Blockchain-enabled and Data-driven Smart Healthcare Solution for Secure and Privacy-Preserving Data Access

Mohamed Younis*, Wassila Lalouani*, Noureddine Lasla‡, Lloyd Emokpae§, Mohamed Abdallah‡,
* Dept. Computer Science & Elec. Eng., University of Maryland, Baltimore County, Baltimore, MD, USA
‡Division of Information and Computing Technology, College of Science and Engineering, HBKU, Doha, Qatar
§LASARRUS Clinic and Research Center LLC., Baltimore, Maryland, USA
Email: {younis, lwassil1}@umbc.edu; {nlasla, moabdallah}@hbku.edu.qa; lloyd.emokpae@lasarrus.com

***Abstract*—The major advances in body-mounted sensors and wireless technologies have been revolutionizing the healthcare industry, where patient's conditions can be remotely monitored by medical staff. Such a model is gaining broad support due to its economic and social advantages. However, the wealth of sensor measurements pose major technical challenges on where to store the collected data, how to ensure its integrity, who control access permissions, and how to enable secure interaction between patients and medical facilities and professionals. This paper aspires to provide a holistic solution based on Blockchain technology. Our solution puts the patient in charge for granting and revoking access permissions and makes it easy for healthcare organizations and providers to meet privacy regulations. The sensor data is to reside on cloud storage while access control and session logs are maintained on Blockchain. In addition, a novel data-driven authentication and secure communication protocol is proposed to mitigate the risk of fraud and identity theft. In order to enforce such a protocol, all interactions between the cloud and patients, and healthcare providers are regulated through smart contracts. The security properties of our solution are analyzed using AVISPA; it is also shown to be computationally efficient.**

***Index Terms*—Smart healthcare, blockchain, data-driven secure communication, key management, authentication.**

I. Introduction

Motivation: Recent years have witnessed major technological advances in the development of wearable sensors, both body-mounted and implants. These sensors enable collection of a broad range of vital measurements, e.g., pulse rate, and temperature, and non-vital measurements such as blood pressure, and electrocardiogram (ECG). By incorporating radio transceivers, wearable sensors allow the collected data to be checked in real-time by a remote medical professional [1]. Such a model, is often referred to as tele-health, allows patients and people at risk to be continuously monitored while living normally and alleviates the demand for hospitalization and clinic visits. Overall, tele-health applications are becoming more popular given the rising cost of healthcare. Such popularity is even expected to increase massively given the COVID-19 pandemic, where access to medical facilities has been constrained due to the contagious disease [2].

The challenge in supporting such a transformed healthcare system will be how to manage access to medical data given the volume of recorded sensor measurements and how to protect the privacy of patients [3]. Basically, the increased sophistication of wearable medical devices and their ability to practically monitor patients at all times will lead to the generation of a large volume of data. Some of such data cannot be consumed instantaneously since variations over time is what matters; also the data would often need to be stored to track the patient history. Furthermore, the data may be tracked by multiple healthcare providers that are not necessarily associated with the same organization. Not only hosting the data is an issue, but also imposing access and update restrictions will be a challenge. It is necessary to respect the patient's right of determining who is allowed to access the data. Cloud storage, such as Amazon AWS, would be a prime choice since the maintenance overhead will be offloaded and high availability will be provided in a seamless way to both the patients and healthcare providers.

Challenges: Nonetheless, additional provisions are required to ensure data integrity and confidentiality, and put the patient in charge of authorizing access to the data. An attacker can be someone who wants to steal the identity of a patient, or use patient data for fraud or blackmail. For example, the data could be used to fool an insurance company to pay for expenses of faked services. Moreover, the data could also be used to unjustly raise premiums and diminish the competitiveness of certain health insurance providers. To achieve these objectives, the attacker may eavesdrop on data transmissions: (i) between the devices (patients) and the storage system, or (ii) between the storage system and the authorized medical professionals. An attacker could also try to impersonate a medical professional to gain access to the stored patient records and potentially manipulate the data.

The use of encryption keys is the conventional approach to achieve the desired protection; yet has the following shortcomings: (i) advanced computational architectures have increased the success probability of cryptanalysis, and thus frequent changes of encryption keys will be necessary, something that will be quite burdensome for patients and medical professionals unless it is automated, and (ii) access to the data may be needed by multiple medical professionals; using distinct keys for each professional will be unrealistic for a patient, while using the same key will elevate the risk of cryptanalysis. Furthermore, a medical professional is generally monitoring the health of multiple patients, ranging from tens to thousand, which make the management of the respective patient's encryption keys a very complex task that is prone to key theft, and consequently jeopardizes the privacy and the security of the whole system. Therefore, the use of dynamic encryption keys that are generated automatically is of paramount importance in tele-health systems. A cloud storage system enables a user (a patient) to authorize access to his/her own data, yet provides little support for key management and potentially constitutes a single point of failure. Also it does not counter identity thefts through hacking usernames and passwords, or even stealing crypto-identities.

Contribution: This paper opts to fulfill the requirements and tackles the shortcomings of contemporary solutions by developing an effective and robust data storage and access control system for tele-health applications. The proposed system promotes a novel data-driven mechanism for safeguarding the integrity of the patient's data and authenticating the identity of the healthcare providers who are given access permissions. Unlike the conventional approach of using a session key for encrypting transmitted data, our system varies the key per packet in a coordinated manner between the communicating pairs, namely, the patient and storage host, and the healthcare provider and storage host. Such coordination is enabled by a novel machine learning model that uses previously transmitted data to generate new encryption keys. Such a time-varying data-driven key generation mechanism allows the system to be very robust against eavesdropping and cryptanalysis attempts.

Our solution employs Blockchain technology, particularly smart contracts to define access rules, store session logs and maintain primitives for authenticating the identity of patients and healthcare providers. The incorporation of Blockchain allows users to define access rights and supports fault-tolerance for such a critical function. As a secure distributed ledger, Blockchain will also enable billing, accounting, and handling of malpractice disputes by logging session activities. Our system prevents the central role of the cloud from making the system vulnerable to a single point of failure. Even if the cloud becomes subject to a successful intrusion attempt, the adversary cannot benefit from the compromised packet keys as they are only for one-time use and depend on the data. Moreover, the adversary cannot actually capture the data since the cloud agent does not know the patient's storage key.

The contribution of the paper can be summarized as follows:

- Develop novel Blockchain-based architecture for tele-health applications. The role of cloud services is limited to supporting secure storage and retrieval of patient's data. The use of Blockchain enables decentralized access control and logging.
- Develop a data-driven security mechanism to safeguard transmission of patient's data to and from the cloud.
- Develop an agile mechanism for enforcing patient's access rules and authenticating authorized users. Such a mechanism leverages smart contracts to enable personalized configuration per patient and healthcare provider.
- Verify the security of our system using the AVISPA toolset [4].
- Validate the performance in terms of the computational complexity and the strength of cryptographic keys.

The next section provides an overview of the state of the art and highlights the technical gaps. Section III discusses the required features and provides an overview of the proposed system. The detailed design of our system can be found in Section IV. Section V analyzes the security properties of our proposed system. The computational overhead is studied in Section VI. Finally the paper is concluded in Section VII.

II. Related Work

We propose a holistic security and privacy solution for tele-health systems where access to sensor data is authorized only by the patient, data transmission is secured against eavesdroppers using data-driven encryption keys, patient's data is stored and exchanged only in an encrypted form, authentication and access control are managed in a decentralized manner, and access log is maintained through a high integrity ledger. Existing work on tele-health data management can be categorized based on the scope into: (i) data storage handling without consideration of security issues [5], (ii) defining biometric fingerprints to enable authentication and data ownership [6], and (iii) security and privacy aware data storage and access architectures [7]. Given our contribution we focus on the third category and the use of biometrics in key generation.

