which will identify the registration via an alias and the SEREBO Notary will issue an authorization code to the SEREBO Black Box. A SEREBO Black Box can be registered with multiple SEREBO Notaries. An instance SEREBO Notary has been set up for public use and is accessible at https://mauricelab.pythonanywhere.com/serebo_notary/services/call/xmlrpc. All registrations can be viewed using viewreg command. After registration, the most important operation is notarized, which calls on a SEREBO Notary to notarize the SEREBO Black Box. During which, the SEREBO Black Box will generate a local date time stamp (known as Black Box date time stamp) and 32-character random string (known as Black Box code) to be transmitted to the SEREBO Notary, together with identity of the SEREBO Black Box. Upon receiving this set of information, the SEREBO Notary will generate a date time stamp (known as Notary date time stamp) and a 32-character random string (known as Notary code). In addition, the SEREBO Notary will generate a set of hashes (known as common code) from the Black Box code and Notary code. All information will be logged in SEREBO Notary before transmitting Notary date time stamp, Notary code, and common code back to the requesting SEREBO Black Box for logging into datalog table, which will in turn trigger the logging into blockchain. All notarizations can then be viewed using views note command.

Using SEREBO black box as a scientist

Current implementation of SEREBO Black Box can be used to support a scientist's daily work in the 3 major ways:

1. Signing / notarization of research notebooks by superiors or fellow colleagues is increasingly important as notarized research notebooks carry more weight than un-notarized notebooks in patent litigations.[24] SEREBO Black Box can be used to generate a logged notarization code (using local code command) to be appended to the notarization, and dating can be further corroborated. This can be used for notarizing own notebooks or colleague's notebooks.
2. Research data files can be generated daily by a scientist. Some of these data, such as video files from time-lapse microscopy, is not suitable to be appended in physical research notebooks. At the same time, it is also common to receive data files from fellow colleagues. Ideally, these files should be logged in a way that modifications can be easily detected. Given that data falsification often refers to the "pruning and massaging" of existing data to give the desired result,[25] it can be expected that such modifications happen after data collection. SEREBO Black Box can be used to log these data files immediately upon reception from equipment (as close to the data source as possible) using logfile command and the date time stamp can be check against experimental records, as a deterrent against temptations to "prune and massage" existing data at analysis stage to give the desired result.
3. In data-intensive research, such as bioinformatics, data files often must undergo stages of analysis and deception can occur within each analytical stage[26] Similarly, SEREBO Black Box can be used to log all intermediate data files.

Procedures to audit SEREBO

There are three procedures for auditing SEREBO Black Box:

a) Internal Consistency Audit, which checks for accuracy within a given SEREBO Black Box;

b) External/Internal Hash Audit, which checks for consistency between an external hash file and data log hash in SEREBO Black Box;

c) Notary Audit, which checks for records between SEREBO Black Box and SEREBO Notary.

Procedure 1: Internal consistency audit

This checks for accuracy within a given SEREBO Black Box. Although SEREBO Black Box is designed to protect against retrospective modification/amendments, internal consistency within SEREBO Black Box cannot defend against a meticulously doctored SEREBO Black Box with the intention to fool. Here are the steps– Firstly, use audit_count command to check for the number of records between the two main data tables in SEREBO Black Box-Data Log table and Block chain table. The number of records in both tables must be the same as a record into Data Log will generate a corresponding record in Block chain. If the number of records is the same, it will further check for date time stamps between the corresponding records in these two tables, which must be the same. If either of these two checks failed, the SEREBO Black Box in question is internally inconsistent.

Secondly, use audit_datahash command to check for consistency

in the data log hash, given the data in other fields in Data Log table. When a log event is called to SEREBO Black Box, the logged data will be used to generate a corresponding data log hash and stored within the same record as the logged data. Hence, this step repeats the data log hash generation from the logged data and compares the newly generated data log hash with the stored data log hash in Data Log table. If the newly generated data log hash with the stored data log hash are different, the SEREBO Black Box in question is internally inconsistent; which can either suggests that a modification in the logged data has occurred or a corruption in SEREBO Black Box.

Thirdly, use audit_data_blockchain command to check for consistency in data log hash (in Data Log table) with the data in Blockchain table. In a log event, the generated data log hash (generated from the logged data) is recorded as data in Blockchain table. Hence, both tables should record the same data log hash. If different data log hashes are recorded in both Data Log table and Blockchain table, the SEREBO Black Box in question is internally inconsistent.

Fourthly, use audit_blockchainflow command to trace the dependency of block chain records (also known as blocks). Blockchain table consists of seven crucial fields-

a. Current date time stamp, which records the date time of log event;

b. Random string generated for current event;

c. Current hash;

d. Date time stamp of the latest pre-existing event, which is the parent date time stamp;

e. Parent random string;

f. Parent hash;

g. Data, which is the data log hash.

Each blockchain record represents a block. Hence, a line of block dependency can be established by tracing the current block's data to the child/descendant block. If this trace cannot be established, the SEREBO Black Box in question is internally inconsistent.

