POLITECNICO DI TORINO
Repository ISTITUZIONALE

Formal Verification and Mitigation of Vulnerabilities in the IMDGuard Protocol

Original
Formal Verification and Mitigation of Vulnerabilities in the IMDGuard Protocol / Coduri, C., Sacco, A., Marchetto, G., Sisto, R.. - (2026). (The 24th International Conference on Pervasive Computing and Communications (PerCom 2026) Pisa (IT) 16-20 March 2026) [10.1109/PerComWorkshops68308.2026.11585400].

Availability:
This version is available at: 11583/3009062 since: 2026-07-11T16:20:58Z

Publisher:
IEEE

Published
DOI:10.1109/PerComWorkshops68308.2026.11585400

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

10 October 2026

Formal Verification and Mitigation of Vulnerabilities in the IMDGuard Protocol

Christian Coduri, Alessio Sacco, Guido Marchetto, Riccardo Sisto
Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy.
Emails: alessio_sacco@polito.it, {first.last}@polito.it

***Abstract*—IMDGuard is a proxy-based authentication protocol designed to secure communications between an Implantable Medical Device (IMD), a technology increasingly used in cardiac care, and its associated Programmer device through a wearable third-party mediator known as the Guardian. In this paper, we formally analyze the security of IMDGuard using the ProVerif verification tool and demonstrate that the protocol is vulnerable to multiple attacks. Our analysis confirms a known susceptibility to denial-of-service attacks and reveals a previously unreported flaw in the authentication mechanism between the Guardian and the Programmer. This vulnerability allows an adversary to impersonate the IMD, inject falsified telemetry data, and issue deceptive responses to legitimate Programmer commands, posing a serious risk to patient safety by potentially leading clinicians to make medical decisions based on compromised information. To mitigate this threat, we propose a remediation mechanism that enforces the mutual authentication process, thereby enhancing protocol resilience against impersonation attacks. The proposed improvements restore the intended security guarantees of IMDGuard, ensuring safer and more reliable communication in cardiac care applications.**

***Index Terms*—Implantable Medical Devices, Cardiac Devices, IMD Security, Formal Verification, ProVerif**

I. Introduction

Implantable Medical Devices (IMDs), such as pacemakers, implantable cardioverter defibrillators (ICDs), neurostimulators, and implantable drug delivery systems, incorporate miniaturized embedded computing systems that autonomously perform critical health monitoring and therapeutic functions. These devices have become indispensable components of modern healthcare, enabling continuous, real-time physiological monitoring and delivering timely, targeted medical interventions [1]. For cardiac implantable devices, such as pacemakers and ICDs, medical personnel use a *programmer* (a specialized device provided by the manufacturer) to interact with the device (e.g., to retrieve diagnostic data or review historical activity logs) during periodic visits. In addition, many patients also participate in *telemonitoring* programs. In this setup, a home-based monitoring system periodically communicates with the IMD to collect operational and physiological data, which are then transmitted to the hospital for remote analysis [2].

Due to their limited resources, IMDs cannot always support energy-intensive cryptographic or signal-processing modules; consequently, many devices lack robust authentication mechanisms, leaving patients exposed to cyberattacks. Moreover, in emergency situations, medical personnel may need rapid access to the IMD to retrieve data or modify therapy parameters while the patient is unable to provide the required credentials (e.g., due to unconsciousness). In such scenarios, strict authentication and access-control mechanisms may be impractical [3]. Therefore, two access modalities are required: a method for normal operation and an emergency mode that temporarily bypasses security to permit what would otherwise be considered unauthorized access.

To secure communication with IMDs, various authentication and access control protocols have been proposed [4], operating either in both normal and emergency modes or exclusively in normal mode, and relying on proximity, biometrics, or proxy-based mechanisms. Proxy-based approaches are particularly promising for real-world deployment, as they offload computationally intensive cryptographic operations to an external device, keeping the IMD lightweight. Within this category, IMDGuard [5] is one of the most frequently referenced protocols, despite being vulnerable to denial-of-service (DoS) attacks and having a weakness in the generation of one of its three cryptographic keys.

