M. Ebrahimabadi, M. Younis, W. Lalouani, A. Alshaeri and N. Karimi, "SWeeT: Security Protocol for Wearables Embedded Devices' Data Transmission," *2022 IEEE International Conference on E-health Networking, Application & Services (HealthCom)*, 2022, pp. 135-141, doi: 10.1109/HealthCom54947.2022.9982744.

https://doi.org/10.1109/HealthCom54947.2022.9982744

SWeeT: Security Protocol for Wearables Embedded Devices' Data Transmission

Mohammad Ebrahimabadi*, Mohamed Younis*, Wassila Lalouani†, Abdulaziz Alshaeri*, Naghmeh Karimi*
*Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC)
†Department of Computer and Information Science, Towson University

Abstract— **Motivated by the quest for decreased healthcare costs and further fueled by the COVID pandemic, wearable devices have gained major attention in recent years. Yet, their secure usage and patients' privacy continue to be concerning. To address these issues, the paper presents SWeeT, a novel lightweight protocol for allowing flexible and secure access to the collected data by multiple caregivers while sustaining the patient's privacy. Particularly, SWeeT deploys Physically Unclonabale Functions (PUFs) to generate encryption keys to safeguard the patients' data during transmission. The computation overhead is significantly reduced by applying very simple encryption operations while enabling frequent change of the keys to sustain robustness. SWeeT is shown to counter impersonation, Sybil, man-in-the-middle, and forgery attacks. SweeT is validated through experiments using implementation on an Artix7 FPGA and through formal security analysis.**

I. Introduction

The major advances in microelectronics and wireless technologies have enabled the development of wearable systems where sensors could be attached or implanted in a human body to assess vital conditions and share the collected measurements over radio links [1]. Prevalence in the use of wearable devices is deemed an effective strategy for coping with the rising cost of healthcare services where patients are allowed to live normal life while their status is being remotely monitored by caregivers. Such a strategy has attracted even more attention in recent years due to the COVID pandemic where many medical services shifted from face-to-face to virtual interactions to promote physical distancing and protect the healthcare professionals. Indeed, providing a continuous service is of utmost importance specially for the cases where the patient symptoms and recovery need to be checked frequently by physicians.

Given their current and future role, wearable systems can be viewed as safety critical cyber-physical systems where actions can be taken autonomously, e.g., pacemaker, or remotely based on physician's advice. This in turn can impose security and privacy concerns for the data that these wearable devices gather and the way such data is shared with the caregivers. In practice, security vulnerabilities on wearable devices can lead to safety concerns for users [2]. One such example is the pacemaker with wireless capabilities which has been shown to be vulnerable against the software radio-based attacks [3]; thus compromising patient's safety and privacy.

Generally, telehealth systems are prone to impersonation, man-in-the-middle, eavesdropping, and data forgery (packet injection) attacks [4]. To tackle the adversaries who target the security, and privacy of the patient's data, several methods have been proposed in literature. These methods mainly rely on employing cryptosystems. However, the asymmetric conventional cryptographic algorithms are prohibitively costly for wearable devices. The symmetric counterparts need to store the data encryption keys used in the memory of wearable devices which in turn can be hacked; in addition a key management protocol needs to be employed to vary the keys overtime as a means for increased robustness against cryptanalysis.

This paper opts to fill the gap by proposing a lightweight and effective security solution for remote data collection from wearable devices. The proposed solution, which is referred to as SWeeT, takes advantage of hardware-based security primitives, namely Physically Unclonabale Functions (PUFs), to generate fresh random keys on the fly without the need to store them on the wearable device. A PUF exhibits unique fingerprinting; hence when embedded in the design of a wearable device it constitutes a tamper-proof identity. PUFs are realized by leveraging the process variations occurring during the fabrication process of the integrated circuits [5]. In particular, SWeeT benefits from one type of PUFs called arbiter-PUFs that covers a large set of input-outputs which is referred to as Challenge-Response Pairs (CRPs). SWeeT employs an effective protocol for protecting the patient data during transmission while imposing very little computational overhead on the devices. It devises a key generation scheme that benefits from the uniqueness and randomness of the incorporated PUFs. Only simple XOR operations are applied to encrypt data and the generated keys are varied on a per-session basis to sustain robustness. SWeeT safeguards the telehealth system against adversaries who eavesdrop on the communication links or exploit leaked security credentials that a patient provides to one or multiple physicians. In summary, our contributions include:

- Devising SWeeT, an effective PUF-based protocol that enables secure transfer of wearable devices data while safeguarding against PUF modeling attacks;
- Metigating the impact of PUF measurement noise and the temperature-change induced noises on the robustness of the session keys generated by SWeeT and in turn the correctness of the decrypted data.
- Analyzing SWeeT's resilience against various attacks, e.g., data forgery, impersonation, and man-in-the-middle;
- Evaluating the proposed method using the data extracted from FPGA implementation of the target PUFs.