Facilitating secure data access for tele-health systems has been subject to a number of studies in recent years [8]. These studies pursued one of the following three configurations: (1) employ the cloud as data storage and applying conventional single/multi-factor authentication to control access [9], or attribute-based encryption to enable search and query of encrypted patient data [10], (2) use Blockchain to store data and manage access privilege [11], [12], and (3) use the cloud to store the data and Blockchain to handle security and privacy [13], [14], [15]. The first configuration requires direct interaction between a patient and a caregiver for each session, which is often impractical from a scheduling point of view; it also puts the burden on the patient for tracking the credential of caregivers. Meanwhile, the second configuration imposes excessive storage and management overhead since the data will be replicated and consensus has to be reached for writing each data sample. Our approach pursues the third configuration.

The use of Blockchain for authentication and access control has been exploited in multiple tele-health application. Most published architectures have put the Blockchain network as a layer between the users and cloud resources [13], or among multiple healthcare providers [14]. Such an approach is not efficient since Blockchain becomes an intermediary for establishing user connections and constitutes a bottleneck; we avoid such a shortcoming and use Blockchain for session logging and access control rules; thus, Blockchain will be engaged only when a valid user credentials are presented to the cloud and additional level of authentication/authorization is to be applied. Like [14], Nguyen et al. [15] uses Blockchain for sharing patient's data among healthcare facilities; however, the patient is kept out of the loop. Some work also used Blockchain to ensure integrity of patient's health records [16].

Shamshad et al. [17] use Blockchain to achieve security and privacy goals. Every medical facility, e.g., a hospital, sets up a private Blockchain using its own computer, where data is stored in an encrypted form using the patient's private key. In addition, a consortium Blockchain is to be established between all medical facilities to simplify search for patient's data using search-able keywords. A proxy re-encryption (PRE) technique is proposed, where the search results are re-encrypted such that the user can retrieve the data without revealing the private key of the patient. However, such an approach suffers serious shortcomings. First the private Blockchain will require excessive storage by imposing unwarranted replication of patient's data; since the execution of consensus protocol is not needed, a simple database would suffice. Second, access to patient's data involves a computationally-heavy asymmetric encryption scheme, i.e., PRE.

Manzoor, et al. [18] puts Blockchain in charge of user authentication and key management. The considered application assumes that the data generated by a sensor is accessed by a user in return for a fee, where the Blockchain also manages transaction recording and billing. The data is stored on the cloud in an encrypted form, yet when access is approved and paid for by a user, the Blockchain instructs the cloud agent to re-encrypt the data using a PRE scheme and store it for the user to download. However, the use of such an asymmetric crypto-system for all relevant operations imposes excessive overhead. Finally, the Blockchain is involved based on the justification of handling sales of data (user accounts); such justification does not apply in the realm of tele-health applications.

Given that tele-health systems involve sensors that are attached or implanted in the human body, some work has used biometric signatures of patients as fingerprints for authenticating the source/owner of the data [6] , or to include watermarks [19]. Other studies have utilized the biometric signals to generate encryption keys [20], [21]. Unlike prior work, our approach authenticates users and generates encryption keys based on multiple consecutive data samples, which introduces more variability and makes the crypto-system more robust against attacks. In addition, our approach employs advanced deep networks, specifically, LSTM (Long Short-Term Memory), which factors in both the data sequence and values, and thus it is impossible for an eavesdropper or unauthorized user to guess the key without mimicking the entire process and using the same LSTM parameters.

III. System Model and Approach Overview

A. System Requirements and Design Goals

A typical operation of a tele-health system involves interaction with both patients and healthcare providers. The relationship is fundamentally many-to-many, where one patient could be monitored by more than one caregivers; similarly, a physician or medical facility is serving multiple patients. Moreover, the various caregivers could be interested in certain sensor modality and/or data collected at specific time of the day, or when the patient is doing specific activities, e.g., during sleep hours. Therefore, direct pairwise connections between a caregiver and patient's wearable sensors would be impractical, and an intermediate entity is needed to buffer/store the data and serve the individual caregivers based on their interest. On the patient side, sensor measurements will be streamed, through a gateway, to a secure storage facility. Such a gateway node has significantly more communication and computation resources than the wearable sensor nodes [22]. A patient will also specify which healthcare providers are allowed to retrieve the stored measurements. Accordingly, the storage facility will provide those authorized healthcare providers access to the data.

1) System Requirements: The growing adoption of wearable technologies and the advantages of tele-health services motivate the development of a holistic solution that supports:

- *Flexibility*: People switch their primary care physicians and testing facilities over time. This could be due to personal preference and experience, emerging health conditions that necessitate consultation with specialists, or changes in health insurance coverage and providers. Thus, a tele-health system should facilitate adjustment in access permissions to adapt to evolving patient associations with the various caregivers.
- *Dependability*: Tele-health constitutes a mission-critical service since it affects the well-being of people. Therefore, ensuring robust and secure data storage and retrieval is a core requirement. The system should avoid the presence of a single point of failure and should be able to simultaneously serve multiple patients and healthcare providers. The patient's data should also be safeguarded against attacks while being stored and shared with authorized users.
- *Scalability*: With the prevalence of wearable technologies a patient could have multiple body-attached or implanted sensors; each is collecting data at various rates. Thus, the storage requirement is high, especially for a large patient population. Moreover such diverse data modality could be examined by multiple physicians and/or medical facilities. Thus, maintaining access rights and managing confidentiality primitives, e.g., encryption keys, are quite challenging.

2) Design Goals: From a security point of view, the tele-health system should sustain privacy and enables data access based on patients' consent. The system should thus ensure that the data cannot be intercepted during transmission while being collected from a patient and being sent to a caregiver. Moreover, the data should also be stored in encrypted form to prevent privacy breaches through intrusion. Specifically, the system should counter the following forms of attacks:

(i) Eavesdroppers who intercept transmissions from a patient to the storage facility and from the storage facility a doctor or hospital, in response to a legitimate data request.
(ii) Identity theft for an owner or an authorized user of data.
(iii) Breaches to the storage system for retrieving or modifying the data records.

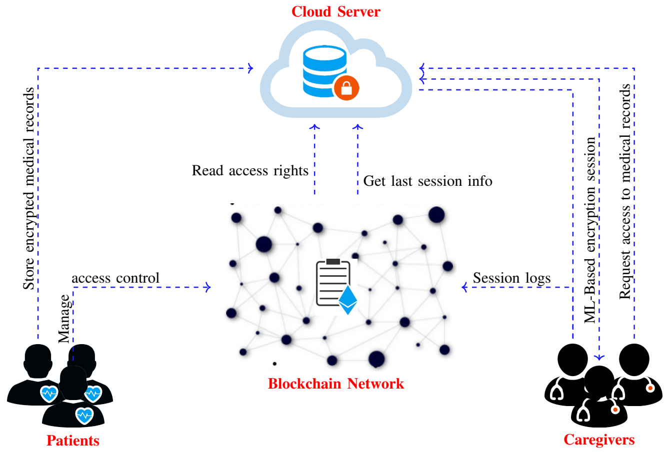


Fig. 1: An overview of the BeDaSH architecture.

Finally, the security solution should withstand increased and sustained attack attempts. Particularly, the major growth in computational power in recent years poses a threat to cryptosystems. The tele-health system should stay robust to crypt-analysis while at the same time continue to be conscious about resource constraints at the communication ends, specifically the limited capabilities of wearable sensors on the patient's side and the fact that caregivers may be using portable personal devices, e.g., smart phones, to access patients data.

B. Approach Overview

The main objective of our proposed Blockchain-enabled Data-driven access control and management system for Smart Healthcare applications (BeDaSH) is to bedash, i.e., ruin, attacker's attempts to intercept, retrieve and manipulate patient's data. Fig. 1 shows an articulation of the overall system architecture and highlights the various components and players. There are four players, namely, the patients, Blockchain, storage system and caregivers. The patient's data is stored in an encrypted form using a patient-picked key. An agent at the cloud will serve as the interface to the users (both data owners and requesters). A smart contract at the blockchain will enable the data owner to specify access rights for authorized caregivers. The blockchain also maintains session logs to facilitate billing, track caregiver access, and provide primitives for future data-driven authentication. No data is stored on the Blockchain to ensure user privacy. All security provisions are applied by the cloud agent. We note that BeDaSH is concerned with the security architecture, which is fully decentralized. The storage of the patient data can be on either a centralized or decentralized platform. In other words, the cloud agent in Fig. 1 is a functional module and could be regulating access to a centralized server or a distributed data center.