Our contribution. In this work, we conduct a comprehensive formal security analysis of IMDGuard using ProVerif, one of the most widely adopted symbolic verification tools [6] [7]. Our verification revealed several potential DoS attacks as well as a critical flaw in the authentication mechanism used in regular mode, which is intended to protect non-emergency operations. This previously unreported vulnerability has been inherited by subsequent works in the literature without being corrected, allowing an adversary to inject falsified telemetry data in responses to legitimate Programmer commands. Although it does not directly harm the patient, it could indirectly do so by leading a physician to make an incorrect clinical decision. We therefore propose a simple modification that enforces mutual authentication between the Guardian and the Programmer, thereby protecting the protocol from impersonation attacks and safeguarding the Protected Health Information (PHI) sent by the IMD.

Structure of the paper. Section II presents a summary of related work. Section III provides a detailed description of the IMDGuard protocol under verification. Section IV explains how the protocol was modeled in ProVerif, and Section V presents the verified security properties and the corresponding results. Finally, Section VI describes our proposal to fix the protocol, and Section VII concludes the paper.

II. Related Work

The security of IMDs today is a critical concern as these systems become more interconnected. Although no cyberattacks causing direct patient harm have yet been reported, researchers have demonstrated that the threat is real and significant, and the entire IMD ecosystem is a potential target [8]. IMDs themselves can be targeted by replay attacks conducted via radio-based techniques to alter or disclose patient information, modify therapy settings, or even disable treatment delivery [9]. Programmers and home monitoring systems are also susceptible, as demonstrated in [10], which revealed poor security practices and multiple vulnerabilities among the four major implantable cardiac device vendors.

To mitigate some of these risks, various access control and authentication protocols have been proposed, categorized into four main approaches: proximity-based, proxy-based, biometrics-based, and hybrid methods [4].

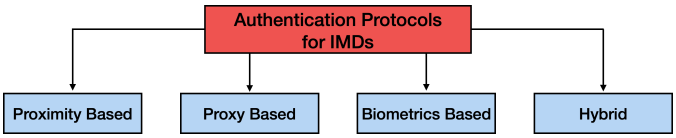

Figure 1. Programmer-to-IMD Authentication and Access Control Schemes

Proximity-based schemes grant IMD access only when the external programmer is sufficiently close to the device, activating emergency access once the distance falls below a defined threshold [11]. *Biometric-based* systems can operate in both regular and emergency modes, but in practice they often function as emergency backdoor mechanisms. These approaches rely on physical or physiological traits, such as fingerprints, iris patterns, or ECG-derived features, to enforce access control, derive cryptographic keys, or support secret-sharing schemes [12]–[16]. *Proxy-based* approach relies on an external security device that patients wear on a daily basis. Acting as an intermediary, it ensures that only authorized programmers can communicate with the IMD, while emergency access remains possible by simply removing the device. Two main architectures implement this concept: IMD-Shield [17], which uses jamming to protect communication, and IMDGuard [5], which employs three different keys among the involved parties. Zhang et al. proposed another valid alternative called MedMon [18], a monitoring device that inspects all IMD communications for malicious activity and alerts patients to critical events. Finally, *hybrid* mechanisms combine multiple authentication strategies to balance security, usability, and emergency accessibility.

IMDGuard, the protocol analyzed in this work, relies on the assumption that an adversary cannot obtain a patient's ECG signal without physical contact, as this parameter is used to derive one of its cryptographic keys. However, prior research has demonstrated that ECG data can, in fact, be captured remotely [19]–[21]. Moreover, Rostami et al. [22] demonstrated that a man-in-the-middle attack can reduce the effective entropy of the same ECG-derived key from 128 bits to 86, further compromising security.