II. Related work

Security, privacy, and integrity of wearable health data is of utmost importance. To achieve secure data transfer, Turner et al., [6] have proposed the so-called Transport Layer Security (SSL/TLS) protocol that employs an asymmetric cryptography algorithm; thus SSL/TLs imposes a high computation overhead which makes it unfit for the resource-constrained wearable devices. Similarly, Elliptic Curve Encryption has been used to secure medical devices against the adversaries who target patients' privacy [7], [8]. Although it is secure against contemporary attacks, e.g., man-in-the-middle, impersonation, and message replay, and is more lightweight than SSL/TLS, the imposed computational overhead is still too significant for a wearable medical device. Classic symmetric cryptographic schemes to secure the transfer of wearable devices' data is not promising either as storing the encryption key in an embedded non-volatile memory makes the device tempering and hacking possible [9], [10]. Alternatively, Trusted Platform Module [11], [12] based protection schemes impose hardware complexity and are not suitable for resource-constrained wearable devices.

Thanks to being lightweight and constituting a unique electronic signature, Physically Unclonable Functions (PUFs) is deemed a promising means for generating secret keys. However, PUF-based protocols can be vulnerable to contemporary attacks such as behavior modeling, message replay, and impersonation [13]. To mitigate vulnerability to these attacks, Chatterjee et al. [14] have proposed using a combination of PUFs and asymmetric cryptographic modules. Although secure and resilient against modeling attacks, the proposed scheme imposes high computational overhead which makes it unfit for an wearable device. To counter PUF modeling attacks, Aman et al. [15] make each submitted challenge a function of the previously queried one; yet leaking one challenge makes the overall scheme vulnerable against impersonation attacks.

Using patient biometrics to generate encryption keys for securing the transfer of telehealth data has been pursued. For example, Pirbhulal et al. [16] generate a cryptographic key based on the heartbeats extracted from Electrocardiogram (ECG) signal. However this method is prone to key guessing attack [17]. Similarly, Zheng et al. [18] extract some features of ECG signals to devise patient-specific cryptographic keys. However, such a method cannot be used for patients with cardiovascular diseases, and hence may not be practical [19]. Employing PUFs and patient biometrics in generating encryption keys is proposed in [20]. Overall the involvement of biometrics requires the incorporation of machine learning models that are often computationally heavy for wearable medical devices. The proposed SWeeT approach overcomes the shortcomings of existing techniques and enables a lightweight and effective mechanism for securing data transfer in telehealth applications.

III. System Model and Preliminaries

A. Arbiter-PUF Architecture

An arbiter-PUF consists of a pair of delay chains; when queried with a challenge bit-stream, it generates one response bit [21]. PUFs operate based on variations in the microelectronics manufacturing process; in the case of Arbiter-PUF these variations introduce a race between two identical paths (top and bottom paths shown in Fig. 1). The race corresponds to the difference in signal propagation delay on these two paths, and affects the value latched by the arbiter. Only the sign of delay difference, rather than the exact value, is important. The sign, extracted by the arbiter, reflects the response and constitutes the PUF identifier. The arbiter can be realized as a simple SR-latch implemented by two cross-coupled NOR gates. Note that each PUF instance generates a 1-bit response (shown as r in Fig. 1). A multi-bit response (shown with R hereafter) is extracted either by instantiating multiple samples of PUFs in the hardware or by embedding only one PUF sample yet querying it multiple times with different challenges. In this paper, we consider the latter case to lower the area overhead.

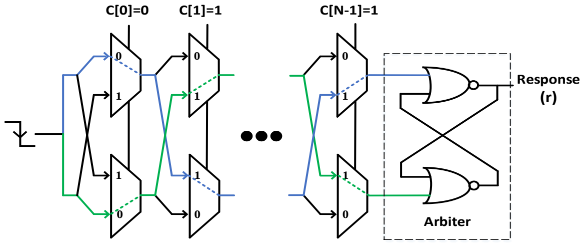


Figure 1: Illustrating the design of an arbiter-PUF.

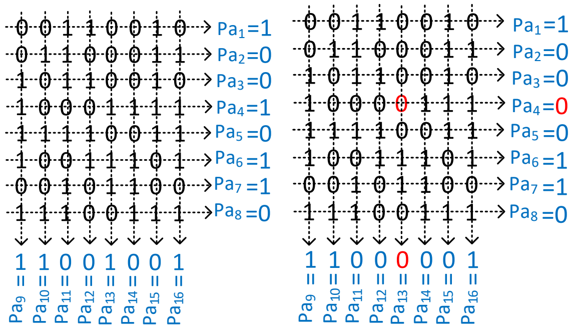
B. Error Correction via Two-Dimensional Parity Coding

As discussed earlier, delay-PUFs, and in particular arbiter-PUFs, operate based on the delay characteristics of the embedded circuit. Thus, environmental variations (e.g., voltage and temperature change) may affect the PUF response by changing the propagation delay of the underlying components. For example, if the delay magnitude of the blue and green paths shown in Fig. 1 are very close to each other, a voltage- or temperature-change may change the outcome of the race occurring between these 2 paths, and accordingly the PUF response deviates from what is expected. As will be discussed in Sec. IV, to secure the data transfer, in this research we opt to use the PUF signature (response R generated for challenge C) as a session key to encrypt the wearable device data during communication. Thereby, the noise-caused changes to the PUF response need to be tolerated to be able to properly decrypt the transmitted data at the receiver side. As discussed in the next section, our proposed SWeeT approach uses a fresh key per data transfer session in order to prevent replay and impersonation attacks. Such a key will be corrected, if noisy, using a multi-dimensional parity scheme.

