SecuWear: Secure Data Sharing Between Wearable Devices

Sujin Han
KAIST
sujinhan@kaist.ac.kr

Diana A. Vasile, Fahim Kawsar, Chulhong Min
Nokia Bell Labs
{diana-alexandra.vasile, fahim.kawsar, chulhong.min}@nokia-bell-labs.com

***Abstract*—Wearable devices, often used in healthcare and wellness, collect personal health data via sensors and share it with nearby devices for processing. Considering that healthcare decisions may be based on the collected data, ensuring the privacy and security of data sharing is critical. As the hardware and abilities of these wearable devices evolve, we observe a shift in perspectives: they will no longer be mere data collectors, rather they become empowered to collaborate and provide users with enhanced insights directly from their bodies with on-device privacy-enhanced processing. However, today's data sharing protocols do not support secure data sharing directly between wearables.**

To this end, we develop a comprehensive threat model for such scenarios and propose a protocol, SecuWear, for secure real-time data sharing between wearable devices. It enables secure data sharing between any set of devices owned by a user by authenticating devices with the help of an orchestrator device. This orchestrator, one of the user's devices, enforces access control policies and verifies the authenticity of public keys. Once authenticated, the data encryption key is directly shared between the data provider and data consumer devices. Furthermore, SecuWear enables multiple data consumers to subscribe to one data provider, enabling efficient and scalable data sharing. In evaluation, we conduct an informal security analysis to demonstrate the robustness of SecuWear and the resource overhead. It imposes latency overhead of approximately 1.7s for setting up a data sharing session, which is less than 0.2% for a session lasting 15 minutes.

I. Introduction

Wearable devices have become an integral part of daily life, blending seamlessly into routines from morning to night. Ranging from fitness trackers to smartwatches, these devices track physical activity, monitor sleep patterns, and provide real-time health metrics like heart rate and calorie expenditures. Beyond fitness, wearables support continuous health monitoring, offering insights into vital signs and aiding in chronic condition management [1], [2]. Their ubiquitous presence highlights their role as both personal wellness tools and critical components in healthcare.

The wearable landscape is evolving. Wearable devices are developed in various forms, such as rings [3] and earbuds [4], and the number of wearable devices owned per person is growing. In these multi-wearable environments, combining data from multiple devices can enrich services [5], [6], [7] and boost inference accuracy [8], [9], as each device captures unique aspects of the human body. The integration of tiny AI accelerators (e.g., Analog MAX78000 [10], Google Coral Micro [11]) has enhanced computational capabilities, allowing fully local AI execution without depending on a smartphone's processing power [5], [12], [13] Thus, applications will soon run directly on wearables, using data collected from multiple devices across the user's body. For example, as shown in Figure 1, a user may have an activity detection application running on a smartwatch and a fatigue detection application running on smart earbuds, both of which rely on data collected from multiple user devices.

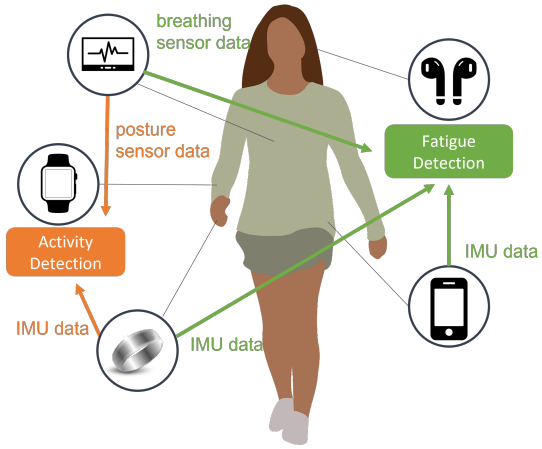

Fig. 1. Data sharing between wearable devices.

Given the sensitive nature of the data collected by wearable devices, there is a critical need for secure data sharing protocols between wearable devices. Without robust security measures, data could be intercepted, altered, or accessed by unauthorised parties, leading to privacy breaches [14], [15], data tampering [16], or even identity theft [17]. Furthermore, data injection attacks on health monitoring devices, like glucose monitors, can critically impact users by influencing medication decisions such as insulin dosage [16].

