Robust and Efficient Data Security Solution for Pervasive Data Sharing in IoT

Wassila Lalouani, Mohamed Younis, Mohammad Ebrahimabadi and Naghmeh Karimi
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County,
Baltimore, Maryland, USA,
{lwassil1, younis, e127, nkarimi}@umbc.edu

Abstract— **Pervasive sensing is shaping up modern societies and opening the door for many unconventional applications. Instead of the contemporary access model where sensor data is disseminated to a single user, multi-access scenarios are becoming more prevalent, which raises the issue of how to authenticate users, how to ensure access authorization, and how to prevent information leakage. To address these issues, this paper presents a novel lightweight protocol that promotes a data-driven methodology. The idea is to employ hardware primitives to support authentication of legit data recipients and to factor in the previously shared data samples in generating encryption keys. Our protocol in essence generates encryption keys that vary per packet and in an implicitly synchronized manner between the data source and each recipient. The generated key is also a function of the hardware primitive and thus effectively prevents data access to unauthorized recipients. We analyze the resilience of our protocol to impersonation and message replay, and hardware primitive modeling attacks. The security properties of our solution is validated using the AVISPA toolset and its performance is compared to the asymmetric cryptography approaches.**

Keywords: Pervasive sensing, Secure data sharing, IoT, Data-driven encryption, Hardware-based authentication.

I. INTRODUCTION

The current era is characterized by the popularity of devices that combine sensing, computation and communication capabilities. An Internet-of-Things (IoT) reflects incorporation of many of these devices to provide unconventional services and enable the realization of smart cities [1]. The key feature that an IoT enables is the support for data collection and sharing at a large scale. The pervasive availability of sensor devices arguably is transformative where the collected data can be used for facilitating operation autonomy in many application domains. Examples include intelligent transportation systems, telehealth solutions, smart manufacturing, home automation, digital battlefield, etc. Most of these systems are usually associated with large volumes of real time data that are disseminated to multiple parties. In fact, the notion of Sensing-as-a-Service has emerged to reflect such a data dissemination model, where a data provider serves multiple subscribers. To illustrate, in a telehealth system, multiple caregivers could have access to data collected by a patient's wearable sensor system. Similarly, a road sensor could provide traffic data to light signals, certain vehicles, traffic authorities, etc.

However, with such a data sharing model comes the concern about security, particularly sustaining data integrity, preserving privacy and ensuring access authorization [2][3]. The integrity of the collected data is paramount in applications involving control and decision support systems. For example, remote monitoring of patients requires accurate and authentic data to assess the health conditions and enable timely intervention when needed. Similar scenarios can be noted about defense systems where weapons are to be engaged and in the transportation domain where the road safety could be risked if decisions are made on tampered data. Privacy and authorized access are also required in many applications, e.g., sharing of patient data. An external attacker may eavesdrop on packet transmissions and capture the data, or impersonate the communicating parties. Therefore, provisions are needed to safeguard the dissemination of the collected sensor data and ensure that only authorized access is allowed.

Employing a cryptosystem is the conventional approach for achieving data integrity and privacy. Access authorization also needs authentication, which is often supported by crypto-identifiers. Since asymmetric cryptosystems impose high computation and communication overhead, the use of lightweight symmetric schemes and authentication primitives is desired in the context of IoT applications [4]. However, the distributed operation mode, the ad-hoc networking, and the pervasive nature of the devices, make the management of symmetric pair-wise keys and device authentication to be quite challenging. Specifically, the presence of a centralized trusted authority for facilitating the interaction among device pairs would constitute a performance overhead and could be even infeasible. In addition, symmetric encryption may not suffice to provide protection for longer duration given the computational power that is available nowadays and which could allow the adversary's cryptanalysis to succeed. Hence, frequent key updates will be necessary which will be inefficient for pervasive data streaming. The challenge even grows when considering the network dynamic and multi-access model, where data can be shared with a restricted subset of devices that can vary over time. Therefore, support of access control would be needed to serve new authorized recipients while depriving unauthorized devices and those whose authorization has expired, from retrieving the data.

To overcome the aforementioned challenges, this paper promotes an innovative solution that supports effective and robust data access control, and enables device-specific data integrity and confidentiality. To ensure authorized data access, we propose a lightweight authentication mechanism using hardware-primitives. Specifically, we employ Physical

978-1-6654-3161-3/22/$31.00 ©2022 IEEE

unclonable functions (PUFs), which constitute hardware fingerprints since they depend on the variations in the fabrication of integrated circuits and are not reproducible even by the same manufacturers [5]. A PUF circuit acts as a unique lookup table that is indexed by a fixed size bit-stream, referred to as a challenge, for which a response is provided. A key advantage of PUFs is that the response is generated instantaneously by the circuit and cannot thus be read from memory [6]. In addition, we propose a novel scheme for supporting data integrity and confidentiality. The idea is to generate data-driven encryption keys that factor in the features of historical streamed data exchange between a communicating device pair. We employ a machine learning (ML) model at both the data sender and receiver that considers the transmitted data values as a series in order to predict the next data value and use it in defining an encryption key. Thus, the keys will dynamically change and will be implicitly inferred rather than explicitly exchanged through a key update protocol. Also the encryption keys are not stored, making our scheme to be more secure than conventional symmetric ciphers, e.g., AES.