To mitigate the PUF noise effects, SWeeT benefits from the incorporation of Error Correction Codes (ECC), in particular a 2-Dimensional parity coding [22] since the associated ECC is separate from the data and can thus be shared without the data itself. In this ECC scheme, each block of data (PUF responses in our case) is represented as a 2-dimensional matrix, and a parity bit is generated for each row and column. Fig. 2(a) shows an example of deploying the 2-dimensional parity for the sample data of $64'x3263B28FF39D2CE7$. As depicted, we first place the data in a 8×8 matrix and then calculate the parity for each column and row. In Fig. 2(b) we assume that a data bit is noisy (located in row 4 and column 5). In this case the parities for both the 4^{th} row and 5^{th} column change. By comparing the parities of the original data with the one generated for the faulty data, one can find the fault location and toggle the corresponding bit. A 2-dimensional parity can correct 1-bit errors. However, as discussed in Sec. IV, with some additional analysis we also can correct errors in multiple bits. As supported by the experimental results in Sec. V, errors that affect more than 2 bits are very infrequent and would not warrant complicating the design.

C. System Architecture

Our proposed solution employs hardware-based identifiers to enable secure transmission of patient's data to the health



(a) Genuine Data (b) Erroneous Data

Figure 2: Two-Dimensional parity for a sample genuine and an erroneous data block. The bit located in the (row=4, column=5) location is faulty; reflected through Parity bit 4 and 13 (i.e., Pa$_4$&Pa_{13}).

service providers (referred to as physician hereafter). We assume that multiple wearable medical sensors are used to measure the physiological biometrics of the patient, and all measurements are aggregated in a processing unit at the patient side, denoted by On-Patient Gateway Node (PGN). The aggregated data is in turn transferred to the physician upon receiving a request. Figure 3 articulates the system architecture for SWeeT. Without loss of generality, this figure shows the case where the patient receives care from N=2 physicians. As depicted, the PGN records all measurements acquired from the patient's wearable devices, e.g., patients' ECG, EMG (stands for Electromyography), temperature, heartbeat, etc.

SWeeT deploys PUFs to devise a robust encryption method to secure the data transfer and prevent data leakage. To mitigate the overhead, SWeeT does not need to include a PUF in each wearable device; rather it only includes 2 PUFs in the PGN. These PUFs are used to generate the encryption keys, as explained in the next section. To initiate a data transfer, the physician sends a data request to the patient. This request includes a challenge bit-stream. Upon receiving the physician's request by the PGN, the challenge is applied to the PUFs. The patient data is then encrypted via the key generated based on the response of the two embedded PUFs, and is sent back to the physician. The physician decrypts the received data using the key tabulated at the time the patient registered with the physician. In practice, a patient may receive service from multiple care providers, therefore in our system model, each patient needs to be registered by each physician individually to receive the related services. Accordingly, each physician has his/her own database to tabulate the PUF responses (encryption key) for each patient.

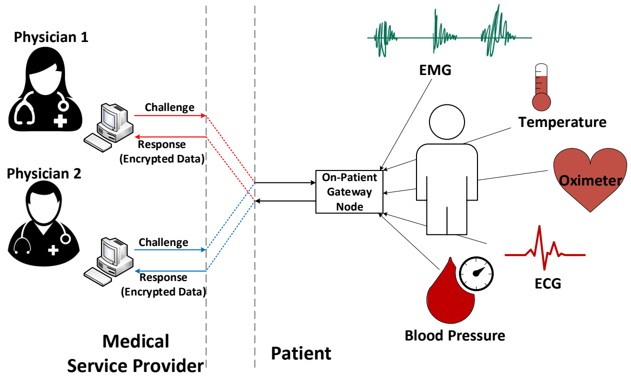


Figure 3: An overview of the target framework.

D. Threat Model

We consider an adversary who is eager to illegally access the patient data. The adversary tries to achieve such a goal by either eavesdropping on the wireless links between the patient and the physician, or hacking the physician's computer system. Although the adversary's attempt, if successful, would suffice for accessing the data, such success is short lived and/or limited in scope as: (i) different sensing modality could be of interest to different physicians, and hence cracking the computer system of physician Ph_i is not enough for getting all patient's data; (ii) a patient often switches physicians over time, and consequently the illegal data access will not be sustained. Therefore, the adversary opts to uncover the security primitive of the patient's PGN rather than just hacking the computer system or the communication links of a certain physician. In the context of SWeeT, the adversary will strive to collect CRPs for the embedded PUFs so that an accurate behavior model is formed and used to predict the responses of unknown challenges. The modeling, which is often conducted through the application of Machine Learning (ML) techniques, will allow the attacker access to the data that the patient shares with other physicians both at the current time and in the future.

In sum, the goal of the adversary is to uncover the encryption keys by collecting sufficient challenge to model the behavior of the embedded PUFs using ML techniques. The adversary will attempt to eavesdrop on the wireless links between patient P_x and physicians to intercept the exchanged packets and infer the challenge and response bits, and/or hack the computer system of physician Ph_k to read the CRPs stored in the memory. If the CPRs that Ph_k has, do not suffice to model the PUF, the adversary may target one or multiple additional physicians, i.e., Ph_j (where $k \neq j$). The latter scenario is referred to as collusive attack where the adversary combines the gained knowledge from multiple compromised physician systems or communication links that originate at the patient in order to grow the list of uncovered CRPs and devise an accurate behavior model for the patient's security provisions. SWeeT strives to counter the aforementioned attack scenarios, as detailed next.