BeDaSH opts to enable: (i) robust, high-integrity and privacy-persevering storage, and (ii) secure and controlled access to the stored data. To achieve these goals, BeDaSH employs ML-based encryption key generation and Blockchain technology with smart contracts. Blockchain will mainly serve as a means to decentralize access control management by giving data owners (patients) the exclusive right to grant and revoke access permission to their data. Thanks to the smart-contract functionality, Blockchain can empower patients to efficiently define and update conditions for data retrieval from the cloud. Blockchain also stores logging information to ensure the integrity and ordering of data records, which are invaluable for supporting streamed data from wearable sensors, managing accounting and billing, handling services disputes, and investigating malpractice cases. Both public and consortium blockchains can be used by our system [23]. The use of public blockchain will not require building a new specific blockchain network for tele-health, and well known public platforms, such as Ethereum can be directly used, instead. However, the use of a public blockchain will incur a fee for each submitted transaction. On the other hand, the use of consortium blockchain can guarantee free transactions, but a special setup is needed to design and build the consortium network among participants.

In our proposed BeDaSH system, the patient's data is stored on the cloud rather than in the Blockchain. The reason is that storing the data on the Blockchain implies replication of potentially voluminous data which is not resource efficient and also raises concerns about privacy [24]. Therefore, the architecture of BeDaSH decouples the data storage from the Blockchain in order to enable scalable operation, efficient resource usage and privacy. All patient's data will be stored on the cloud while Blockchain will be supporting user authentication, managing access rights and defining the security parameters for the data transmission protocol. Nonetheless, Blockchain will maintain a log of user access and index to the data records in order to expedite data retrieval and update. Such a capability safeguards the patient's data against manipulation. In addition, by leveraging smart contracts, a patient will have full control on defining data access policies and granting authorization permissions to certain healthcare providers. The access policies can be based on the patient's privacy preferences. Such flexibility and high integrity features are not supported by conventional cloud-based storage systems.

BeDaSH achieves data confidentiality through three provisions: (i) encrypted storage, (ii) robust user authentication, and (iii) encrypted transmission using time-varying data-driven keys. First, all the patient's data will be stored on the cloud in an encrypted form using a patient-picked key. Symmetric encryption is pursued for such a storage purpose since the data could be accessed by multiple caregivers; by using symmetric keys the involvement of the patient in the data retrieval process will be limited to sharing the storage key with authorized caregivers and data gets directly sent from the cloud to the physician or medical facility. We note that the storage key is not shared with the cloud agent. User authentication in BeDaSH, on the other hand, involves asymmetric cryptography. Each patient and caregiver is identified by a public-private key pair and a Blockchain address. Each patient will maintain a list of user (caregiver) identifiers to be authorized for data access. Such a list is patient-specific and is part of the individualized smart contract of the patients. When access is requested, first the identity of the user will be authenticated

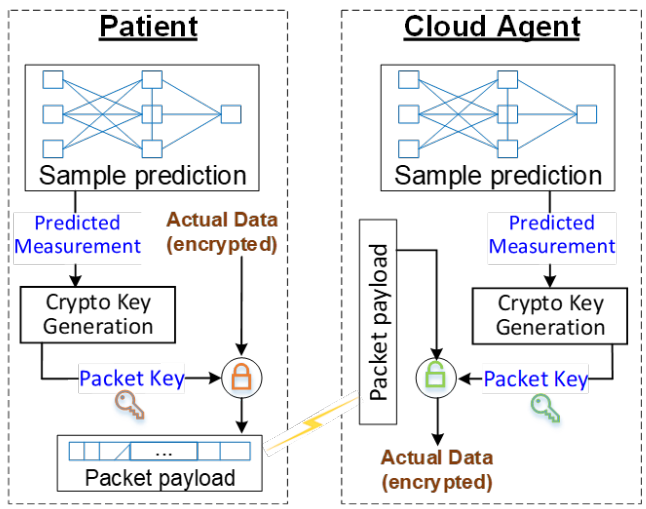


Fig. 2: Overview of the data transmission between the patient and the cloud agent (storage system) according to BeDaSH. A similar process is followed when the storage system responds to data access request made by an authorized healthcare provider.

using its digital signature. This applies for both patients and healthcare providers. However, for the latter, an additional check with the smart contract is made to ensure that such a provider is indeed authorized by the patient who owns the data. In order to mitigate the threat of identity theft, BeDaSH further employs an additional verification step where the most-recently accessed data items by a user are factored in to generate a data-driven user signature, as explained below.

Data is transmitted from the patient to the cloud using a novel data-driven encryption mechanism. At the patient side, e.g., at the gateway node, BeDaSH employs a deep network model that considers the sensor samples as a time series. Based on the previous $m-1$ samples, the model predicts what the next, m^{th}, sample is. Such a predicted sample will be further used to generate a key K_D^m for encrypting the next packet, which includes the actual value of the m^{th} sample. The access agent at the cloud side will apply the same process, to regenerate K_D^m so that it can decrypt the packet. Such a process, which is articulated in Fig. 2, is very powerful since in essence every packet could be encrypted by a distinct key. The fact that the communicating parties are able to implicitly agree on the keys, based on prior data transmissions, enables successful retrieval of the data while making it almost impossible for an eavesdropper to succeed in uncovering the data. Cryptanalysis fundamentally is not applicable since there are not sufficient intercepted packets that use the same key where every packet will use a different one.

The patient feeds the deep network with an encrypted version of the samples; for that a patient-picked storage key, K_P^{store}, is used. Such a key is not shared with the cloud in order to protect the patient's privacy; yet the patient gives K_P^{store} to authorized caregivers based on a confidentiality agreement. After retrieving the (encrypted) patient data from the received packet, the access agent will save it, as is, on the cloud. A similar process is applied when a healthcare provider requests data of a certain patient. We note that unless the provider is given the parameters of the deep network and K_P^{store}, data cannot be decrypted. This provision enables BeDaSH to counter the threat of identity theft and is thus used for authentication as noted above. The smart contract at the Blockchain will log the last set of data samples accessed by a user in a session; such data is provided by the cloud agent, i.e., in the encrypted form stored by the patient, in order to sustain privacy. When a request is made, the cloud agent retrieves the last accessed data samples from the smart contract to apply the aforementioned authentication procedure.

The data-driven keys suffice for sustaining data integrity and confidentiality during transmission against an external attacker who may know the approach but does not know the exact parameters, i.e., had no prior access to the data. In order to achieve forward secrecy and protect the patient data from access by a previously authorized caregiver, BeDaSH includes a random number in the key generation process. The seed for a pseudo random number generator is provided by the cloud agent at the time of session establishment. In essence, the generated random numbers constitute a time series so that a former user of the system cannot predict the packet key despite knowing some data records. Thus, a former caregiver will not succeed in intercepting and decoding messages of other users. In addition, the random number will further ensure packet key variability in case the patient's sensor measurements do not significantly change, e.g., staying at normal levels, which leads to having a sequence of similar data samples. We note that relying on the seed only would not suffice as the key sequence could become predictable to an outsider, as explained in IV-B.

IV. Detailed Design

BeDaSH consists of three modules, namely, measurement prediction, data-driven key generation, and smart contract based access and communication protocol, as explained below.

A. Measurement Prediction

BeDaSH enables transmission of the measurements of the patient's wearable sensors while obscuring them from the attacker. The idea is to deprive an eavesdropper from uncovering the data payload of an intercepted packet by employing time-varying encryption keys. Since the key needs to be known to the communication pair, BeDaSH derives the key from the data itself. Basically, the key for packet m, depends on the measurements that were transmitted in packet $(m-1)$, $(m-2)$, ..., $(m-\mu)$, where μ is a parameter that is set by the patient at the time of registration and can only be changed by the patient. The value of μ affects the complexity and performance of BeDaSH. Increasing the number of samples (LSTM inputs) will increase the number of parameters and consequently the run time; yet it becomes more difficult for the adversary to guess the LSTM output or mimic its operation. In Section VI, we study the effect of μ using an ECG dataset.