While these studies focus on cryptographic weaknesses in IMDGuard, they overlook vulnerabilities arising from protocol design. Formal analyses of IMD authentication schemes [23] revealed a general lack of rigorous verification during development. Ideally, every proposed protocol should include formal validation, as done by Camara et al. [24], who developed ACIM, a proximity and identity-based access control protocol, and verified its security using ProVerif.

III. IMDGuard Protocol

The IMDGuard Protocol is a notable example of a proxy-based authentication system designed for secure cardiac IMD communication [5]. The components involved in this protocol include the IMD, the Programmer, and an external security device called Guardian, as illustrated in Figure 2.

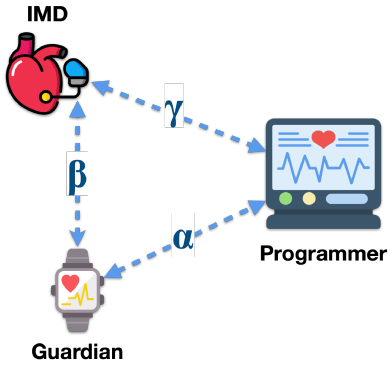

Figure 2. IMD Guardian Architecture

Under normal conditions, the Guardian (G) functions as a controllable ON/OFF switch, regulating secure access to the IMD by the Programmer (P). In contrast, during emergency situations, when the Guardian is removed, the IMD automatically enters an emergency mode, permitting unsecured access to prioritize patient safety.

A. Communication Channels and Key Management

The architecture involves three main logical communication links, all serving distinct purposes within the protocol and having their own specific key management procedures:

- **Link Alpha** ($\alpha : G \Leftrightarrow P$): used for access control. The Guardian and the Programmer each maintain a list of the other's legitimate public keys, which are securely *pre-installed* during deployment in a clinical setting.
- **Link Beta** ($\beta : G \Leftrightarrow IMD$): used for initial pairing. As an external device, the Guardian must establish a shared secret key with the IMD without any prior association. Pre-deploying a key is impractical because resetting the IMD is invasive; instead, the system derives the *shared key from the patient's ECG signals*.
- **Link Gamma** ($\gamma : IMD \Leftrightarrow P$): used to send commands, extract data, or reprogram specific therapy configurations.

This channel is secured using a *session key* generated and distributed by the Guardian to both the IMD and the Programmer, enabling secure communication.

B. Protocol Workflow

In both normal and emergency modes, the interaction begins when the Programmer (P) initiates an access request to the IMD. Upon receiving this request, the IMD replies with its identifier and a randomly generated nonce (`n1_IMD`), starts a timer ($T1$), and waits for a response from the Guardian (G).

1) Regular Mode: During the timer interval, G authenticates P using a simple challenge-response mechanism. Specifically, G sends a random nonce to P, which responds with its own nonce and the received one signed with its private key. Since G already holds the Programmer's public key, it can verify the signature and authenticate P.