IV. Secure Data Transmission Protocol

This section presents the detailed design of the proposed SWeeT protocol. Before describing the protocol steps, we provide an overview of the underlying methodology and how it meets the requirements.

Design Overview: Transfer of a patient's wearable sensor data to a caregiver should achieve two fundamentally objectives: (i) confidentiality in order to sustain patient's privacy, and (ii) integrity to ensure correct patient conditions assessment by the physician. As pointed out, the use of conventional cryptosystems imposes computational overhead that would require increased device complexity and cost. Hence, a third requirement for the data transfer protocol is to be lightweight. Finally, a patient could be served by multiple physicians either at the same time based on the sensor modality, e.g., EEG, ECG, heart sound, etc., or over periods of time as patients change their treating doctors. That leads to a fourth requirement of sustaining patient privacy across different physicians.

SWeeT meets the above requirements by employing PUFs and devising a simple data sharing protocol. First, a PUF is employed where data encryption keys are generated based on the PUF responses of some challenge bit streams. Each physician will be given the challenges and the corresponding keys. To limit the computational overhead, the encryption is simply performed through applying an XOR for the data and key. To mitigate the vulnerability of such simple encryption, the key is varied per session (request) and the request is controlled by the physician. Thus, for an adversary, it becomes a moving target, and insufficient patterns will be available for cryptanalysis. To ensure secrecy across physicians, a second PUF is employed where the physician ID is used as a challenge (input) and the PUF output is further factored in the key generation. In other words, if two physicians, Ph_j and Ph_k (where $k \neq j$) use the same challenge bit pattern, C, SWeeT generates two distinct keys for them. Employing the second PUF, enables SWeeT to safeguard the system against collusion attacks involving multiple malicious physicians, who are impersonated or whose computers were compromised by an attacker.

Registration Phase: Before data can be accessed by a physician, registration has to be conducted. In such a registration phase, some security credentials have to be provided to the physician either by the patient or by a trusted server on behalf of the patient. Algorithm 1 summarizes the steps taken by SWeeT during the registration phase. All interactions during the registration phase are assumed to be secure, e.g., by using public-private key cryptography. Basically, the patient could be connecting its PGN to a personal computer (PC) during the registration phase, where the PC can apply an elaborate encryption algorithm on behalf of the PGN (which does not have the computational resources to do so under SWeeT).

First, the physician, Ph_i, establishes a connection with the PC of a patient, P_x. The latter will create a set of randomly generated challenge bit-patterns; let $S^C_{x,i}$ denote such a set. The PC of P_x will then query the PGN for the key corresponding to each challenge $C \in S^C_{x,i}$. To generate the key for a challenge C, the PGN will first apply C to one of its embedded PUFs, namely, PUF_1 (Fig. 4). To make the session key for each challenge unique per physician, the PGN uses a second PUF, denoted PUF_2, and feeds it with the unique ID of the physician. Finally, the key is formed based on the two PUF responses, namely, R_C and R_{ID}.

We note, however, that if the same data is shared in multiple sessions, the adversary may be able to infer the key and then use the uncovered keys to model the PUF behavior using ML techniques and extract the key for each session. This scenario cannot be ruled out since in reality some of the sensor measurements may not change frequently. Therefore, SWeeT employs another challenge, P, whose response, R_P is used to permute the concatenated string of R_C and R_{ID}. In other words, SWeeT again applies P to PUF_2 and then uses a $key\_generator$ function that change the order of bits of R_C || R_{ID} based on the value of R_P; the result constitutes the session key corresponding to ID, C and P. Hence, along with $S^C_{x,i}$ and the set of corresponding keys, $\Omega^C_{x,i}$, the patient P_x will share another set $S^P_{x,i}$ with Ph_i reflecting some randomly generated values of P. Each P is applied to PUF_2, and the response R_P is then used to determine its 2-dimensional parity bits, ζ_P. The latter is sent to Ph_i to be used during the run-time phase. Basically, at time of data request (i.e., run-time), ζ_P is provided by the physician to P_x along with C, ID, and P to mitigate the effect of PUF noise when generating R_P.

Algorithm 1: Registering Physician Ph_i by Patient P_x

```
input : Number of CRPs ($N$)
input : Physician's identifier ($ID_i$)
output: $S^C_{x,i}$,$S^P_{x,i}$,$\Omega^C_{x,i}$, $ECC^P_{x,i}$
$R_{ID} \leftarrow PUF_2(ID_i)$
for ($k < N$) do
    Generating a randomly selected C and P
    Add $C$ to $S^C_{x,i}$, and $P$ to $S^P_{x,i}$
    $R_C \leftarrow PUF_1(C)$
    $R_P \leftarrow PUF_2(P)$
    $key \leftarrow key\_generator(R_C, R_P, R_{ID_i})$
    $\zeta_P \leftarrow Parity\_gen(R_P)$
    Add $key$ to $\Omega^C_{x,i}$, and $\zeta_P$ to $ECC^P_{x,i}$
return $S^C_{x,i}$, $S^P_{x,i}$, $\Omega^C_{x,i}$, $ECC^P_{x,i}$
```

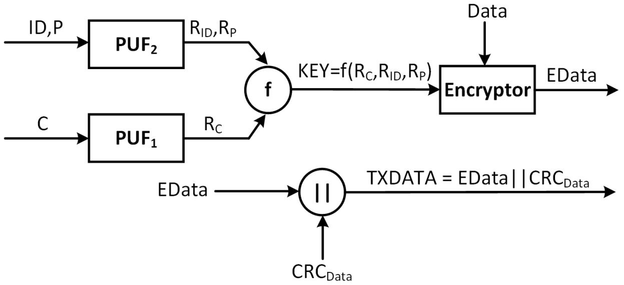


Figure 4: Block diagram of the encryption mechanism in patient side.