Lastly, use audit_blockchainhash command to check for internal consistency in Blockchain table. The current hash is generated from four items - (a) parent date time stamp, (b) parent random string, (c) parent hash, and (d) data. Hence, if the hash generated from the four items (parent date time stamp, parent random string, parent hash, and data) differs from the recorded hash, the SEREBO Black Box in question is internally inconsistent.

Procedure 2 External/internal hash audit

This procedure checks an exported data log hash, from Data Log table into a file, against the data log hash within a SEREBO Black Box. Procedurally, use dumphash command exported data log hash, from Data Log table into a file (known as hash file) with the following format: record ID|date time stamp|hash. Command check hash can then be used to check the data log hashes in SEREBO Black Box against a previously generated hash file.

Since hash file only contains the date time stamp and the corresponding record hash rather than the actual logged data, generating hash file does not compromise data confidentiality as it is practically near impossible (takes far too long with current computing system) to reverse hash. Hence, it is possible to put the hash file in a public code repository or code versioning system; such as Git Hub (https://github.com), Git Lab (https://about.gitlab.com/), Source Forge (https://sourceforge.net), or Bit Bucket (https://bitbucket.org); where versioning can be used to verify and date the earliest presence of a log event. Therefore, external / internal hash audit is a substantially stronger version of audit_datahash.

Procedure 3: Notary audit

Notary audit is to check for records of notarization(s) by SEREBO Notary(ies) based on the records in a SEREBO Black Box. This is analogous to taking the signature(s)/notarization(s) of an independent person (such as a Notary Public) to the said person for verification. There are 2 steps to this procedure. Firstly, each Black Box registration with a Notary must be verified. The registration(s) for a Black Box can be viewed using viewreg command. If there is more than one registration, each registration should be checked and verified with audit_register command to ensure that all registrations are valid – the status should show that *Registration found in SEREBO Notary*. Secondly, each notarization by SEREBO Notary(ies) must be verified. A list of notarizations by one or more SEREBO Notary(ies) can be obtained using viewsnnote command, which can then be verified with the respective SEREBO Notary(ies) using audit_notarizebb command. As many registrations and notarizations should be verified as possible.

Conclusion

Data fabrication / falsification and purposeful miss-dating of notarization are serious scientific problems which may have legal implications.[27] The line between honest error and research misconduct is often fussy and a scientist absolved of such accusations is likely to suffer from psychological stress and damaged reputation.[28] Hence, it is advisable to prevent such accusations by having auditable research records. SEREBO is based on blockchain technology, which can help individual scientists or research teams to prove data authenticity after logging data files into the system, and to provide traceable notarization records. Although SEREBO cannot prevent wilful research misconduct, it can increase confidence in data integrity and authenticity, which supports the argument against research misconduct when used appropriately.

Availability

SEREBO is available for forking at https://github.com/mauriceling/serebo under GNU General Public License version 3 for non-commercial or academic use only. Installation instructions are available at https://github.com/mauriceling/serebo/wiki/Obtaining-and-Installing-SEREBO.

Acknowledgements

None.

Conflict of interest

The author declares that there is no conflict of interest.

Appendix

Appendix A–List of Operations

1. SEREBO Black Box Operations

1.1. backup: Backup SEREBO Black Box

1.2. dump: Dump out data (text backup) from SEREBO Black Box

1.3. fhash: Generate and print out hash of a file

1.4. init: Initialize SEREBO Black Box

1.5. intext: Insert a text string into SEREBO Black Box

1.6. localcode: Generate a random string, and log this generation into SEREBO Black Box

1.7. localdts: Get date time string

1.8. logfile: Log a file into SEREBO Black Box

1.9. ntpsign: Self-sign (self-notarization) SEREBO Black Box using NTP (Network Time Protocol) Server

1.10. searchmsg: Search SEREBO Black Box data log for a message

1.11. searchdesc: Search SEREBO Black Box data log for a description

1.12. searchfile: Search SEREBO Black Box data log for a file log

1.13. selfsign: Self-sign (self-notarization) SEREBO Black Box

1.14. shash: Generate hash for a data string using SEREBO Black Box

1.15. sysdata: Print out data and test hashes of current platform

1.16. sysrecord: Record data and test hashes of current platform into SEREBO Black Box

1.17. viewntpnote: View all self-notarization(s) by NTP time server for this SEREBO Black Box

1.18. viewselfnote: View self-notarization(s) for current SEREBO Black Box

2. SEREBO Notary Operations

1.1. changealias: Change alias for a specific SEREBO Notary registration

1.2. notarizebb: Notarize SEREBO Black Box with SEREBO Notary

1.3. register: Register SEREBO Black Box with SEREBO Notary

1.4. viewsnnote: View notarization(s) by SEREBO Notary(ies) for current SEREBO Black Box

1.5. viewreg: View current SEREBO Notary registrations for current SEREBO Black Box

3. Audit Operations

1.1. audit_blockchainflow: Trace the decendency of blockchain records (also known as blocks) within SEREBO Black Box

1.2. audit_blockchainhash: Check for accuracy in blockchain hash generation within SEREBO Black Box

1.3. audit_count: Check for equal numbers of records in data log and blockchain in SEREBO Black Box