To validate data access authorization, the legit receiver will be authenticated in the data packet by factoring in a challenge-response pair (CRP) of the embedded PUF at such a receiver. This is achieved by making the generated encryption keys a function of the CRPs as well as the predicted data value. In the absence of consensus between the communicating party, the packet payload cannot be decrypted to retrieve the data. Thus, an eavesdropper will fail to gain access to data or even extract any significant information from a single packet given the variability of the key over time, which is caused by variation in the sensor measurements and the employed CRPs for a receiver and across different receivers. We validate the performance by comparing the overhead to popular cryptographic approaches, specifically PKI and ECC. We also provide security analysis to show that our proposed Data Access Control and Integrity and Confidentiality (DACIC) solution is able to counter all imminent threats including impersonation, Sybil, man-in-the-middle, and message replay. We further show that DACIC mitigates the threat of PUF modeling attacks since the CRPs of a receiver are not explicitly exchanged and their use for authentication is by verifying their correctness through their effect on the generated keys.

To the best of our knowledge, DACIC is the first approach that combines hardware-based authentication primitives, and data-driven methodologies to achieve an effective, lightweight, robust and scalable security solution for pervasive data sharing in the context of IoT applications. The rest of the paper is organized as follows. The next section sets DACIC apart from existing solutions. Section III discusses the considered system model and covers some preliminaries. The design of DACIC is presented in Section IV and its security properties are analyzed in Section V. The performance of DACIC is evaluated in Section VI. Finally, the paper is concluded in Section VII.

II. Related Work

Given the contribution of DACIC, we focus on prior work on data-and PUF based security solutions in the realm of IoT.

Data driven authentication/encryption: The use of data in devising a security solution is popular in the realm of wearable sensors where biometric signatures are authenticating data sources, e.g., patients [7], identify verification, e.g., to granting access to a bank account [8], and including watermarks in records [9]. Biometric signals have also been leveraged to generate encryption keys [10][11]. Unlike these techniques, DACIC authenticates clients and generates encryption keys based on the most-recent data samples, which introduces more variability and makes the crypto-system more robust against attacks. In addition, DACIC employs an LSTM (Long Short-Term Memory) model in predicting the data based on which the encryption keys are derived; since the LSTM factors in both the data sequence and values, it is impossible for an eavesdropper or unauthorized user to guess the key without mimicking the entire process and using the same LSTM parameters.

PUF-based security solutions: Security of IoT is an active research topic, where most attention has been dedicated to authentication and generation/management of cryptographic keys [3][4]. Many of the existing IoT authentication schemes store the device identifier and encryption keys in its memory, which could be revealed through hacking. To avoid such vulnerability, quite a few studies have explored the use of PUFs. Although these authentication schemes are lightweight, and benefit from the unique footprints of PUF devices, they suffer from vulnerabilities to security threats such as modeling attacks, replay attacks, and impersonation attacks [12][13].

Chatterjee et al. [14] use PUFs to generate public and private keys to be used for securing data transfer in IoT. The proposed scheme is resilient against replay attacks, yet it is computationally intensive and would not suit resource-constrained IoT devices. J. R. Wallrabenstein [15] opts to avoid storing the private key in the device memory in order to achieve tamper resistance. The approach is to embed an Elliptic Curve Cryptosystem (ECC) on the IoT device, to be used along with the PUF to regenerate the private key when needed. However, the approach imposes increased hardware overhead on the IoT device. Meanwhile, Li el al. [16] employ ECC along with the PUF to achieve mutual authentication. DACIC avoids such complexity by enabling the use of only lightweight symmetric cryptosystems. In addition, it provides mutual authentication for the communicating parties as well as per packet authentication. Along with the PUF, DACIC factors in the exchanged data in achieving authorization, confidentiality and authentication. Particularly, DACIC ensures the legitimacy of the data receivers on a per-packet basis and is thus deemed to be a continuous authentication method [17]. To our knowledge such a security solution has not been considered in the literature.

III. System Model and Preliminaries

A. System Model

A typical operation in an IoT pervasive data sharing system involves clients that subscribe to receive data provided by distinct sources. A source could be offering the data as a service, e.g., road-side units sharing traffic data with on-road vehicles, or sharing the data to get a service, e.g., sharing the wearable system measurements of a patient with remote caregivers. In

both scenarios the operation of the system requires a robust mechanism to ensure data integrity, confidentiality and access control, hence there is a need for authenticating communicating parties. DACIC strives to achieve the aforementioned security goals by incorporating of hardware fingerprints and employing a data driven approach to secure packet transmissions. A data-driven methodology is pursued to achieve continual packet protection over time and counter data provider impersonation attempts. In essence, generating encryption keys based on the previously shared data will couple the communicating pair, i.e., data provider and client, and enable authenticating the sender of the packet.