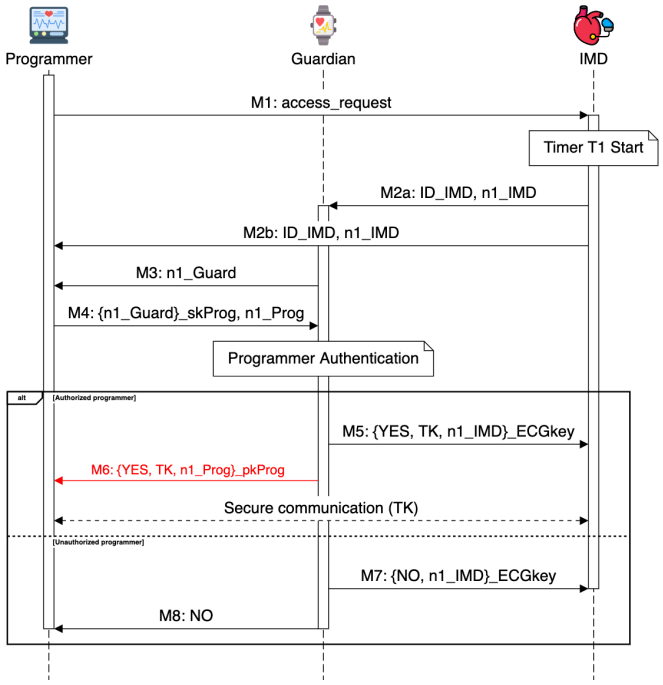

Figure 3. Regular Mode: Sequence Diagram

If authentication succeeds before the $T1$ timer expires, the system transitions to regular operation mode. The Guardian subsequently distributes a temporary key (`TK`) to both the IMD and the Programmer, thereby establishing the secure communication channel γ. Conversely, if authentication fails, the Guardian transmits a denial message to both entities.

2) Emergency Mode: If the Guardian is unavailable, after three consecutive timer expirations, the system switches to emergency mode. In emergency mode, IMDGuard assumes no adversary is present, leaving communication between the Programmer and IMD unencrypted and exposed to interception or modification. To mitigate jamming attacks that could falsely trigger this mode, the Guardian employs a proper defensive mechanism (see [5] for details).

As no security measures are active in emergency mode, our formal verification focuses exclusively on the regular (secure) mode.

IV. Modeling the Protocol with ProVerif

We model IMDGuard using ProVerif, an automated tool for the formal verification of security protocols, based on symbolic models. This framework allows us to represent communicating processes and analyze the protocol to detect potential flaws or confirm the robustness in the presence of a powerful and active adversary. Because ProVerif assumes perfect cryptography, modeling cryptographic primitives as ideal and unbreakable black boxes, attacks exploiting algorithmic or mathematical weaknesses are outside the scope of the verification [6] [7].

In our model, asymmetric encryption, symmetric encryption, and digital signatures are encoded using standard symbolic representations commonly adopted in the literature and documented in the ProVerif manual [7].

A. Threat Model

ProVerif follows the Dolev-Yao model, where the adversary can eavesdrop on all exchanged messages, decrypt or encrypt messages when possessing the necessary keys, and freely replay, delete, modify, forge, or inject messages using the knowledge acquired during protocol execution. In our formalization, all communications occur over wireless public channels, fully exposed to the adversary, thereby maximizing the range of potential malicious interactions within the protocol.

We assume the adversary knows the public keys of G and P, as well as the formats of the initialization access string (sent by P in message $M1$ of Figure 3) and of the `YES` (used in $M5$, $M6$) and `NO` (used in $M8$) terms. Private and shared keys are treated as secret, whereas nonces and other terms are considered secret only until they are transmitted in cleartext or encrypted with a key that the adversary can obtain. Based on this model, we evaluate whether the attacker can perform the following attack categories:

- **Eavesdropping Attack**, targeting the *confidentiality* of exchanged data. The adversary passively monitors public communication channels to obtain sensitive or private information that should remain secret.
- **Impersonation and Message Manipulation Attack**, targeting the *authenticity* and *integrity* of communications. By altering protocol messages, the attacker can impersonate legitimate entities or modify exchanged data, causing devices to accept falsified information as authentic.
- **Denial-of-Service (DoS) Attack**, targeting the *availability* of the protocol and device access. The adversary may forge malicious message exchanges to prevent the protocol from completing successfully or to deny legitimate access requests.

B. Protocol Processes

After defining the cryptographic operations, communication channels, and the attacker's initial knowledge, we specify a main process that instantiates each protocol participant.

We instantiate single (non-replicated) processes because our analysis targets a single execution of the protocol; replication would introduce unnecessary concurrent sessions not relevant to IMDGuard's one-to-one interaction model.

```
process
  (* G and P each have a private-public key pair *)
  new kpProg:keymat;
  new kpGuard:keymat;

  (* Assume P and the IMD already share a key *)
  new ECG_based_key:key;

  (
    (* Give the public keys to the attacker *)
    out(ch_alpha, pk(kpProg)); 0  |
    out(ch_alpha, pk(kpGuard)); 0 |

    (* Instantiate all processes *)
    IMD(ECG_based_key) |
    Programmer(kpProg, pk(kpGuard)) |
    Guardian(kpGuard, ECG_based_key, pk(kpProg))
  )
```