BeDaSH predicts the next sensor measurement and derives from it an encryption key for transmitting the next packet, which includes the actual data. We will describe how the keys are generated in Section IV-B. BeDaSH leverages the prediction capabilities of LSTM networks by viewing the sensor measurements as a time series. Our model consists of one input layer, one LSTM layer with four LSTM cells, one dense layer and one output layer. Fig. 3 shows the architecture of the LSTM cell with three gates: forget, memory and output. The following is performed in each cell at time epoch t [25].

$$i_t = \sigma(W_i.[H_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f.[H_{t-1}, x_t] + b_f) \tag{2}$$

$$\widehat{C}_t = \tanh\left(W_{\widehat{C}}.[H_{t-1}, x_t] + b_{\widehat{C}}\right) \tag{3}$$

$$C_t = C_{t-1}.f_t + i_t.\widehat{C}_t \tag{4}$$

$$o_t = \sigma(W_o[H_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t.o(c_t) \tag{6}$$

where:

- x_t is the input sequence (vector) for a LSTM cell and constitutes the n^{th} previous sample encrypted using K_P^{store}.
- C_t is the new state vector at t.
- H_t is the cell output, which depends on C_{t-1} and x_t.
- i_t, f_t, o_t are input, forget, and output gate sub-tensors.
- $\widehat{C}_t$ is a new cell candidate in an input sequence at t.
- b is bias for appropriate input sub-tensor.
- W_i, W_o, and $W_{\widehat{C}}$ are the weight vectors for the three gates, respectively. They are determined through training.

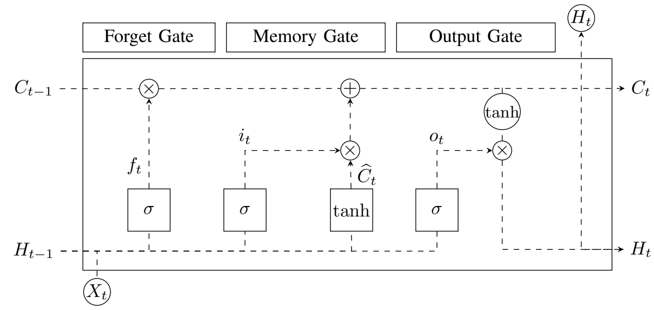


Fig. 3: Showing the detailed design of the LSTM cell.

TABLE I: Definition of the used notation.

Symbol	Description
ID_x	The identity of the patient ($x = P$), data requester ($x = R$), and cloud agent ($x = C$); this could be determined at the time of user registration.
K_x^{Priv}	Private key for the patient ($x = P$), data requester ($x = R$), and cloud agent ($x = C$)
K_x^{Pub}	Private key for the patient ($x = P$), data requester ($x = R$), and cloud agent ($x = C$)
S_m	Data sample #m
K_P^{store}	Storage key for patient's data
ES_m	Encrypted version of PS_m using K_P^{store}
PS_m	Predicted value for data sample #m using the LSTM
K_D^m	Encryption key for the packet of data sample #m
H	One way hash function, with g bits input and q bit output.
$Seed_\alpha$	A distinct seed for a random number generator for session α in the cloud (for either patient or caregiver); the random numbers is factored in when forming the packet key.

First, the forget gate determines what information is to be discarded based on H_{t-1} and the input vector, i.e., samples, and generates the output f_t. Then, the memory gate decides on what to store in the cell state. The input gate sub-tensor calculates the new value of i, while the $tanh$ layer creates a vector of new candidate values, $\widehat{C}_t$, to be included in the state. The new value of the state is obtained by multiplying the previous state by the forget gate output and adding $i_t \times \widehat{C}_t$. To calculate the output, we use a sigmoid layer to determine the parts of the cell state affecting the output. Then, we pass the cell state through $tanh$ and multiply it by the output of the sigmoid gate, so that we only output the relevant parts.

The communicating pair, i.e., patient and storage system, or storage system and healthcare provider, needs to employ the same LSTM architecture. A wearable sensor collects measurements (samples) periodically. For each period t on the patient side, BeDaSH forecasts the current sample using LSTM based on the μ previous sequential samples in the time series. All input samples are in fact encrypted using K_P^{store}, and thus the same LSTM can be applied at the cloud agent side, which in essence have access only to the encrypted version of the data using the patient-specific storage key, K_P^{store}. The output of the LSTM is used to derive the packet key, as discussed in the next subsection; such a key is used to encrypt the actual patient data sample. By running the same LSTM and key generation process, the cloud agent will generate the packet key and decrypt the payload to retrieve the actual data. We again note that all patient data sent to the cloud are encrypted using K_P^{store}. The same process is followed when an authorized medical professional requests access to the patient's measurements. In such a case the cloud agent will use LSTM to predict the next stored sample and generate keys, while the receiver side will again employ the same LSTM model to regenerate the key and retrieve the data.

B. Key Generation and Management

BeDaSH opts to employ mostly symmetric keys in order to keep the computational complexity low, especially as the data volume in tele-health applications is often large and involves many packets. The use of asymmetric primitives is confined to just user authentication. The data owner (patient), cloud agent and caregiver (data consumer) are assumed to have a private-public key pair. Meanwhile, data exchange relies on symmetric keys generated based on the data, user identity, and random numbers, as we explain below. Table I enlists all keys and defines the notation. The employed keys can be categorized as static and time-varying. The storage key, K_P^{Store}, is set up by the patient and does not get updated. Similarly the private and public key pairs for each patient and caregiver are fixed and do not change. On the other hand, the per-packet key, K_D^m, varies for each data sample and for each data receiver.

BeDaSH sustains the confidentiality of the patient's data through the use of data-driven keys. The generation process of such a data-driven key will virtually make each packet encrypted by a distinct key. As discussed in the previous subsection, the LSTM will factor in the most recent μ data samples to predict the next sample. We note that since the

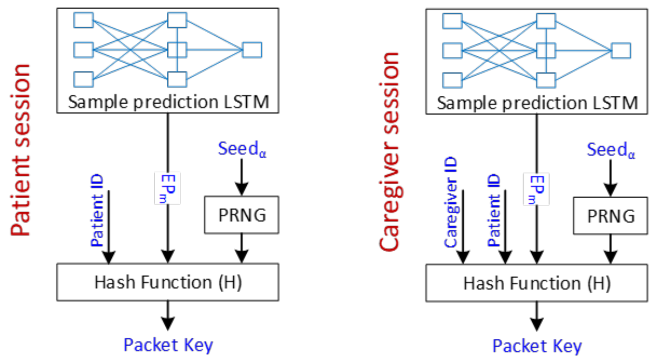


Fig. 4: Illustrating the generation process for the packet encryption key K_D^m for data sample m. In addition to the patient's ID, the caregiver ID is factored in generating the packet key during data retrieving from the cloud.

LSTM is fed with encrypted versions of S_m, semantically the LSTM is not predicting the next data sample unless homomorphic encryption machine learning (HEML) is used [26]. A homomorphic encryption function ϕ holds the property that $\phi(i \odot j) = \phi(i) \odot \phi(j)$, where $\odot$ is a mathematical operator, e.g., multiplication. Given the complexity of HEML, BeDaSH simply feeds the LSTM with $ES_{m-\mu}, ..., ES_{m-2}, ES_{m-1}$ and uses the output to generate the packet key K_D^m. However, there are two issues to be addressed. First, if the data is accessed by multiple caregivers, the keys should be different; therefore we factor in the identity of the users as well. The second issue is to ensure forward secrecy against a former user whose access credentials are revoked. To elaborate, let's consider the case of a caregiver X whose service is terminated by the patient. User X could be aware of the latest patient's data, i.e., just after termination, and can thus perform data sample prediction correctly, i.e., infer the output of the LSTM. If such a terminated caregiver knows (or steals) the identity of another legitimate user Y, it would be possible to generate the next Y's key and intercept the data traffic directed to Y. To tackle such a concern, BeDaSH employs a randomly-picked session identifier. The main requirement here is that sessions for the same user should have different identifiers in order to ensure variability of the packet keys. The session identifier should also be mutually known to the communicating parties.