Each client is to be equipped with a PUF that serves as a fingerprint. The data providers will leverage the client's fingerprint in the data sharing process in order to ensure the authenticity of authorized receivers. A PUF can be viewed as a lookup table that maps a bit-stream, referred to as challenge, to corresponding response bits. For the client such a table is not stored, but instead generated through a simple circuit, as we explain in the next subsection. During enrollment, e.g. data service subscription, the client transmits a subset of its PUF challenge-response pairs to the data provider. The enrollment process is assumed to be secure, e.g., facilitated by a trusted third party, and is beyond the scope of this paper. In DACIC, the data provider S_x uses the CRPs that it knows about a client U_y to ensure that a data packet is only readable by U_y, i.e., the CRPs are used to implicitly authenticate the identity of authorized clients. DACIC assumes that a data provider cannot be manipulated. In addition, a PUF is not needed for authenticating the provider in DACIC, as we explain later. Finally, we assume a reliable link layer protocol is employed for communication in the system such that a packet delivery is acknowledged to confirm reception.

B. Physical unclonable functions

Fundamentally, a PUF design leverages process variation in the manufacturing of semiconductor devices. Such variation affects the timing that signals travel at a scale that is arbitrary in nature. We note that the variation can be observed among devices from the same manufacturer. By building a circuit like the one shown in Fig. 1, one can define a hardware fingerprint for the individual devices [5]. Fig. 1 shows the circuit of an Arbiter PUF, which is one of the prominent PUF designs. Each of the signals C_0, …, C_{n-1} impact the setting of the multiplexer and consequently the path the input signal takes to reach the arbiter (realized as a latch in Fig. 1). Given that the signal propagation delays are different due to manufacturing variation and the latched value will vary. The bits C_0, …, C_{n-1} are considered as PUF challenge and the output Q constitutes the PUF response. A multi-bit response can be realized by replicating the circuit or querying it with multiple challenges. As for each same set of challenges, two distinct PUF circuits generally provide different response, the PUF can be used to identify clients and ensure their authenticity.

C. Attack Model

DACIC opts to safeguard the system against two attacker types. The first is an external attacker that eavesdrops on the communication links between data providers and clients. The objective of such an attacker is to intercept transmissions and:

(i) applies cryptanalysis to access the data. This could be for data piracy reasons, gaining knowledge of privacy information, or manipulating the shared data, i.e., for launching man-in-the-middle attack.

(ii) impersonates any of the communicating parties to fool other nodes in the systems. Impersonating a client could be for example to charge it for a service that the attacker receives. For data providers, impersonation could be to send bogus data and defame the source for competition reasons.

(iii) replays messages in order to either confuse clients by providing outdated data or to defame the source.

The second type is an internal attacker that knows the security protocol. The attacker could be a client that: (1) has ceased to subscribe and wants to illegally access the data, or (2) opts to violate the privacy of other clients by knowing what data they receive. An internal attacker could also try to impersonate a data provider. A prominent example for such an attack scenario is when a provider S_i that lost a client U_y to a competitor S_2, tries to impersonate and defame S_2 by sending bogus data to U_y.

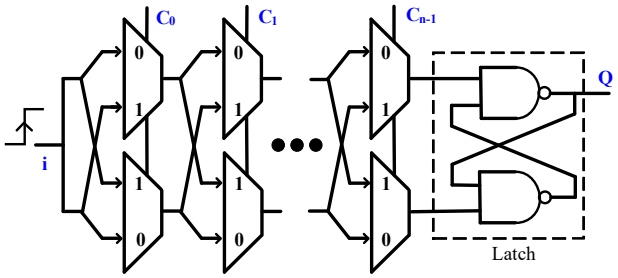

Fig. 1. Schematic diagram description of an Arbiter PUF, where the challenge bits control the individual multiplexers and cause the input signal to experience different delays on distinct devices and consequently the latched value (Q) would differ.

IV. DACIC Design

A. Appraoch Overview

In smart cities, a client (user) could subscribe to a data delivery service. Such data may be the transmission of sensor measurements, traffic conditions, stock market updates, etc. In these application scenarios, the received data may be used in making critical decisions and hence sustaining data integrity during the transmission becomes very important. In addition, the subscription itself could be fee-based or subject to privacy agreement, and consequently the provided data should be accessible only to the authorized clients. DACIC opts to achieve the data integrity goal and obscure the transmission from eavesdroppers, including former clients. While employing an asymmetric cryptosystem will be a viable option, it is undesirable in the realm of IoT due to resource constraints. To overcome the vulnerability of symmetric encryption to cryptanalysis and the complication of the associated key management process, DACIC employs a data-driven key generation process that enables keys to vary overtime.