Then, we model each role (the Programmer, the IMD, and the Guardian) as a distinct process and specify their interactions, including the creation, transmission, reception, and manipulation of values. The modeled interaction includes the entire authentication procedure illustrated in Figure 3. Furthermore, once `TK` has been distributed to both the IMD and the Programmer, we model the first encrypted interaction over channel γ. In this phase, the Programmer sends a data retrieval request (`retr_command`), to which the IMD responds with `resp_phi`, containing the relevant Protected Health Information (PHI) it stores.

V. Security Properties and Verification Results

Finally, we define a set of security properties that enable ProVerif to systematically explore all possible interactions between honest participants and the adversary in search of potential attack traces. This section presents these properties, explains how they are modeled, and discusses the results.

A. Confidentiality of Sensitive Data

A fundamental security goal is to ensure that adversaries cannot gain access to the participants' secrets, even when actively engaging in the protocol. In our model, the primary confidential element is `resp_phi`, which carries the PHI sent by the IMD to the Programmer over channel γ. In ProVerif, confidentiality is represented as a reachability property: a secret is considered confidential if the adversary cannot reach (i.e., derive) the corresponding term using all the knowledge and capabilities acquired. This can be easily tested as follows.

```
query attacker(resp_phi).
```

The result of this query shows that the adversary cannot derive `resp_phi`, providing formal evidence that the protocol maintains the confidentiality of the sensitive information.

B. Authenticity and Integrity of Sensitive Data

To ensure that the sensitive data retrieved from the IMD (`resp_phi`) is also authentic and unaltered, we define a correspondence assertion that verifies each PHI value received by the Programmer corresponds to a unique transmission from the IMD, thereby enforcing a strict one-to-one relationship between receipt and legitimate origin.

```
query data:phi_data, tk:key;
  inj-event(phi_received(data, tk))
    ==> inj-event(phi_sent(data, tk)).
```

The query result reveals a breach of the intended authenticity property, indicating that *the attacker can deceive the Programmer into accepting data that the IMD never transmitted.*

1) Attack Analysis: The observed violation arises from the lack of mutual authentication: although the Guardian authenticates the Programmer upon receiving message $M4$ (Figure 3), the Programmer does not cryptographically verify the origin of message $M6$, leaving no assurance that it was generated by a legitimate Guardian. An attacker can exploit this weakness by first impersonating the Guardian and installing a forged `TK` on the Programmer. Using this falsified key, the attacker can then impersonate the IMD and transmit counterfeit data to P, compromising both authenticity and integrity.

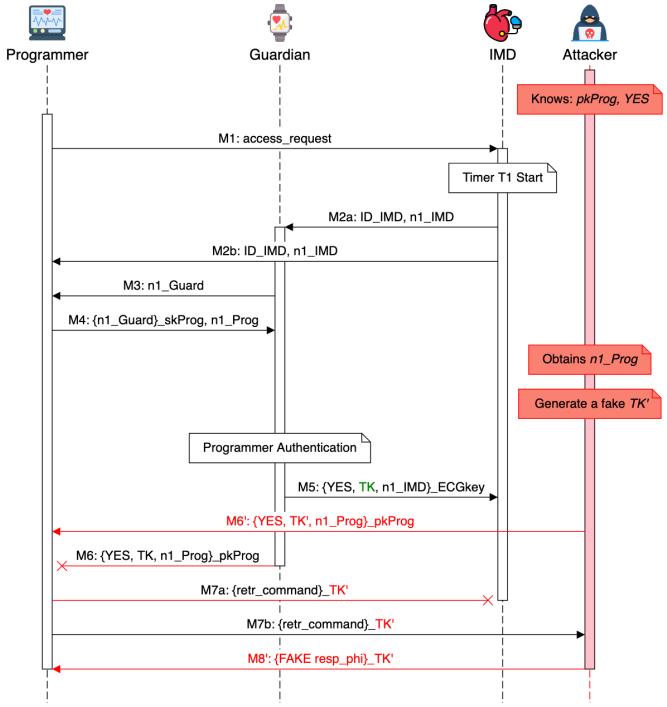

Figure 4. Attack Trace: an adversary exploits the flawed authentication to install a forged session key and deliver falsified PHI to the Programmer