Run-time Phase: Fig. 4 shows the overall encryption process performed by the patient's PGN during data transfer to the physician. Each physician may request new measurement data from the patient by sending a challenge (C) as well as the permutation index (P). Upon receiving the query, the PGN infers the physician's ID from the packet and uses it along with C, and P to generate an encryption key. Factoring in the physician ID prevents sharing the measurements with unauthorized recipients, even other physicians. The patient would then transfer the measurements, e.g., heartbeat rate, in an encrypted form to the requester according to Algorithm 2.

As pointed out earlier, SWeeT sustains the integrity of the data and confidentiality of the patient by encrypting the transferred data; yet unlike existing work SWeeT avoids elaborating cryptographic algorithms, both symmetric and asymmetric, and instead employs a simple XOR operation of the data and a key. Robustness against cryptanalysis is enabled by changing the key per session and per patient. The key is generated at the patient side instantaneously without storage to avoid any potential leakage. The role of C and ID in the key generation has been already discussed in the registration phase, we here focus on how P is factored in the process and why it is useful.

To motivate the need for P, let's assume that both Ph_i and Ph_j requested the same data and submitted the same C. In such a case, an attacker could correlate packets sent from the patients to Ph_i and Ph_j and uncover the R_C as well as R_{IDi} and R_{IDj} since key is just a concatenation of R_C and R_{ID}. To mitigate such vulnerability, SWeeT introduces unpredictability in how the key is defined based on R_C and R_{ID}. By using R_P as means for defining the key, the mixing of R_C and R_{ID}

bits becomes dependent on the PUF (unfixed or stored at the patient's PGN). Moreover, the physician does not even know R_P and cannot reverse engineer the key formation process despite being able to decrypt the patient data. Since R_P is critical in forming the key it is very important that the effect of noise be tolerated. To do so, at the time of registration, the patient gives the physician both P and ECC_P. The latter will be included in the data request so that the patient can correct R_P if needed. Such an approach is not considered as leakage since the physician does not know R_P. Note that ECC_P is extracted from R_P; not P itself.

While $|R_P|$ could be generally M bits (size of PUF response), in practice there is no need to consider numerous options for mixing R_C and R_{ID}. Hence only a subset of the bits of R_P could suffice. Without loss of generality, let's assume that only two bits of R_P are used in the key generation process. Alg. 2 shows an example of how to concatenate R_C and R_{ID} with different permutations based on the value of R_P. Note that in Algo. 2, we only considered the 2 Least Significant Bits of the R_P for devising the permutation strategy. However, one can involve more bits with a negligible overhead.

Finally, to mitigate the effect of noise on R_C and R_{ID}, and consequently the key, SWeeT includes the 2-dimensional parity of the generated key, denoted P_{Key}, in the data packet to the physician. The main purpose of P_{Key} is to guide the physician on adjusting its tabulated key to what the patient used. In other words, P_{Key} does not correct bit flips in the PUF response due to noise, instead it helps the physician know how to adjust the expected key to what is actually used. This point will be elaborated in the next subsection. Moreover, the CRC of the data (CRC_{Data}) is sent along with the encrypted data ($EData$) to the physician. Such CRC will allow the physician to validate the integrity of the data, especially if key adjustment is performed based on P_{Key}.

Algorithm 2: Data packet encryption by the patient

```
Input  : Queried packet from physician ($ID||C||P||ECC_P$)
Output: $EData$, $P_{Key}$, $CRC_{Data}$
$R_C \leftarrow$PUF$_1(C)$
$R_{ID} \leftarrow$PUF$_2(ID)$
$R_P \leftarrow$PUF$_2(P)$
$R_P \leftarrow ECC($R$_P$,$ECC_P)$
if $R_P[1:0] = 0$ then
    $Key$ = $R_C$ || $R_{ID}$
else if $R_P[1:0] = 1$ then
    $Key$ = $R_{ID}$ || $R_C$
else if $R_P[1:0] = 2$ then
    $Key$ = $R_C[l-1:0]||R_{ID}||R_C[m-1:l]$
else if $R_P[1:0] = 3$ then
    $Key$ = $R_{ID}[l-1:0]||R_C||R_{ID}[m-1:l]$
end
$EData \leftarrow XOR(Key, Data)$
$P_{Key} \leftarrow Parity\_gen(Key)$
return $EData$, $P_{Key}$, $CRC_{Data}$
```

Noise Mitigation: SWeeT employs two levels of error corrections to mitigate the noise impact on the PUF response and ensure the robustness of the session keys generated by the patient's PGN. We treat the noise related to generation of R_P in the PGN itself (hardware level) since a change of one bit in R_P could result in a different scrambling pattern of the bits of R_C and R_{ID} within the session key and thus prevent proper decryption of the transmitted data by physician. However, to reduce the computational burden on the PGN, the noise effect on R_{ID} and R_C is treated at the physician's side in software.