A block diagram description of DACIC is shown in Fig. 2. DACIC ensures the integrity and the confidentiality of the data

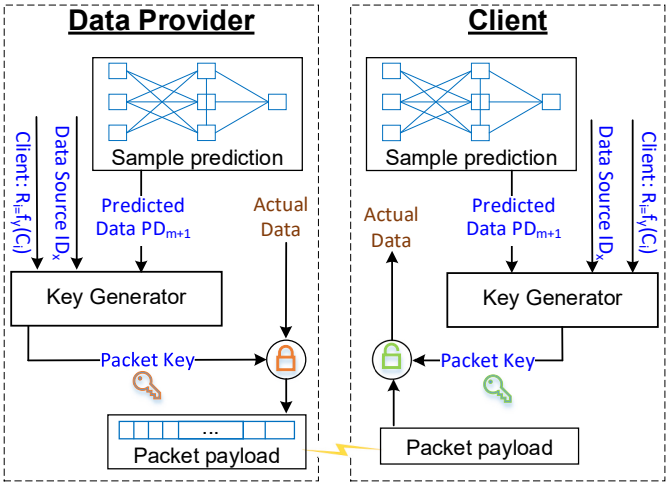

Fig. 2. A block diagram description of DACIC capturing the operation at the data provider and client side by side.

traffic using per packet encryption keys. The idea is to factor in the historical data that the source provided to a client in generating an encryption key. DACIC employs a deep learning model that views the packet payloads as a data series, D_1, D_2, …, D_n, and predicts the next data value, PD_{n+1}. By applying the same model at both the data provider and the client, the encryption key can be inferred without an explicit key management procedure. Any attempts to impersonate the source will fail given dependence on the historical data. Moreover, the variability of the encryption among packets makes cryptanalysis ineffective. We note that the use of deep learning models, rather than producing PD_{n+1} using a function, enables great protection since the model cannot be reversed engineered by capturing the most recent n packets.

As noted earlier, each client is to be equipped with a PUF that serves as a fingerprint. When a client U_y enrolls for a data service from a particular source S_x, a set of CRPs of the client's PUF is provided to S_x. The latter will utilize these CRPs to implicitly authenticate the receiver of a transmitted packet, i.e., the client. In addition to the predicted data, PD_{n+1}, DACAC factors in the response of U_y's PUF in a challenge bit stream in the generation of the encryption keys. Hence, the packet will be decrypted only by U_y, which is the authorized receiver in such a case. By considering the client's PUF response and due to the fact that a new key is generated for each packet, DACIC practically enables authentication of the client on a per-packet basis. This is a very powerful protection and makes packet interception useless for an external attacker. We note that an internal attacker that knows CRPs for U_y, e.g., a former service provider, does not know the data and hence cannot infer PD_{n+1}, and decrypt the message. By avoiding explicit dynamic key management, reliance of simple hardware fingerprinting primitive, and using a symmetric cryptosystem, DACIC limits the computation and communication overhead for secure data sharing in a multi-source and multi-client IoT system.

B. *Data Sample Prediction*

The underlying design process for the data-driven encryption is that streamed data in the aforementioned applications exhibits temporal properties. For example, data from a wearable sensor system reports changes in body conditions and generally reflects trends. Similarly, variations in traffic intensity data and weather are temporally dependent. Deep learning models, particularly, recurrent neural networks (RNN), are the most appropriate for capturing the variability of the transmitted data over time. DACIC leverages such temporal data dependence to adaptively vary the employed encryption key in a synchronized manner between the data provider and individual clients. The idea is to deploy a deep network at the source that factors in the previous n data items D_1, D_2, …, D_n, e.g., the five most recent traffic densities, to predict the new data, PD_{n+1}. The same deep network will be employed at the client and hence will yield the same predicted data. We note that the predicted data PD_{n+1} is used only for key generation; the actual data D_{n+1} will be still included in the packet, encrypted by the key generated through data prediction.

In our recent work [18], we have demonstrated the viability of data sample prediction in the context of wearable sensory system, where the temporal dependency pattern is modeled using RNN. DACIC leverages such a technique in the general context of IoT applications, where the data payload of the n last transmitted packets is fed to a Long Short Term Memory (LSTM) model to forecast the next data item. Due to their mitigation of the gradient vanishing and their ability to extract patterns, LSTM models have been widely used in traffic and measurement prediction. LSTM makes such a prediction using specific architectures containing three gate types, namely, Forget, Memory and Output gates. Each LSTM cell contains a state vector and in each step determines the part of historical information to be forgotten, and filters out the relevant information to the prediction. By running the same LSTM, both the data provider and client will have the same predicted value in order to generate the symmetric key. By employing such a data prediction scheme, an eavesdropper and former service subscriber will not be able to decode messages due to the dependency to the latest data measurements. We finally note that unlike other functions that factor in the historical data, deep learning models capture the application semantics and also cannot be reverse engineered. While the complexity of LSTM could be viewed as high, we argue that pursuing offline training diminishes the computational complexity drastically. The model parameters as well as the initial data records are exchanged during a secure enrollment phase. The model still evolves over time based on the received data.