This attack, illustrated in Figure 4, consists of a **Message Forgery and Impersonation Attack** and proceeds as follows:

1) The attacker A intercepts message $M4$ to extract the nonce `n1_prog`.

2) A, *impersonating the Guardian*, forges multiple instances of $M6'$ by leveraging the intercepted nonce and the known format of the `YES` message, inserting a fraudulent temporary key `TK'`. To prevent the attack from failing if the legitimate `TK` is received first, A floods the Programmer with a large number of forged $M6'$ messages, all encrypted with the known `pk_Prog`.
3) The Programmer, unable to verify the origin of the message, accepts the forged key and subsequently issues a `retr_command` encrypted with `TK'`.
4) The adversary, now *impersonating the IMD*, can decrypt the Programmer's command (encrypted with `TK'`) and respond with falsified protected health information ($M8'$), while the legitimate IMD is unable to decrypt the command.

2) Real-World Scenario: Using the same type of query applied to the received and sent PHI, we also tested the authenticity and integrity of the `retr_command`. No violations were found, indicating that an attacker cannot impersonate the Programmer and inject unauthorized commands into the IMD.

Nevertheless, the manipulation of medical data sent to the Programmer, as illustrated in Figure 4, remains a serious security risk. Forged data can lead to misdiagnosis, inappropriate treatment, or delayed intervention, compromising patient safety and potentially resulting in regulatory, legal, and reputational consequences for healthcare institutions involved.

C. Availability and Access Control

The final part of our evaluation focuses on the protocol's availability and access control mechanisms. Our goal is to determine whether an adversary can trigger a sequence of messages that either forces the protocol into a denial state or causes an incorrect progression through the access-control logic. To this end, we define two types of events that capture authorization and denial outcomes for each device, enabling us to formally reason about whether the protocol can be driven into unintended or inconsistent states.

1) Denial-of-Service in Session Setup: In our model, denial events should never be triggered during correct protocol execution. By querying their reachability in ProVerif, we can identify potential attack traces and demonstrate that the protocol's availability can be compromised not only through radio-based attacks, such as jamming, but also through manipulations of the protocol messages themselves.

```
query event(guardian_denies_comm()).
query event(programmer_access_denied()).
query event(imd_access_denied()).
```

Three traces in which denial events occur were detected:

- **Guardian denies communication:** A tampered signature of `n1_Guard` in $M4$, sent in response to the challenge, causes the Guardian to reject the session.
- **Programmer access denied:** Forged or modified responses, such as a `NO` message ($M8$) or an incorrect `n1_Prog` in $M6$, lead the Programmer to deny access.
- **IMD access denied:** Triggered indirectly when the Guardian refuses communication, causing the IMD to also reject the Programmer.

2) Authorization Flow: In contrast, `allow` events should always be triggered during correct protocol execution. By defining two correspondence assertions, we ensure that the Programmer and the IMD can reach the `access_allowed` "state" only if the Guardian has explicitly granted authorization by triggering the corresponding event beforehand.

```
query x:pkey, y:pkey, tk:key;
  inj-event(programmer_access_allowed(x, y, tk))
    ==> inj-event(guardian_allows_comm(x, y, tk)).

query x:pkey, y:pkey, tk:key;
  inj-event(imd_access_allowed(tk))
    ==> inj-event(guardian_allows_comm(x, y, tk)).
```

When verified, ProVerif confirmed the second assertion, showing that the IMD accepts communication only after explicit Guardian authorization. The first assertion, however, failed, revealing a subcase of the previously described attack (Figure 4): when the Programmer receives the forged `YES` message in $M6'$, it triggers the `access_allowed` event even though the message did not originate from the legitimate Guardian and no authorization event was issued.