BeDaSH avoids the exchange of messages to agree on the session identifier; instead it gets the cloud agent, to pick the session identifier using a pseudo-random number generator (PRNG) at the time of user authentication. BeDaSH employs the same PRNG at all user sessions; yet with different seeds. Basically the seed used in a session will be logged on the Blockchain when the session is terminated. When authenticating a user U, the cloud agent will receive the $Seed_{\alpha-1}$ of the last session α of U from the smart contract; such a seed is then used for generating $Seed_\alpha$. The important note is that the seeds should vary among consecutive sessions.

The session identifier is used by BeDaSH as a factor for determining the packet key, K_D^m as stated above. Thus, for a patient there are three essential factors, namely, the predicted data sample and the patient identity, and the random number corresponding to the session identifier. To access the data, the caregiver identity is also factor in. BeDaSH uses an $H : g \rightarrow q$ one-way hash function to generate the key, as illustrated by Fig. 4. The input to the hash function reflects the concatenated bit-patterns of the three (four) factors, and the output is the key with some desired length q. The key length will be determined based on the capabilities of the involved user devices in order to balance the high security and low overhead goals. Obviously, the same function H should be employed at the two ends of the communication, i.e., cloud and caregiver, or patient and cloud; yet different H can be used for the patient link and caregiver. We again note that the caregiver ID is not a factor in generating keys for the cloud-patient sessions.

C. Access and Communication Protocol

BeDaSH facilitates interactions between two distinct pairs of players, specifically, (i) a patient and the cloud, and (ii) the cloud and healthcare entity. Communication between a pair follows a two-phase process by authenticating the identity and confirming access authorization, and then establishing an attack-resilient communication link for streaming data. First a new user, patient or caregiver, needs to register in the system.

Initialization: In BeDaSH, each user has to create a pair of public and private keys, K_x^{Pub}, and K_x^{Priv}, and a crypto-identifier (Blockcahin address) derived from the public key, e.g., $Hash(K_x^{Pub})$. For a patient, the value of μ is also provided. Meanwhile, for a caregiver, two smart-contracts are deployed to the Blockchain network, one for access-control and the other for session information logging. The pseudo code of access-control is shown in in Algorithm 1. Due to space constraints, the code for the session-logging smart contract is not included, yet it can be found at [27]. A patient interacts with the access-control smart-contract to create a new entry for their list of authorized caregivers by sending a transaction that makes a call to the *authorizeNewUsers()* function which requires as input the crypto-identifiers of authorized caregivers. Similarly, to revoke an already-authorized caregiver, the patient sends a transaction to call the *revokeUsers()* function with crypto-identifiers of the caregivers to revoke.

In order to create a new account with the cloud provider, the patient's crypto-identity needs to be provided. The cloud will use the patient's crypto-identity to consult with the Blockchain about the list of authorized caregivers before granting access to the patient's data. The cloud agent reads the list of authorized caregiver by sending a transaction that invokes the function *getAuthorizedUsers()*, with the patient's crypto-identity as an input. It can also ask if a particular caregiver is authorized by calling *isAuthorized()* function with the crypto-identities of both the patient and the caregiver as inputs. The cloud agent interacts with the session-logging smart-contract to update and retrieve the last session logging information of a specific user.

User Authentication and Authorization Verification: All access requests are to be made to the cloud agent, which in turn consults with the Blockchain to authenticate and validate the authorization of the requester. The blockchain authenticates

Algorithm 1 Pseudo code for access-control contract.

```
contract AccessControl {
 struct MyAuthUsers {
        bool registered; address [] users; }
 mapping (address => MyAuthUsers) public authUsers;
 function authorizeNewUsers(address [] users_) {
    if (!authUsers[msg.sender].registered) {
        authUsers[msg.sender].registered = true; }
    for (i = 0; i < users.length; i++) {
        authUsers[msg.sender].users.push(users_[i]);} }
  function revokeUsers(address [] users_) {
    for (i = 0; i < users.length; i++) {
        authUsers[msg.sender].users.remove(users_[i]); } }
  function getAuthorizedUsers(address patient_) public view
        returns(address []) {
     require(authUsers[patient_].registered == true);
     return authUsers[patient_].users;}
  function isAuthorized(address patient_, address user_)
       public view returns(bool) {
     require(authUsers[patient_].registered == true);
     for (i = 0; i< authUsers[patient_].users.lenght; i++) {
        if (authUsers[patient_].users[i] == user_) {
            return true; } }
     return false;}
}
```

the cloud agent based on its public key using conventional protocols. The authentication process is summarized in Fig. 5 and slightly differs based on whether the patient or caregiver is requesting access. A patient's request is handled as follows:

(i) A patient's request includes its identity, ID_P, and public key K_P^{Pub}, as well as the last session identifier, $Seed_{\alpha-1}$. The patient will feed the last stored μ data samples, i.e., $S_{m-\mu}, ..., S_{m-2}, S_{m-1}$, to the LSTM to predict EP_m, as described in Section IV-A. The request will be encrypted using the public key of the cloud agent, K_C^{Pub}.
(ii) Upon receiving the request, the cloud agent will decrypt the packet using its private key, K_C^{Priv}, and extract ID_P.
(iii) The cloud agent will consult with the session-logging smart contract to get the last session information for patient ID_P.
(iv) The smart contract will respond to the cloud agent by sending an encrypted version of the last μ data samples provided by ID_P during the most recent session, i.e., $ES_{m-\mu}$, ..., ES_{m-1}. These values are to be provided by the cloud agent at the end of each session. Again these values reflect the μ samples encrypted by K_S^{Store}, which neither the cloud agent nor the smart contract knows. The response of the smart contract will also include $Seed_{\alpha-1}$.
(v) If the patient's identity is verified, the cloud agent will complete the authentication process by feeding $ES_{m-\mu}$, ..., ES_{m-1} to the LSTM to get EP_m.
(vi) If the agent's generated EP_m matches what is included in the patient's request, and the identifier of last session, $Seed_{\alpha-1}$, is equal to what the smart contract provided, the access is approved where the cloud agent sends a message encrypted using the public key of the patient, K_S^{Pub}. The agent will also generate $Seed_\alpha$ and include in the message.

In case the request is made by a caregiver, the packet will include the crypto-identity of the data owner (patient) since the provider may be serving multiple patients. When the access-control smart contract is consulted, it will verify

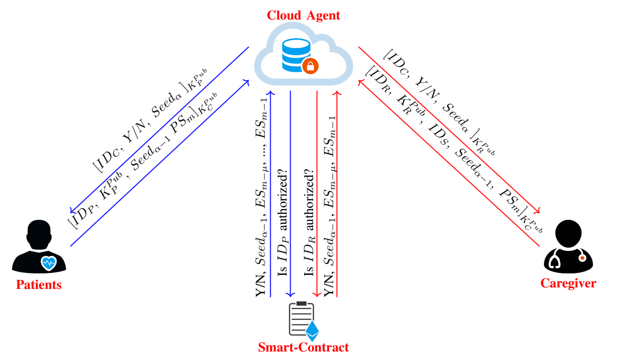


Fig. 5: Summary of the authentication process for both a data owner (patient) and healthcare provider. The smart contract validates the authenticity of the crypto-identity and the authorization of the user. If the user is validated, the encrypted values of the last data samples accessed by the user are sent to the agent to complete the authentication process.

that the caregiver ID_R is indeed authorized by the patient ID_S. Again the session-logging smart contract will provide $ES_{m-\mu}$, ..., ES_{m-1} for the last session of such a caregiver. The remaining steps are similar to their patient authentication counterpart. Finally, We note that the crypto-identity of each message sender is included in the corresponding packet in order to achieve non-repudiation. Also, each transmission will be acknowledged to ensure successful delivery.