C. *Key Generation*

DACIC achieves pairwise authentication of the communicating party for each packet while ensuring data confidentiality against eavesdroppers. The client needs to ensure that the data it receives is from the correct provider; at the same time the provider makes sure that the packet can be decrypted exclusively by the intended receiver. To achieve such objective, the client can agree with the data server on a set of CRPs to use in the communication. In essence, the client can determine that the packet originated from the node that processes the correct response for its challenges. At the same time, the process of generating a key using the PUF of the client

ensures that only the client can decrypt the message. However, the per-packet keys will be thus dependent on the number of CRPs that a client shares with data sources, which will not scale for many-to-many communication traffic. Furthermore, sharing a restricted number of CRPs will make the key repetitive and degrade the system resilience to message replay attacks. DACIC overcomes the aforementioned issues by pursuing an innovative data driven key generation process. The fundamental idea is to factor in the recent packet exchange between a data provider and a client to derive a communication link fingerprint that ensures the authenticity of the next data packet.

DACIC employs symmetric keys per packet in order to keep the computational complexity low and sustain high resilience to message replay and impersonation attacks. A symmetric key is generated using the predicted data, data provider identity, and client authenticity code. The latter is in essence the response R_i of the client's PUF to a challenge bit stream C_i. As discussed earlier, the LSTM will factor in the most recent n data payloads to predict the next value; that is:

$$PD_{n+1} = \Omega(D_1, D_2, \ldots, D_n) \quad (1)$$

To generate the packet key, DACIC uses a simple function $H: g \rightarrow q$, as illustrated by Fig. 2. The input g to such a function reflects the concatenated bit patterns of the three factors, and the output is the key with some desired length q. The key length will be determined based on the encryption algorithm and on the capabilities of the involved data providers and clients in order to balance the high-security and low-overhead goals. Obviously, the same function H should be employed at the client and the data provider.

$$K = H(ID_x \,||\, PD_{n+1} \,||\, R_i) \quad (2)$$

where || is concatenation operator, ID_x is the identification number of a data provider S_x, and $R_i = F_y(C_i)$ is the PUF response for challenge C_i on client U_y. If the combined size of all three factors exceeds q, DACIC strives to fully utilize R_i and subsets of the bits of PD_{n+1} and ID_x. Meanwhile, if $q > |ID_x|+|PD_{n+1}|+|R_i|$, bit padding will be applied. By incorporating PD_{n+1}, K will be dependent on the data and implicitly authenticates the communicating parties. The value of n will be subject to trade-off; a small n will simplify the LSTM, yet it yields predicted value that may not vary if the actual data does not change frequently enough. Nonetheless, the shortcoming of a small n can be mitigated by varying C_i so that *the key, K,* becomes different for consecutive packets.

D. Communication Protocol

As noted in Section III, at the enrollment phase, a client U_y will share a small subset, ξ, of CPRs for its PUF with the data provider S_x. DACIC also requires that different CRP subsets are shared with the various providers, i.e., $\xi_{y,x} \neq \xi_{y,z}$, $\forall\ S_x \neq S_z$. Particularly, there has to be at least two CPRs, (C_d, R_d) and (C_p, R_p), that is shared with only S_x, i.e., $(C_j, R_j) \in \xi_{y,x} \,|\, (C_j, R_j) \notin \xi_{y,z}$ $\forall\ S_x \neq S_z, \wedge\ (j=d\ \vee\ j=p)$. To establish a session, U_y and S_x mutually authenticate each other through multiple CRP validation, where S_x sends a nonce encrypted with the response of one of the challenges C_j whose response R_j is only shared with S_x. Specifically, the data provider sends the nonce encrypted using R_d as follows:

$$\{C_d, [nonce]_{R_d}, C_p\} \quad (3)$$

The other challenge bit-stream, namely, C_p, that is included in (3) is to instruct U_y on what to use in the reply back. Upon reception of the message, U_y will decrypt the payload and extract the nonce. U_y will then re-encrypt the nonce using the response, R_p.

$$\{C_p, [nonce]_{R_p}\} \quad (4)$$

If U_y is a new client for S_x, there will be no historical record of shared data, i.e., it will be the first session for U_y. Hence, S_x will send an initial set of data items encrypted using R_d or R_p; recall that (C_d, R_d) and (C_p, R_p) are shared only with S_x. Alternatively, the nonce, exchanged challenges and corresponding responses can be fed to the LSTM to generate the first predicted data item. After the mutual authentication is complete, data delivery services will commence. In DACIC a client subscribes to a data delivery service; in this case the data provider will randomly pick a CRP, (C_i, R_i), for the client when forming each packet. The response R_i will be used in Eq. (2) to generate the encryption key, while C_i will be included in the payload, as shown in (5), so that the client knows how to decrypt the data. Upon successful reception, the client will send an acknowledgment. Fig. 3 summarizes the aforementioned protocol steps.

$$Payload_{Push} = \{C_i, [D_{n+1}]_K\} \quad (5)$$

Finally, we note that each of the data payloads in (3) – (5) will include a checksum before encryption in order to enable detection of bit errors. This is typical for detecting wireless transmission errors; yet it is particularly important in the context of DACIC to mitigate noise errors that affect the PUF response. In essence, receiving a packet that cannot be correctly decoded will lead the client to ask for retransmission. Therefore, DACIC will ensure the synchronization of the historical data record on the provider and the client.