VI. Fixing the Protocol

A summary of the main security issues identified by ProVerif in the IMDGuard protocol is presented in Table I. To address most of these issues, it is essential to ensure that the temporary key is transmitted by the legitimate Guardian, thereby guaranteeing mutual authentication. Applying this correction preserves the integrity and authenticity of the PHI transmitted by the IMD and prevents impersonation attacks. The updated sequence diagram incorporating these fixes is shown in Figure 5.

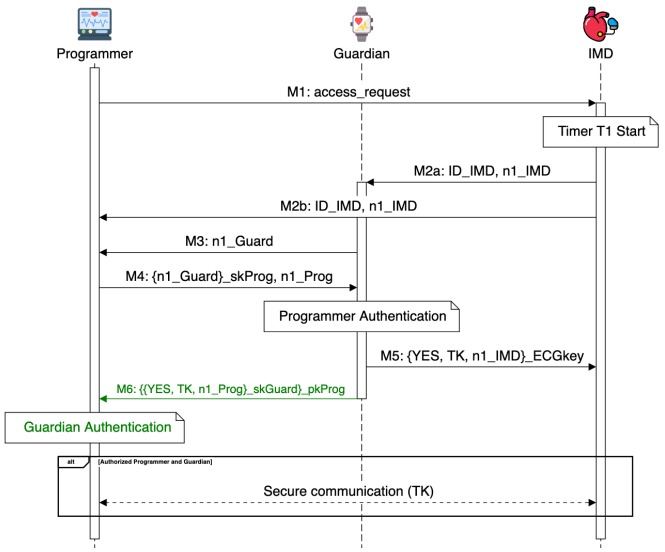

Figure 5. Fixed Regular Mode: Sequence Diagram

Table I
COMPARISON OF SECURITY PROPERTIES BETWEEN THE ORIGINAL IMDGUARD PROTOCOL AND THE PROPOSED FIX

Protocol Version	**Mutual Authentication**		**PHI Security Properties**			**Attack Resistance**	
	G Authenticates P	P Authenticates G	Confidentiality	Integrity	Authenticity	Impersonation	DoS
IMDGuard [5]	✓	✗	✓	✗	✗	✗	✗
Our Solution	✓	✓	✓	✓	✓	✓	✗

To introduce mutual authentication, instead of directly sending message $M6$ (as shown in Figure 3), the Guardian now signs it with its private key and then encrypts it with the Programmer's public key before transmission. On the receiving side, the Programmer, being the only entity able to decrypt the message with its private key, can then ensure that the message is coming from the Guardian by verifying the signature, and then it proceeds with the same protocol checks: confirming that it has authorization by checking if the first field is `YES` and finally that the nonce `n1_prog` matches the one it previously sent, thereby also ensuring freshness and preventing replay attacks. After implementing these corrections and re-running ProVerif, the attack trace shown in Figure 4 is no longer produced. Table I provides a final comparison between the original and corrected versions of the protocol.

VII. CONCLUSION

In this paper, we present a formal verification of IMDGuard, a well-known proxy-based protocol, through which we identify multiple vulnerabilities. Our verification revealed a previously unreported flaw in its regular (secure) mode that enables an attacker to impersonate the IMD and inject falsified data into the Programmer. To address this issue, we propose a simple modification that preserves the original protocol structure, preventing impersonation attacks and ensuring the authenticity and integrity of the data delivered to the Programmer.

REFERENCES

[1] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 30–39, 2008.

[2] G. Zheng, R. Shankaran, M. A. Orgun, L. Qiao, and K. Saleem, "Ideas and challenges for securing wireless implantable medical devices: A review," *IEEE Sensors Journal*, vol. 17, no. 3, pp. 562–576, 2017.