Data Transmission Protocol: Unlike user authentication, confidentiality of the transmitted data is safeguarded through the use of symmetric cryptography. The advantage of such an approach is clearly the major reduction in computational overhead, which scales massively given the volume of sensory data in tele-health applications. BeDaSH mitigates the risk of cryptanalysis by employing time-varying keys. As stated in Section IV-B, the key is a function of the μ previously transmitted data samples, the identity of the patient, and a random number (RND). The latter is incorporated in order to mitigate the possibility of having the same measurements, which could be happening under normal health conditions. The steps for collecting data from a patient go as follows:

(i) The patient uses its LSTM to get PS_M based on $ES_{m-1}, ..., ES_{m-\mu}$.
(ii) The key K_D^m is calculated using $H(ID_S, RND, PS_m)$.
(iii) A packet payload is $[ID_S, Sequence\#, (S_m)_{K_P^{Store}}]_{K_D^m}$. The sequence number is included as a means for ordering the delivery of packets and indicating lost ones. The current time at the source can be used instead if the data sampling is periodic. The sequence number enables countering replay attacks as discussed in Section V.

The cloud agent will apply the same steps to calculate PS_M and K_D^m in order to decrypt the packet payload and retrieve ES_m, i.e., $(S_m)_{K_P^{Store}}$ and store it on the cloud. We note that the smart contract is not involved here, which avoids unwarranted delay that could degrade the freshness of the data. A similar process to the one outlined above is pursued for data transmission from the cloud to a caregiver. The only difference could be in the incorporation of the patient identifier in the key generation process. Finally, we note that all data transmissions are to be acknowledged in order to make sure that the communicating parties continue to be synchronized and calculate the encryption keys consistently.

Session Logging: In BeDaSH, the Blockchain plays two important roles, namely, authenticating users and session logging. Here, we focus on the logging aspect since it enables authorized data access. Basically, the last accessed data is considered for authenticating users and determining the session identifier, as we explained earlier in this section. Blockchain is proven as a robust ledger that does not suffer a single point of failure; BeDaSH leverages such a capability in logging relevant user activities. Specifically, for every session the involved user, either a patient or caregiver, is noted as well as the last μ data samples being sent. These samples constitute the most recent data provided by the patient or the last record retrieved by the caregiver for such a patient. The cloud agent reports these data to the smart contract when the session is terminated by the user. Again, the data stays in an encrypted form, i.e., the agent sends $ES_{m-1}, ..., ES_{m-\mu}$. In addition, the seed of the PRNG used in the session is also logged. Such a seed is further retrieved the next time the same user requests the establishment of a new session, as discussed above. We note that Bloackchain will never get access to the patient's data or know any secret related to the data exchanges; therefore, the privacy of the patient will continue to be sustained. In addition to supporting secure tele-health applications, the involvement of Blockchain enables billing for medical services and accountability of the caregivers. BeDaSH logs relevant session information such as duration, volume of retrieved data, and time of data access; such information can be used to estimate and validate charges, and settle liability claims in case of malpractice or failure of wearable sensors.

V. Security Analysis

A. Informal Analysis

Variability of encryption keys: A distinct feature of BeDaSH is encrypting data packets with time varying keys by factoring in: (1) a predicted data value based on a subset of the recently shared biomedical data items, (2) a pseudo random number that is generated using an evolving function, i.e., using new seed every session, with different initial state for each user, and (3) the unique crypto-identity of the user. The concatenation of these three values constitute a combined vector that is provided as input to a one-way hash function. The output of the hash function is of the same length of the input. We analyze the potential for having similar keys for the possible scenarios:

- *Different users*: Since each user has a distinct crypto-identity, the input to the hash function will be different for any pair of users and consequently the keys will be different. Moreover the seed of the random number generator is independently picked for each user and for each session. Thus, the probability for two users having the same random number for their sessions is in fact the probability of setting the parameters for their sessions similarly, which is quite small given the independence in seed selection for the corresponding PRNG. Moreover, users may not be synchronized and their most recently accessed data items could be different. Although the predicted data and random number variability diminish the probability of having two similar keys used by two users at the same time, the crypto-identity suffices as a discriminator among keys employed for the packets of different users.
- *Consecutive Packets for a user*: In this case, the crypto-identity stays the same, yet the data sample and the random number would vary. The predicted data value is expected to change unless the patient conditions are very stable and the precision of the measurements is low. To illustrate, if the resolution of the measurements is 3 decimal digits, the probability for having two similar predictions in a row is lower than when the resolution is only a single decimal digit. Nonetheless, even if the data sample does not change over time, the incorporated random number will.

Resilience against eavesdroppers: For an eavesdropper to retrieve the patient's data during transmission, the intercepted packet needs to be successfully decrypted. Since BeDaSH employs a time varying encryption key, cryptanalysis will fail due to the unavailability of sufficient encrypted data for each key. The only option for the eavesdropper is to regenerate the keys. We distinguish between outsider and insider attackers.

- An outsider is someone who never used the system but knows the security provisions used. To regenerate the key, such an attacker needs to guess/acquire all three inputs for the key generation hash function, namely, the crypto-identity of a legitimate user, the last data such a user received (or sent in case of a patient), and the setting for the seed for the PRNG. Such information cannot be obtained except through collusion with a legitimate user. According to our system model, caregivers who are given access to the data are assumed to be trustworthy. In fact, a colluding user can simply give away the data without burdening the attacker by eavesdropping and cracking the security of the system.
- Internal attacks reflect the case that some who used the system continue to retrieve the data despite losing access privilege. Consider for example, a caregiver who no longer is associated with the patient. In such a scenario, the attacker will know the system well and could also have the latest data, e.g., the attacker starts right after losing access permission, which implies that the attacker can predict the data value used as input to the hash function. However, such an attacker will still fail since the crypto-identity used in the intercepted packets will be different (and unknown to the attacker). In the extreme worst case, the attacker steals the crypto-identity of a legitimate user. However, the PRNG seed varies across users and the attacker will fail to overcome such a hurdle unless the legitimate user colludes.

Resilience against replay and impersonation attacks: Being unable to decrypt intercepted packets and manipulate the data, an attacker may replay a message as is, in order to get the communicating parties out of sync. To illustrate, replaying the packet for a data sample S_m can lead to one of the following two scenarios: (1) the predicted sample PS_{m+1} happens to match S_m, allowing the receiver to successfully decrypt the packet payload and wrongfully assume the included data to reflect S_{m+1}, or (2) PS_{m+1} differs from S_m causing the receiver to reject the packet. Obviously the second scenario is harmless; yet the first scenario needs further consideration. As discussed in Section IV-C, a sequence number (or time stamp) is included in each data packet. Consequently, the receiver will detect the duplication and discard the replayed packet. In summary, DeBaSH is resilient to replay attacks.

On the other hand, an adversary could try to impersonate one of the communicating parties. Such impersonation may take the form of requesting access using a stolen crypto-identity of a legitimate user or sending malicious data packets while masquerading as the patient or cloud agent. Our data-driven authentication process will thwart the first scenario since the attacker does not have the LSTM model used to predict the next data sample. Referring back to Fig. 5, the adversary will not know PS_m. For the same reason, masquerading a data source or the cloud agent will fail. The only possible serious scenario is when an internal attacker, as defined earlier, takes advantage of the data samples that were received before access termination and a stolen crypto-identifier of a legitimate user in order to make a request for a new session. To illustrate let's assume that a user X is a freshly terminated caregiver, and user X has stolen ID_Y of an active caregiver Y. It is also not difficult to know K_Y^{Pub} since it is not generally secret. In such a case, X could send a request to access the data of patient S. Although such a scenario is not impossible, it is impractical since both X and Y should have been synchronized in terms of what data of S they have received overtime. Nonetheless, such an attack scenario can be easily countered by BeDaSH. Recall that the access request from a caregiver Y takes the form $[ID_Y, K_Y^{Pub}, ID_S, Seed_{\alpha-1}, PS_m]_{K_C^{Pub}}$. Since X does not know $Seed_{\alpha-1}$ for the last session $\alpha - 1$ of user Y, the cloud agent will detect the impersonation attack.