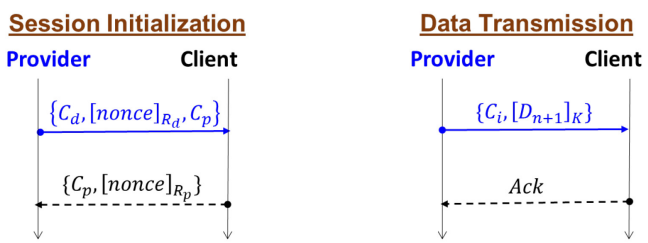

Fig. 3. A sequence diagram summary of DACIC's communication protocol.

V. Security Analysis

In this section, we show that DACIC achieves the desired security properties and mitigate known attacks.

Node Impersonation: An adversary may claim to be the provider during the session establishment or data packet transmission. By picking challenge bit-streams in (3) and (4) that are known only to the specific data provider S_x, another provider or an eavesdropper cannot decrypt the messages and extract the nonce. The probability that the adversary could guess the correct response to find the nonce is $1/2^m$ where m is the size of the response bit-stream. If the adversary fails to know such a response, establishing a session will not be possible.

Meanwhile, to impersonate a provider during data transmission, the adversary has to generate the exact key. An eavesdropper does not have access to R_i, the ML model and to the last n data items (packet payloads) used by such a model. A packet that is encrypted using a random key will be wrongly decrypted and detected by the client using the checksum. When the attacker is an internal client U_a and thus knows the ML model, such an attacker does not know R_i of the legitimate client U_y; each Client's PUF provides distinct responses which prevent U_a from generating the correct key and impersonating the provider. Finally, an internal attacker is another provider S_a that knows some CRPs for the client; such an attacker will fail to know PD_{n+1}, since it does have the actual data. The use of LSTM is advantageous here as S_a does not have the model parameters and cannot even guess the value of PD_{n+1}.

On the other hand, given the usage of the PUF, it is not possible for an attacker to impersonate a client during session establishment, as noted above. In addition, impersonating a client during data transmission is not viable since the provider transmits the data and no information is sent by the client.

Replay attack: Being unable to decrypt packets, the attacker may resend them at a later time, which could: (i) degrade the client's analysis by providing outdated data, and (ii) prevent the synchronization between the communicating parties, and lead to denying access for legitimate clients. Replaying a provider's data message will be detected by the client due to the variability of the key over time. The provider's packet will be encrypted using distinct keys that depend on the PUF response and the ML model output; thus it is impossible to reverse the key generation process and decrypt the packet payload. Even if a CRP usage is intermittently repeated, the response is still unknown to the adversary and also the considered data in the ML model differs. Thus even internal attacker cannot mimic the legitimate client as the attacker does not have access to the latest PUF response. For the synchronization issue it can be proven by recurrence. Indeed, any replied packet can be detected by the client. Thus consecutive replayed packets can be detected and thus the attacker cannot cause desynchronization.

False data injection: To modify the data, the adversary needs to first crack the keys using cryptanalysis in order to decrypt the packet payload; this is extremely hard since each packet will have a unique key. In case the data does not vary over a number of consecutive packets and the output of the ML model stays unchanged, the used CRP will differ and consequently the keys will not be the same. Thus, any data manipulation attempt will not succeed as the adversary will not be able to generate the keys and incorporate any falsified information in the packet. Any attempt to generate random keys can be easily uncovered by the client.

PUF Modeling Attack: In DACIC, the client shares a limited number of CRPs with the individual data providers in order to prevent modeling attacks. We note that DACIC assumes that data providers cannot be manipulated. Nonetheless, even a malicious provider cannot accurately model the PUF of any of its data subscribers. Moreover, an eavesdropper will not be able to know the response of the used challenge bit-streams since they are used in key generation and are not sent explicitly in any of the packets. The keys vary per packet and hence even using cryptanalysis, the response cannot be inferred from the key.

VI. Validation Experiments

To validate the effectiveness of DACIC, we considered two example of data sharing applications, namely, smart telehealth and smart grid. For the telehealth system, we used the ECG samples extracted from PhysioNet dataset [19]. For smart grid measurement, we used simulated data using MatPower for real loads [20]. For both applications, an LSTM with three cells has been used in DACIC. The function H reflects the scrambling of the input bits. To validate the key variability over packets, we measured the degree of similarity of keys of the same client over time and between every pair of clients. The metric used to assess the similarity is the Levenshtein distance. This is a metric for assessing similarity between strings. It measures the difference between two sequences of characters based on the minimum number of characters to insert, delete or substitute in order to match two given strings. We further evaluated the security properties of DACIC using the AVISPA toolset. Finally, we report the runtime complexity of DACIC compared to contemporary crypto systems.