As explained earlier, during the registration phase, the physician tabulates the 2-dimensional parity of R_P along with the corresponding Key. Such a parity scheme can tolerate one-bit errors. As the experimental results in Sec. V depicts, the probability of 2-bit (or more) measurement noise is very low ($\approx 0.7\%$) in room temperature and hence the provisioned protection suffices most of the time. In case of 2 (or more) bit errors, the 2-dimensional parity cannot directly pinpoint the exact erroneous bit; rather it can suggest the erroneous bit candidates which need to be checked subsequently to determine the exact faulty bit. For example in Fig. 2(a) if the bit in row 3 and column 4 as well as that in row 5 and column 2 are simultaneously erroneous, then the parity bits Pa_3, Pa_{12}, Pa_5, and Pa_{10} will change which brings ambiguity that the errors are either in locations (3,4) and (5,2) or in locations (3,2) and (5,4). Note that even in the case of 2 (or more) bit errors in R_P the packet is detected as erroneous by the physician; thanks to using the CRC of data. In that case the packet is discarded and data re-transmission is requested.

On the other hand, the noise affecting R_C and R_{ID} (and in turn Key) is treated at the physician's side (software level) to avoid loading the PGN. To do so, we need to generate the corresponding parity bits during runtime in the wearable devices and send it to the physician along with the encrypted data to remove the impact of the noise accordingly (P_{Key} in Alg. 2). In this case we can correct multiple-bit errors; thanks to the transferred CRC bits of the original data.

V. Experimental Results and Discussions

A. Experiment Setup

We have validated SWeeT using 2 Xilinx ARTIX7 FPGA boards, each representing one patients' PGN, and using two PCs to mimic two physicians. We used Zigbee transceivers to connect the physicians to patient nodes. We assigned a 64-bit unique number as the ID to each physician. We assume both PUF_1 and PUF_2 have 64 bit challenges and each generate a 64-bit response. The adversary is assumed to either get access (via hacking a Physician's PC) to a subset of the stored challenges and keys, or eavesdrop on the communication links between the patient and physician (to intercept the challenge and the encrypted data) and perform cryptanalysis to extract the transferred patient data.

The Support Vector Machine(SVM), Logistic Regression (LR), and Neural Network (NN) schemes are used to realize modeling attacks. The later is a 5-layer fully connected architecture including one input layer, three nonlinear hidden layers, and one output layer. The modeling accuracy is used as a metric to gauge the resilience of SWeeT against ML-assisted modeling attacks. We show the results for varying sizes of the training dataset (CRPs). We have also employed AVISPA to analyze the security of the underlying protocol.

B. Performance Result

Resistance against modeling attacks: SWeeT changes the key per session; if the patient's data varies it is not possible to model the key generation process using the encrypted data. However, if the same data is shared in multiple sessions subsequently, the adversary may be able to model the PUF behavior using ML techniques and extract the encryption keys. Owing to the nature of human bodies, such a case is probable. According, Fig. 5 shows the resiliency of SWeeT against the state of the art modeling attacks when the transferred data is constant. As depicted, the accuracy of such an attack in presence of SWeeT is 59.75%, 58%, and 58.25% when the adversary deploys LR, SVM, and NN, respectively, for the training size of 2000 intercepted payload. Even by increasing the training size to 200,000, the modeling accuracy does not exceed 63%. Due to the binary nature of the PUF responses, the baseline success to predict the response is 50%; thus 63% success is almost similar to guessing the response blindly. As shown, the accuracy of the modeling attack when SWeeT is not applied is over 97% with the training size of 2,000 CRPs.

Uniformity and Uniqueness of Encryption Key: Key uniformity reflects the distribution of zeros and ones in the key bits, and its favorable value is 50%. To assess the uniformity, we generated 200,000 random C and P values (each 64 bit) and extracted the corresponding SWeeT key for each case (we considered 2 physicians and 2 patients). The results exhibited the average uniformity of 50.53% for the generated keys.

To assess the uniqueness, we used the CRPs discussed above and extracted the average intra Hamming Distance (HD) between the keys generated by each patient P_i $(1 \leq i \leq 2)$. We also extracted the inter-HD between the keys used for transferring data between each patient P_i and physician Ph_j $(1 \leq i, j \leq 2)$. The average intra- and inter-HD are 46.8% and 50.5%, respectively; highly close to the ideal value of 50%.

Hardware and Power overhead: Implementing SWeeT (mainly two 64-bit Arbiter-PuFs, a simple controller and a 2-dimensional parity generator) on the Artix7 FPGA occupied 690 6-input LUTs. Compared to a wearable device, e.g., a smartwatch that can represent the PNG node, SWeeT imposes almost 0% hardware overhead. Although the embedded processing unit in a smartwatch is deemed to be lightweight, prototyping such a unit on an FPGA requires at least 100K LUTs. In our prototype implementation, the latency of encrypting each packet using SWeeT was negligible (around 3 us), and the power consumption was $\approx 2\ mW$. The smartwatch power consumption in wake-up mode is around 300mW [23]. In summary, SWeeT imposes very negligible overhead and thus fits the resource-constrained wearable devices.