B. Formal Verification

We have verified the security properties of BeDaSH using AVISPA [4], which is a widely used formal security verification framework to assess vulnerability to active and passive attacks such as impersonation, message relay. etc. We have described BaDeSH's authentication and data exchange protocols using HLPSL, which is a High Level Protocol Specification Language used by AVISPA. First, we have defined the communicating parties in a session, i.e., patient, cloud, Blockchain and caregiver. We have then specified for each party all possible states and transitions, as well as its initial state, keys and data records and machine learning function. Transitions between states can involve transmission and reception of messages and other security properties. The environment role includes the possible sessions and the specification of initial role session parameters. A role session may involve an intruder. We assume that the intruder knows the communicating parties and their public keys. The HLPSL based description of BeDaSH can be found in [27].

We have validated BeDaSH using a multi-step analysis to cover the different operations. We have created two parallel sessions for caregivers and patients. The security goals for the AVISPA simulation are the authentication of patients and caregivers and the secrecy of the data and the keys. This allows the detection of any outsider/insider attack. The proposed protocol is simulated using the SPAN (Security ANimator for AVISPA) simulation tool and tested using the OFMC backend checker. The output of is shown in Fig. 6, where:

– SUMMARY: reports whether the protocol is SAFE or UNSAFE, or INCONCLUSIVE.
– DETAILS: indicates the settings of the tested protocol.
– PROTOCOL: names the name of the protocol.
– GOAL: indicates default security goals or user defined.
– STATISTICS: reports the execution time, search-time, visited nodes (state) and the depth of the state transition graph analyzed by the OFMC backend checker.

The OFMC results confirm the safety of the different phases of BeDaSH, implying robustness against eavesdropping, man-in-the-middle, replay and impersonation attacks.

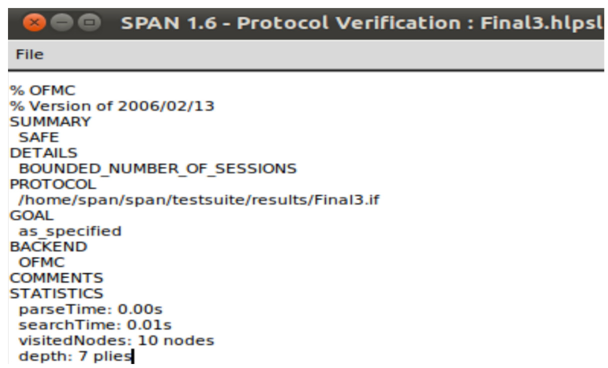


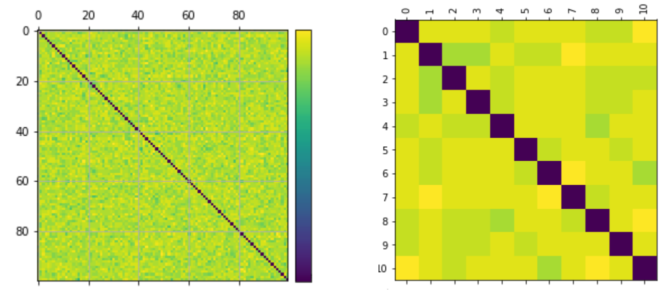
Fig. 6: A screenshot of the OFMC output, confirming BeDaSH robustness.

VI. Performance Assessment

The performance of the tele-health system, e.g., data access latency, depends on the underlying cloud system, the specific Blockchain, and communication infrastructure, which is extensively covered in the literature. Hence, in this section we focus on studying the effect of μ on the precision of the predicted data. We also confirm the dissimilarity among the generated keys. Then, we assess the computational overhead of BeDaSH and show that it is significantly less than conventional methods and will positively impact the latency.

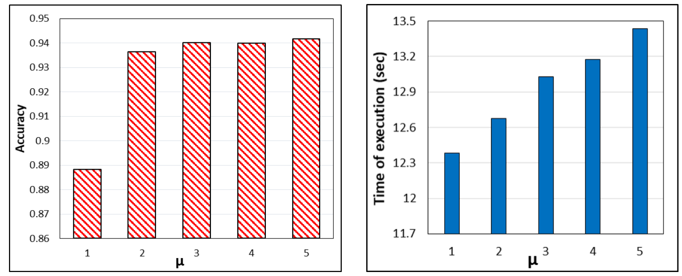
A. Similarity of Data-Driven Keys

Encrypting each packet with a distinct key would achieve ultimate resilience to cryptanalysis. To validate the effectiveness of our key generation approach, we use a popular dataset from PhysioNet [28]. The dataset contains 24 hours ECG measurements, collected during patient monitoring. In the validation, we have used the ECG data for ten patients. We compare the performance of BeDaSH in terms of key uniqueness over time for the same patient and among multiple patients. The metric used to assess the key similarity is the



(a) (b)

Fig. 7: The similarity of data-driven keys for: (a) the same patient, and (b) multiple patients.



(a) (b)

Fig. 8: The effect of μ on the data (a) sample prediction accuracy and (b) LSTM execution time including key generation.

Levenshtein distance between the keys of the same user, and the least Levenshtein distance among the keys of user pairs. The Levenshtein distance is a string metric for measuring the difference between two sequences based on the minimum number of characters to insert, delete or substitute in order to match two given strings. Fig. 7(a) shows the similarity matrix for keys used for the same patient. The matrix is simply diagonal implying the uniqueness of the generated keys, e.g., each key is only similar to itself. The figure shows the results for 100 keys. Fig. 7(b), on the other hand, reports the similarity of keys of every pair of patients. The matrix is again diagonal implying that none of the 10 patients has similar keys.

B. Complexity of LSTM Model

Recall that BeDaSH factors in the most recently accessed μ data samples in generating the per-packet key and in authenticating users when establishing a new session. The larger μ gets, the more difficult for the adversary to guess the encryption key becomes. Yet, growing μ boosts the complexity of the LSTM model. We have studied such a trade-off using the ECG dataset; here we use unencrypted samples as we are interested in the convergence of the LSTM for various settings of μ. A dataset of 50,000 ECG records is divided into two subsets of 80% for training and 20% for testing. Assuming the current time is t, we want to predict the data value at epoch $t+1$ given the measurements for current and $\mu-1$ previous time epochs. The LSTM is trained for 1000 epochs with ECG measurements of a single patient. The experiment tracks the required execution time using the $time.time()$ function in Python while executing the LSTM model on a windows desktop computer with 4 cores of Intel Core i7 and 16G RAM.

Fig. 8 reports the observed variation of the prediction accuracy and run-time for each setting of μ. The run-time does not include training since it is done once. The results indicate that μ does not affect much the test time for LSTM. Fig. 8 also shows that considering more than three samples is unwarranted and will grow the session log unnecessarily. Basically, a key that is created based on three samples will be similar to that generated while considering fours samples. Hence, having $\mu > 3$ will not contribute any confusion for the adversary if the number of samples are incrementally tried (assuming that the adversary could know some samples).

C. BeDaSH Computational Overhead

BeDaSH uses patient-picked and data-driven symmetric keys for storage and communication, respectively. Such an approach makes BeDaSH lightweight compared to alternatives that use asymmetric cryptography, e.g., [17] and [18]. Fig. 9 shows the execution time for BeDaSH in comparison to when either RSA or ECC is applied for encrypting the data using 1024-bit keys. For BeDaSH, each generated symmetric key is used to encrypt a packet of 70 ECG records using the DES (Data Encryption Standard) algorithm. To capture the computation overhead over time, we have progressively increased the size of the overall data transmitted and captured the execution time. The results clearly demonstrate the advantage of BeDaSH where the performance gap grows wider as more data is handled. Fig. 9 testifies for the scalability of BeDaSH.

We have also estimated BeDaSH overhead using an Arduino microcontroller that has an active current of 1.23mA when clocked at 16 MHz; the average power consumed during processing is approximately 5mW. We have used the Valgrind profiler with Verrou tools to estimate the set and the number of instructions for the applied algorithms while handling the prediction. Such instruction count is further multiplied by the number of cycles per instruction to estimate the runtime. We note that the model is trained offline. The estimated execution for predicting a data sample by our LSTM is approximately 0.4496 ms, and corresponding energy is 22.5μJ. The estimated time for key generation and encryption is 13.18s for 10000 bytes data size. The comparative results are found to be consistent with those of the Inter Core i7 PC.