Key uniqueness: Fig. 4(a) capture the similarity of the keys generated for the same provider as a matrix; the matrix is diagonal implies that the keys are unique. Fig. 4(b) shows the similarity matrix for the keys for pairs of providers. This variability of the keys per packet is due to the difference in the transmitted data and the employed PUFs. The results are consistent for both telehealth and smart grid applications, which demonstrates independence of the underlying data.

Computational Overhead: DACIC is a lightweight data sharing framework. This property is due to the use of a symmetric cryptosystem. We have compared DACIC with prominent asymmetric techniques. Fig. 5 demonstrates the effectiveness of DACIC in terms of execution time in comparison to RSA or ECC using 1024-bit keys. The results reflect the encryption of packets of 70 ECG records using the DES algorithm. The results clearly demonstrate the advantage of DACIC.

Security Properties: We have employed AVISPA to verify the security properties and the vulnerability of DACIC. AVISPA is a widely-used formal security verification framework. We have described data exchange protocols using the underlying High Level Protocol Specification Language, and defined the client and data provider roles, all possible states and transitions including data exchanges and initializations and enrollment steps. The security goals for the AVISPA simulation are the authentication of clients and the provider and secrecy of the data and the keys. The OFMC results demonstrate the robustness of DACIC against eavesdropping, man in the-middle, replay and impersonation attacks. Figure 6 is screenshot of the output of the analysis, confirming the safety of the protocol of communication and data transmission.

VII. Conclusions

This paper has presented DACIC, a novel lightweight authentication and access authorization protocol. DACIC adopts a data-driven methodology and hardware fingerprints for

secure communication in multi-access data sharing systems. The idea is to generate encryption keys that vary per packet and in an implicitly synchronized manner between the data provider and each client. The generated encryption keys factor in hardware primitives to support authentication of legit data recipients, and the shared data pattern and previously transmitted data. We have validated the performance of DACIC and analyzed its resilience against impersonation, data manipulation and message replay, and hardware primitive modeling attacks. Our future plan is to extend DACIC to mitigate tempering attacks.

REFERENCE

[1] B. Ahlgren, M. Hidell and E. C. -H. Ngai, "Internet of Things for Smart Cities: Interoperability and Open Data," *IEEE Internet Computing*, vol. 20, no. 6, pp. 52-56, Nov.-Dec. 2016.

[2] Y. Yang et al., "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Jour.*, vol. 4, no. 5, pp. 1250–1258, 2017.

[3] A. Ahanger and A. Aljumah, "Internet of things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2019.

[4] X. Liu et al., "A Security Framework for the Internet of Things in the Future Internet Architecture," *Future Internet*, vol. 9, No. 3, p. 27, 2017.

[5] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," *Proc. of DAC*, 2007, pp. 9–14.

[6] A. Shamsoshoara et al., "A survey on physical unclonable function (puf)-based security solutions for internet of things," *Computer Networks*, vol.183, p. 107593, 2020.

[7] P. Gope and T. Hwang, "BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network," *IEEE Sensors Journal*, vol. 16, no. 5, pp. 1368-1376, March 2016.

[8] A. Ross, S. Banerjee, and A. Chowdhury, "Security in smart cities: A brief review of digital forensic schemes for biometric data," *Pattern Recognition Letters*, Vol. 138, pp. 346-354, 2020.

[9] A. Anand and A. K. Singh, "An improved DWT-SVD domain watermarking for medical information security," *Computer Communications*, Vol. 152, pp. 72 - 80, 2020.

[10] 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.

[11] G. Zheng et al., "Multiple ECG Fiducial Points-Based Random Binary Sequence Generation for Securing Wireless Body Area Networks," *IEEE Journal of Biomedical & Health Info.*, vol. 21, no. 3, pp. 655-663, 2017.

[12] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-Based Secure Communication Protocol for IoT," *ACM Trans. Embedded Computing Systems*, vol. 16, no. 3, pp. 1-25, 2017.

[13] M. Aman et al., "Mutual Authentication in IoT Systems Using Physical Unclonable Functions," *IEEE IoT J.*, vol. 4, no. 5, pp. 1327–1340, 2017.

[14] U. Chatterjee et al., "Building PUF Based Authentication and Key Exchange Protocol for IoT Without Explicit CRPs in Verifier Database," *IEEE Trans. on Depend. & Sec. Comp.*, vol. 16, no. 3, pp. 424–437, 2019

[15] J. R. Wallrabenstein, "Practical and Secure IoT Device Authentication using Physical Unclonable Functions," *Proc. IEEE 4th Int'l Conf. on Future Internet of Things and Cloud (FiCloud)*, pp. 99–106, 2016.