[3] N. Ellouze, M. Allouche, H. B. Ahmed, S. Rekhis, and N. Boudriga, "Security of implantable medical devices: limits, requirements, and proposals," *Security and Communication Networks*, vol. 7, no. 12, pp. 2475–2491, 2014.

[4] S. Challa, M. Wazid, A. K. Das, and M. K. Khan, "Authentication protocols for implantable medical devices: Taxonomy, analysis and future directions," *IEEE Consumer Electronics Magazine*, vol. 7, no. 1, pp. 57–65, 2017.

[5] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li, "Imdguard: Securing implantable medical devices with the external wearable guardian," in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 1862–1870.

[6] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, 2001, pp. 82–96.

[7] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial," *Version from*, vol. 16, pp. 05–16, 2018.

[8] S. Das, G. P. Siroky, S. Lee, D. Mehta, and R. Suri, "Cybersecurity: The need for data and patient safety with cardiac implantable electronic devices," *Heart rhythm*, vol. 18, no. 3, pp. 473–481, 2021.

[9] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 129–142.

[10] B. Rios and J. Butts, "Security evaluation of the implantable cardiac device ecosystem architecture and implementation interdependencies," *WhiteScope, sl*, 2017.

[11] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 410–419.

[12] G. Zheng, W. Yang, C. Valli, L. Qiao, R. Shankaran, M. A. Orgun, and S. C. Mukhopadhyay, "Finger-to-heart (f2h): Authentication for wireless implantable medical devices," *IEEE journal of biomedical and health informatics*, vol. 23, no. 4, pp. 1546–1557, 2018.

[13] X. Hei and X. Du, "Biometric-based two-level secure access control for implantable medical devices during emergencies," in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 346–350.

[14] M. Rostami, A. Juels, and F. Koushanfar, "Heart-to-heart (h2h): authentication for implanted medical devices," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 1099–1112.

[15] G. Zheng, G. Fang, M. A. Orgun, R. Shankaran, and E. Dutkiewicz, "Securing wireless medical implants using an ecg-based secret data sharing scheme," in *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, 2014, pp. 373–377.

[16] T. Belkhouja, X. Du, A. Mohamed, A. K. Al-Ali, and M. Guizani, "Biometric-based authentication scheme for implantable medical devices during emergency situations," *Future Generation Computer Systems*, vol. 98, pp. 109–119, 2019.

[17] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu, "They can hear your heartbeats: Non-invasive security for implantable medical devices," in *Proceedings of the ACM SIGCOMM 2011 conference*, 2011, pp. 2–13.

[18] M. Zhang, A. Raghunathan, and N. K. Jha, "Medmon: Securing medical devices through wireless monitoring and anomaly detection," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 6, pp. 871–881, 2013.

[19] R. M. Seepers, W. Wang, G. de Haan, I. Sourdis, and C. Strydis, "Attacks on heartbeat-based security using remote photoplethysmography," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 3, pp. 714–721, 2018.

[20] M.-Z. Poh, D. J. McDuff, and R. W. Picard, "Advancements in noncontact, multiparameter physiological measurements using a webcam," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 7–11, 2011.

[21] A. Calleja, P. Peris-Lopez, and J. E. Tapiador, "Electrical heart signals can be monitored from the moon: Security implications for ipi-based protocols," in *Information Security Theory and Practice*, R. N. Akram and S. Jajodia, Eds. Cham: Springer International Publishing, 2015, pp. 36–51.

[22] M. Rostami, W. Burleson, F. Koushanfar, and A. Juels, "Balancing security and utility in medical devices?" in *Proceedings of the 50th annual design automation conference*, 2013, pp. 1–6.

[23] D. G. Duguma, I. You, Y. E. Gebremariam, and J. Kim, "Can formal security verification really be optional? scrutinizing the security of imd authentication protocols," *Sensors*, vol. 21, no. 24, p. 8383, 2021.

[24] C. Camara, P. Peris-Lopez, J. M. De Fuentes, and S. Marchal, "Access control for implantable medical devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1126–1138, 2020.