Noise Impact: This set of results shows the observed error bits in the generated keys caused by the measurement noise and the temperature-change induced noise. For the former, the response of each PUF (per queried challenge) was measured twice at the room temperature while for the latter the PUF response per challenge is extracted one time at room temperature and another time in $100°C$. In both cases, we applied 2,000 challenges and the average bit error rates with and without the deployed error correction were tracked. The results are reported in Table I. As depicted the measurement noise (in the same temperature) highly diminishes after applying error correction, resulting in a reliable key in over 99.9% of the cases. The reliability was about 88% without error correction. Moreover, with the deployed correction scheme the temperature-change induced errors also decrease significantly, yielding correct keys in over 97.7% cases. Note that the temperature change we considered in this study is too pessimistic as the temperature of the wearable devices used by the patient does not go more than $43°C$ considering the normal temperature range of human bodies. Thanks to transferring the CRC of the original data to the physician, even in rare cases where the key is noisy (post ECC), the physician will be informed; thus wrong data will not get accepted and the physician sends another query to request the data again.

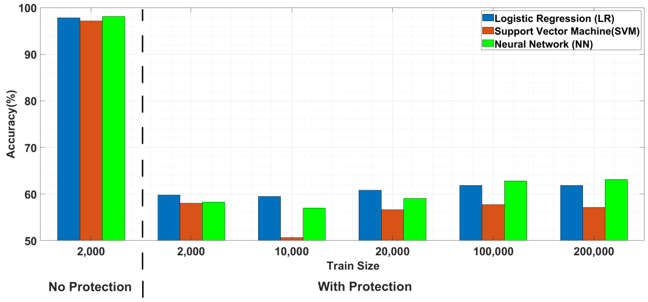


Figure 5: Accuracy of properly extracting the encryption key when different ML schemes are used to model the target PUFs in the presence/absence of SWeeT.

Table I: Percentage of error in the extracted session keys before and after applying the deployed error correction scheme. Errors are caused by the PUF measurement noise and temperature-induced changes of the PUF outputs. A 64-bit key is considered as erroneous even if 1 bit of the key has been changed.

Test Condition	ECC	1-bit error rate (%)	2-bit error rate (%)	>2-bit error rate (%)	Reliability (%)
measurement noise	Pre	11.29	0.72	0.02	87.97
	Post	0	0.1	0	**99.9**
temperature-Induced noise	Pre	31.0	9.35	3.25	56.41
	Post	0	1.9	0.4	**97.7**

C. Security Analysis:

Preventing Impersonation and Sybil Attacks: This attack is realized when an adversary tries to masquerade as one (or more) legitimate physicians to get access to the patient's data. In SWeeT the data is XORed with a key that is a function of the physician's ID, challenge bits and permutation index. Thus, as the experimental results also confirm, the utilized key and in turn the original (unencrypted) data cannot be uncovered via eavesdropping. Thereby, an adversary cannot impersonate a legitimate physician. Moreover, as SWeeT uses a fresh session key for each encryption, even if one session key is revealed, such a key is not valid for sending the following set of data.

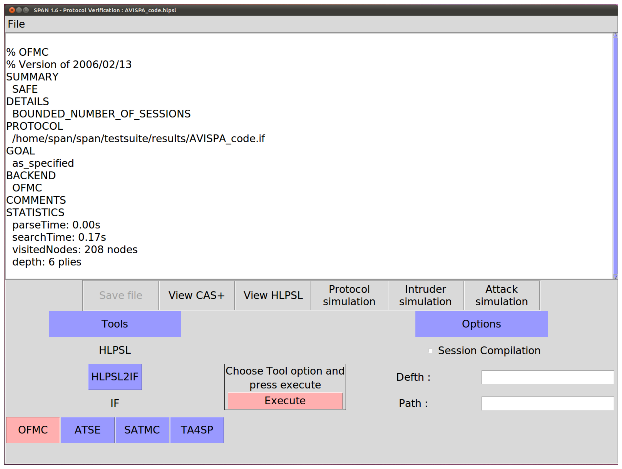
Resilience to modeling attacks: SWeeT employs two independent PUFs and derives a key that is based on the intertwined responses of these two PUFs to distinct challenges. Thus, in essence what an attacker will use to model SWeeT is a key representing the shuffled version of the two responses and challenges, i.e., C and ID. This is assuming that the attacker manages to successfully hack the physician PC; otherwise the attacker has access only to the challenges as well as the encrypted data. In both cases there is no correlation that could be modeled. Moreover, the shuffling pattern also depends on the PUF and varies per session. Thus in essence the adversary will have incoherent inputs and outputs.

Thwarting Data Manipulation Attempts: Two attacks fall under this category: (i) data forgery where wrong data is sent to the physician, and (ii) man-in-the-middle where a data packet is intercepted, data get modified and then the packet is retransmitted to the physician. Fundamentally the difference between these two attacks is whether the data is originally faulty or after being changed in transition to the recipient (physician). Since in SWeeT every session is initiated with a physician request, any unsolicited packet would be discarded. In addition, SWeeT prevents data modification by (a) encrypting the measurement data with a key that is generated in part by PUFs, (b) using a distinct key per session, and (c) appending a CRC of the data payload in the packet. Thus any well-orchestrated data modification requires uncovering the key in a timely manner, which is not possible given factorization of the PUF response for C, P and ID and the inability for modeling the PUF, as shown earlier. On the other hand, any random changes in the data will result in inconsistency between the decrypted data and the CRC of the data on the physician side.