1.4. audit_data_blockchain: Check for accuracy in data log and blockchain mapping in SEREBO Black Box

1.5. audit_datahash: Check for accuracy of hash generations in data log within SEREBO Black Box

1.6. audit_notarizebb: Check for SEREBO Black Box notarization records in SEREBO Notary

1.7. audit_register: Check for registration between SEREBO Black Box and SEREBO Notary

1.8. checkhash: Compare record hash from SEREBO Black Box with that in a file

1.9. dumphash: Dump out record hash from SEREBO Black Box into a file

References

1. Marina CG. *Complainant vs. Atty. Calixto B. Ramos. A.C. No. 6649*. Republic of the Philippines: Supreme Court; 2005.
2. Kwall JL, Duhl S. Backdating. *The Business Lawyer*. 2008;63(4):1153–1186.
3. RT v Parliament. EUECJ T-98/17. *Court of Justice of the European Communities*; 2018.
4. https://www.straitstimes.com/singapore/courts-crime/man-jailed-for-forging-fathers-cheques-making-off-with-27000.
5. Ling MH. NotaLogger: Notarization Code Generator and Logging Service. *The Python Papers*. 2014;9(1).
6. Fanelli D. How many scientists fabricate and falsify research? A systematic review and meta-analysis of survey data. *PLoS ONE*. 2009;4(5):e5738.
7. Hofmann B, Myhr AI, Holm S. Scientific dishonesty:a nationwide survey of doctoral students in Norway. *BMC Med Ethics*. 2013;14:3.
8. Pupovac V, Prijic-Samarzija S, Petrovecki M. Research Misconduct in the Croatian Scientific Community: A Survey Assessing the Forms and Characteristics of Research Misconduct. *Sci Eng Ethics*. 2017;23(1):165–181.
9. Rivest R. *The MD5 message-digest algorithm*. MIT Laboratory for Computer Science and RSA Data security Inc.; 1992.
10. Gammon K, Experimenting with blockchain: Can one technology boost both data integrity and patients' pocketbooks? *Nature Medicine*. 2018;24(4):378–381.
11. Ichikawa D, Kashiyama M, Ueno T. Tamper-Resistant Mobile Health Using Blockchain Technology. *JMIR mHealth and uHealth*. 2017;5(7):e111.
12. Zheng Z, Xie S, Dai H, et al. *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*. In: 2017 IEEE International Congress on Big Data (BigData Congress); 2017. 557–564 p.
13. Puthal D, Malik N, Mohanty SP, et al. The blockchain as a decentralized security framework. *IEEE Consum Electron Mag*. 2018;7(2):18–21.
14. Narayanan A, Bonneau J, Felten E, et al. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press; 2016.

Citation: Ling MHT. Secured recorder box (SEREBO) based on block chain technology for immutable data management and notarization. *MOJ Proteomics Bioinformatics*. 2018;7(6):169–174. DOI: 10.15406/mojpb.2018.07.00253

15. Kuo TT, Kim HE, Ohno Machado L. Blockchain distributed ledger technologies for biomedical and health care applications. *J Am Med Inform Assoc*. 2017;24(6):1211–1220.

16. Chowdhury MJM, Colman A, Kabir MA, et al. *Blockchain as a Notarization Service for Data Sharing with Personal Data Store*. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). New York, United States; 2018. 1330–1335 p.

17. Kleinaki AS, Mytis Gkometh P, Drosatos G, et al. A Blockchain-Based Notarization Service for Biomedical Knowledge Retrieval. *Computational and Structural Biotechnology Journal*. 2018;16:288–297.

18. Pierce A. The Evolution of the Airplane Black Box. *Tech Directions*. 2010;70(2):12.

19. Ling MH. *A Cryptography Method Inspired by Jigsaw Puzzles*. Nova Science Publishers; 2018. 129–142 p.

20. Rasjid ZE, Soewito B, Witjaksono G, et al. A review of collisions in cryptographic hash function used in digital forensic tools. *Procedia Computer Science*. 2017;116:381–392.

21. Boneh D, Boyen X. On the impossibility of efficiently combining collision resistant hash functions. *Springer-Verlag*; 2006. 14 p.

22. Broder A, *Mitzenmacher M. Using multiple hash functions to improve IP lookups*; 2001. 1454–1463 p.

23. Pierro MD. *Web2Py: Enterprise Web Framework*. New Jersey: John Wiley & Sons, Inc; 2008.

24. Nickla JT, Boehm MB. Proper laboratory notebook practices: protecting your intellectual property. *J Neuroimmune Pharmacol*. 2011;6(1):4–9.

25. Bornmann L. Research Misconduct: Definitions, Manifestations and Extent. Publications. 2013;1(3):87–98.

26. Resnik DB, Stewart CN. Misconduct versus honest error and scientific disagreement. *Account Res*. 2012;19(1):56–63.

27. Collier R. Scientific misconduct or criminal offence? *CMAJ*. 2015;187(17):1273–1274.

28. Resnik DB, Elliott KC, Soranno PA, et al. Data Intensive Science and Research Integrity. *Account Res*. 2017;24(6):344–358.