[16] S. Li, T. Zhang, B. Yu and K. He, "A Provably Secure and Practical PUF-Based End-to-End Mutual Authentication and Key Exchange Protocol for IoT," *IEEE Sensors Journal*, 21(4), pp. 5487-5501, Feb. 2021.

[17] F. H. Al-Naji, R. Zagrouba, "A survey on continuous authentication methods in Internet of Things environment," *Computer Communications*, Vol. 163, pp. 109-133, 2020.

[18] W. Lalouani, M. Younis, I. White-Gittensb, R. N. Emokpae, Jr. and L. E. Emokpae "Energy-Efficient Collection of Wearable Sensor Data through Predictive Sampling," *Smart Health*, Vol. 21, July 2021, 100208.

[19] V. Novak et al., "Cerebral flow velocities during daily activities depend 1137 on blood pressure in patients with chronic ischemic infarctions," *Stroke*, 1138 vol. 41, no. 1, pp. 61–66, 2010.

[20] R. D. Zimmerman, C. Murillo-Sanchez, and R. J. Thomas, "Matpower's extensible optimal power flow architecture," *Proc. the IEEE Power and Energy Soc. General Meeting*, pp. 1–7, Calgery, Alberta, Jul. 2009.

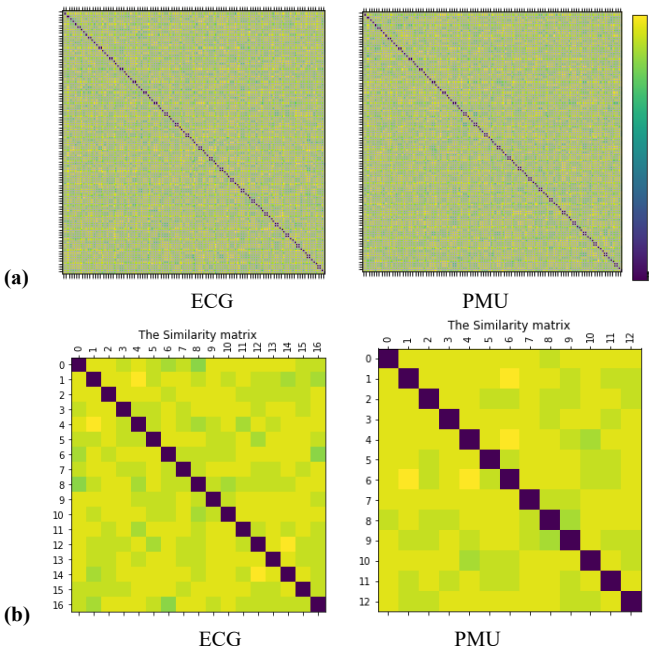

Fig.4. Similarity of DACIC keys for: (a) the same provider, and (b) distinct providers.

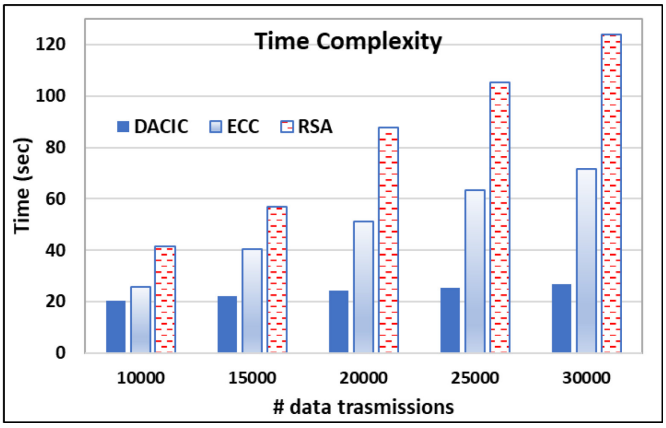

Fig.5. The time complexity of DACIC compared to RSA or ECC algorithms

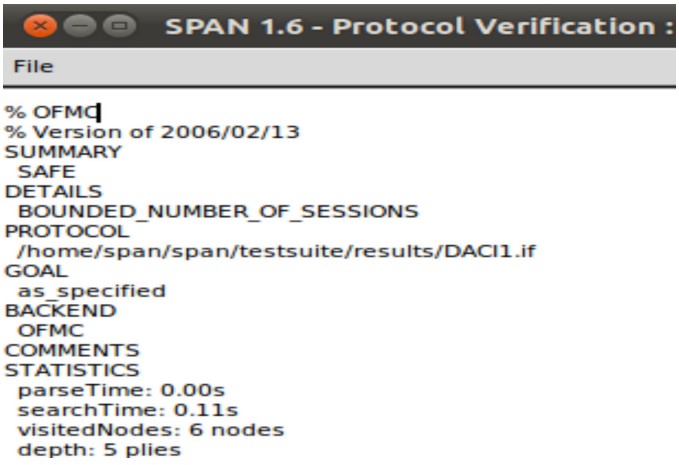

Fig. 6. Screenshot of the OFMC output, confirming the robustness of DACIC