Formal Security Validation: We have used AVISPA to validate the security properties of SWeeT. We have specified the following security goals that have to be fulfilled: i) the PUF response (key) is to be a secret and cannot be accidentally disclosed, ii) the secrecy of the patient's data is to be guaranteed and access to data is allowed only to the corresponding physician, i.e., even if a data packet is sent accidentally to an intended physician, the data should not be extracted from the packet payload, and iii) a unilateral authentication meaning that the physician authenticates the patient on the received enciphered data; in other words, when the physician deciphers and validates the patient's data, it is confirmed that the patient is actually present in the current session and sent such data. As shown in Fig. 6, AVISPA deems our scheme safe and shows that all the security goals are satisfied.

VI. Conclusion

Telehealth is being viewed as the predominant mode for medical services for quite a variety of illnesses. Wearable devices are an enabler for telehealth where the patient can be monitored remotely through a set of body-attached and implanted sensors. However, security and privacy issues have to be addressed before adopting these telehealth systems. Specifically, the patient's data should be protected against unauthorized access and manipulation during transmission, and the authenticity of the data source should be ensured. This paper presents SWeeT that tackles these issues. SWeeT takes advantage of hardware fingerprinting primitives, namely PUFs, to generate encryption keys on the fly. The keys are also personalized by factoring in the identity of the receiver in order to thwart impersonation, data forgery, and man-in-the-middle attacks. Each session has a distinct key to counter any attempts for modeling the underlying security provision. The effectiveness of SWeeT is validated through implementation and security analysis. The validation results not only demonstrated its attack resilience but also its very low overhead. Our future plan is to test SWeeT in a prototype wearable system.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/AVISPA_code.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.17s
  visitedNodes: 208 nodes
  depth: 6 plies
```

Defth :
Path :

Figure 6: Screenshot showing the result of OFMC module of AVISPA. The results confirm the robustness of SWeeT.

Acknowledgement

This work has been supported by the National Science Foundation MRI Award (1920079).

References

[1] S. B. Baker *et al.*, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Acc.*, vol. 5, pp. 26 521–44, 2017.

[2] M. N. Bhuiyan *et al.*, "Internet of things (iot): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities," *IEEE IoT J.*, vol. 8, no. 13, pp. 10 474–98, 2021.

[3] D. Halperin *et al.*, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Symp. on Security and Privacy*, 2008, pp. 129–142.

[4] A. Ghubaish *et al.*, "Recent advances in the internet-of-medical-things (iomt) systems security," *IEEE IoT J.*, vol. 8, no. 11, pp. 8707–18, 2021.

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

[6] S. Turner and T. Polk, "Prohibiting secure sockets layer (ssl) version 2.0," *Request for Comments*, vol. 6176, 2011.

[7] T. Belkhouja *et al.*, "Biometric-based authentication scheme for implantable medical devices during emergency situations," *Future Generation Computer Systems*, vol. 98, pp. 109–119, 2019.

[8] A. Kumari *et al.*, "Csef: cloud-based secure and efficient framework for smart medical system using ecc," *IEEE Acc.*, vol. 8, p. 107838, 2020.

[9] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM journal on computing*, vol. 32, no. 3, pp. 586–615, 2003.

[10] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the internet of things," in *Int'l Conf. on future networks and distributed systems*, 2017, pp. 1–5.

[11] K. Eldefrawy *et al.*, "Smart: Secure and minimal architecture for (establishing dynamic) root of trust." in *Ndss*, vol. 12, 2012, pp. 1–15.

[12] G. Dessouky *et al.*, "Litehax: lightweight hardware-assisted attestation of program execution," in *ICCAD*, 2018, pp. 1–8.

[13] M. Ebrahimabadi *et al.*, "Countering puf modeling attacks through adversarial machine learning," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2021, pp. 356–361.

[14] U. Chatterjee *et al.*, "Building PUF based authentication and key exchange protocol for iot without explicit crps in verifier database," *IEEE Trans. on Dep. & Sec. Comp.*, vol. 16, no. 3, pp. 424–437, 2019.

[15] M. N. Aman *et al.*, "Physical unclonable functions for IoT security," in *Int'l Workshop on IoT privacy, trust, and security*, 2016, pp. 10–13.

[16] S. Pirbhulal *et al.*, "A joint resource-aware and medical data security framework for wearable healthcare systems," *Future Generation Computer Systems*, vol. 95, pp. 382–391, 2019.

[17] R. U. Rasool *et al.*, "Security and privacy of internet of medical things: A contemporary review in the age of surveillance, botnets, and adversarial ml," *Journal of Network and Computer Applications*, p. 103332, 2022.

[18] 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, 2016.

[19] S. R. Moosavi *et al.*, "Low-latency approach for secure ecg feature based cryptographic key generation," *IEEE Access*, vol. 6, pp. 428–442, 2017.

[20] W. Lalouani *et al.*, "Robust and efficient data security solution for pervasive data sharing in IoT," in *CCNC*, 2022, pp. 775–781.

[21] M. Ebrahimabadi et al., "A PUF-based modeling-attack resilient authentication protocol for IoT devices," *IoT J.*, vol. 9, pp. 3684–3703, 2022.

[22] H. Kamabe et al., "Integrated interleaving ecc and high dimensional parity codes," in *Int'l Magnetics (INTERMAG)*, 2005, pp. 993–994.

[23] X. Liu *et al.*, "Characterizing smartwatch usage in the wild," in *Proc. the 15th Int'l Conf. on mobile sys., app., and services*, 2017, pp. 385–398.