D. Smart-contract Performance

We have validated the *AccessControl* and *SessionLogging* contracts on a public and consortium blockchain network. Both smart-contracts are compiled in the Remix IDE and deployed to the Ethereum official public and private test networks Rinkeby and Kovan, respectively. Rinkeby uses the Proof-of-Work (PoW) consensus algorithm, whereas Kovan uses a Proof-of-Authority (PoA) algorithm. The average execution time for these contracts is found to be 16 and 4 sec for Ropsten and Kovan, respectively. Such time is not deemed a bottleneck since in BeDaSH transactions are sent to the blockchain only during the initialization, authentication and session logging and are not needed during data transmission. Moreover, the

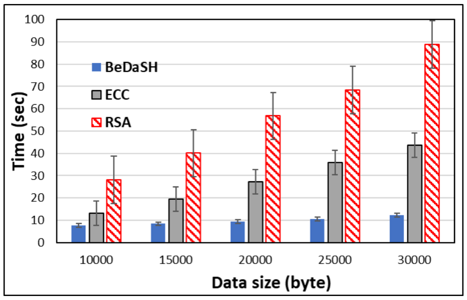


Fig. 9: Comparing the time complexity of BeDaSH to a baseline approach that uses RSA or ECC algorithms for data storage and communication.

measurements are made on a test network; using a customized blockchain network would provide better performance.

A public blockchain applies PoW and thus incurs more transaction latency than a consortium blockchain. Moreover, transaction fees in public networks depends on the underlying cryptocurrency, which is not stable in value. For instance, the fee for deploying *AccessControl* to the Ethereum network is about \$166; such fee would have been less than \$1, few years ago. Hence, we recommend building a consortium blockchain with the help of medical institutions to run BeDaSH.

VII. Conclusions

Tele-health systems have been gaining popularity in recent years that even peaked with the COVID-19 pandemic. While these systems offer numerous advantages for patients, caregivers, and insurance providers, they introduce major challenges in secure handling of the voluminous data while preserving privacy. To tackle these challenges, this paper has presented BeDaSH, a novel solution that enables secure storage and dissemination of patient data and puts patients in charge of defining access rules. BeDaSh exploits the storage capability of the cloud while ensuring complete privacy preservation by keeping the data encrypted using a patient-controlled key. To counter potential data leaks through eavesdropping on the wireless communication links, BeDaSH promotes a new data-driven key generation mechanism that enables every packet to be encrypted using a distinct key. Furthermore, BeDaSH exploits the smart contract features of Blockchain to offer user authentication and access control. The overall system mitigates the most prominent attacks on tele-health systems, including impersonation, message replay, and data forgery. The effectiveness of BeDaSH has been confirmed through performance validation and formal security analysis.

Acknowledgement: Younis, Lalouani, and Emokpae are supported by the National Science Found., Contract #2030629.

References

[1] C. Koziatek, et al., "Use of a telehealth follow-up system to facilitate treatment and discharge of emergency department patients with severe cellulitis," *The American Journal of Emergency Medicine*, 2020.

[2] N. Liu, et al., "Telehealth for noncritical patients with chronic diseases during the covid-19 pandemic," *J Med Internet Res*, vol. 22, no. 8, p. e19493, Aug 2020.

[3] C. Thapa and S. Camtepe, "Precision health data: Requirements, challenges and existing techniques for data security and privacy," *Computers in Biology and Medicine*, vol. 129, p. 104130, 2021.

[4] L. Viganò, "Automated security protocol analysis with the avispa tool," *Electronic Notes in Theoretical Comp. Sc.*, vol. 155, pp. 61 – 86, 2006.

[5] A. Celesti, et al., "Information management in iot cloud-based telerehabilitation as a service for smart cities: Comparison of nosql approaches," *Measurement*, vol. 151, p. 107218, 2020.

[6] S. Pirbhulal, P. Shang, W. Wu, A. K. Sangaiah, O. W. Samuel, and G. Li, "Fuzzy vault-based biometric security method for tele-health monitoring systems," *Computers & Electrical Eng.*, vol. 71, pp. 546 – 557, 2018.

[7] J. J. Hathaliya and S. Tanwar, "An exhaustive survey on security and privacy issues in healthcare 4.0," *Computer Communications*, vol. 153, pp. 311 – 335, 2020.

[8] S. Shi, D. He, L. Li, N. Kumar, M. K. Khan, and K.-K. R. Choo, "Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey," *Computers Security*, vol. 97, p. 101966, 2020.

[9] S. Challa, et al., "An efficient ecc-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks," *Computers & Elec. Eng.*, vol. 69, pp. 534 – 554, 2018.

[10] Y. Bao, W. Qiu, and X. Cheng, "Secure and lightweight fine-grained searchable data sharing for iot-oriented and cloud-assisted smart healthcare system," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[11] H. Huang, P. Zhu, F. Xiao, X. Sun, and Q. Huang, "A blockchain-based scheme for privacy-preserving and secure sharing of medical data," *Computers Security*, vol. 99, p. 102010, 2020.

[12] B. S. Egala, A. K. Pradhan, V. R. Badarla, and S. P. Mohanty, "Fortified-chain: A blockchain based framework for security and privacy assured internet of medical things with effective access control," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[13] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, "Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42 – 52, 2018.

[14] R. Kumar, et al., "An integration of blockchain and ai for secure data sharing and detection of ct images for the hospitals," *Computerized Medical Imaging and Graphics*, vol. 87, p. 101812, 2021.

[15] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Bedgehealth: A decentralized architecture for edge-based iomt networks using blockchain," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[16] Q. Zhao, S. Chen, Z. Liu, T. Baker, and Y. Zhang, "Blockchain-based privacy-preserving remote data integrity checking scheme for iot information systems," *Information Processing & Management*, vol. 57, no. 6, p. 102355, 2020.

[17] S. Shamshad, Minahil, K. Mahmood, S. Kumari, and C.-M. Chen, "A secure blockchain-based e-health records storage and sharing scheme," *Journal of Info. Security and Applications*, vol. 55, p. 102590, 2020.

[18] A. Manzoor, A. Braeken, S. S. Kanhere, M. Ylianttila, and M. Liyanage, "Proxy re-encryption enabled secure and anonymous iot data sharing platform based on blockchain," *J. of Net. Comp. App.*, p. 102917, 2020.

[19] A. Anand and A. K. Singh, "An improved dwt-svd domain watermarking for medical information security," *Computer Communications*, vol. 152, pp. 72 – 80, 2020.

[20] S. Pirbhulal, O. W. Samuel, W. Wu, A. K. Sangaiah, and G. Li, "A joint resource-aware and medical data security framework for wearable healthcare systems," *Future Gen. Comp. Sys*, vol. 95, pp. 382–391, 2019.

[21] G. Zheng, et al., "Multiple ecg fiducial points-based random binary sequence generation for securing wireless body area networks," *IEEE J. of Biomedical & Health Informatics*, vol. 21, no. 3, pp. 655–663, 2017.

[22] M. Usman, M. R. Asghar, I. S. Ansari, and M. Qaraqe, "Security in wireless body area networks: From in-body to off-body communications," *IEEE Access*, vol. 6, pp. 58 064–58 074, 2018.

[23] M. J. M. Chowdhury, et al., "A comparative analysis of distributed ledger technology platforms," *IEEE Access*, vol. 7, pp. 167 930–167 943, 2019.

[24] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges." *IJ Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[25] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "Recurrent neural networks for short-term load forecasting," in *SpringerBriefs in Computer Science*, 2017.

[26] A. Vizitiu, C. I. Niă, A. Puiu, C. Suciu, and L. M. Itu, "Applying deep neural networks over homomorphic encrypted medical data," *Comp. and Math. Methods in Medicine*, vol. 2020, no. 3910250, p. 26, 2020.

[27] [Online]. Available: https://github.com/noureddinel/BC-SmartHealth.git

[28] V. Novak, et al., "Cerebral flow velocities during daily activities depend on blood pressure in patients with chronic ischemic infarctions." *Stroke*, vol. 41, no. 1, pp. 61–66, 2010.