Unfortunately, existing methods for data sharing between wearable devices are not suitable in scenarios where these devices require data from other devices. Currently, Samsung's

Workshop on Security and Privacy in Standardized IoT (SDIoTSec) 2025
24 February 2025, San Diego, CA, USA
ISBN 979-8-9919276-6-6
https://dx.doi.org/10.14722/sdiotsec.2025.23xxx
www.ndss-symposium.org

Health SDK [18] and Apple's HealthKit [19] accumulate data from wearable devices on a smartphone and mobile applications make requests to the smartphone to receive necessary data. This design introduces two critical limitations. First, from a security perspective, it introduces a single point-of-failure where a compromised smartphone not only renders all applications non-usable but also potentially enables an attacker to gain access to user data collected from all wearable devices. Second, from an application's perspective, such smartphone-relayed data sharing incurs significant overhead in terms of energy consumption and latency when remote data needs to be processed on wearable devices.

To support more secure and sustainable data sharing between wearable devices, we design a comprehensive threat model and propose *SecuWear*, a protocol for secure data sharing between wearable devices. In this work, we consider all devices that a user possesses when carrying out daily activities (ex. smartphone, smartwatch, smart ring, smart earbuds) to be wearable devices. In SecuWear's architecture, we introduce the concept of an *orchestrator device*, the most computationally capable device in the set of available user devices and responsible for setting up a session for direct data sharing between data provider devices and data consumer devices. With SecuWear, user data is securely and directly shared with devices that need the data and not with the orchestrator device. Moreover, the role of the orchestrator can be handed over to another device in case the original orchestrator device runs out of battery or becomes compromised.

The key difference of SecuWear from existing data sharing protocols is that it decouples trust establishment from data exchange. A central device, the orchestrator, acts as a trusted third party when setting up a data sharing session. Specifically, the public keys of each device involved in the data sharing are shared through the trusted orchestrator device, guaranteeing authenticity. Within a data sharing session, the data provider device directly sends the data to the data consumer device, eliminating a central repository for all user data.

Through an informal security analysis, we demonstrate that SecuWear provides strong security guarantees against an active local adversary. We also implement a prototype and show that SecuWear imposes minimal system overhead suitable for resource-constrained wearable devices. When evaluated with Raspberry Pi 4 and Raspberry Pi Zero devices, setting up a data sharing session takes 1.7s on average, which is less than 0.2% latency overhead for a 15-minute data sharing session.

In summary, the paper makes the following contributions:

1. We develop a comprehensive threat model for data sharing framework for wearables communicating sensitive data.
2. We design and implement a prototype of SecuWear, which provides secure and efficient data sharing for wearable devices by decoupling trust establishment from data sharing.
3. We conduct an informal security analysis of SecuWear to outline the security guarantees it provides and when it fails.
4. We also conduct a system overhead evaluation to demonstrate that SecuWear imposes acceptable overhead to be integrated into resource-constrained wearable devices.

II. MOTIVATION

Collaboration between wearables is becoming a reality with the advancement of tiny AI accelerators enabling fully local execution of applications on wearable devices [5], [12]. This move is prompting research into new security protocols to facilitate collaboration in a secure and private way, such as sandboxing approaches [13], user identification or authentication approaches for devices without user interface [20]. Our goal is to develop a secure data sharing protocol for collaboration between wearable devices.

The target scenario involves multiple devices with varying computational power, typically including a smartphone and several wearable devices with limited processing capabilities. Each device is equipped with multiple sensors to collect user data. Each device can run various applications fully locally and those applications may use local data and also request data from other devices owned by the same user. Therefore, these devices must wirelessly share data while meeting strict latency requirements for real-time processing. However, accumulating all collected data in a single device is unnecessary.

A naive way to support secure data sharing in this scenario would be establishing secure one-to-one communication each time data sharing occurs. However, such an approach has two limitations. First, an authentication mechanism is necessary to establish trust between devices, which may require a trusted third party or additional user involvement. Second, duplicate data streams (i.e., identical data transmitted from the same device) significantly reduce efficiency.

Existing data sharing methods often resolve the above concerns by accumulating user data in a centralized storage, such as a smartphone [18], [19] or cloud storage [21], [22]. However, such "data hub" approaches introduce significant security risks. A compromised smartphone can halt collaboration and potentially leak data from all connected devices, and storing sensitive data in cloud storage is not ideal in terms of privacy. Moreover, some applications enable direct data sharing between user devices running the same application [23], but this approach is also limited. It relies on the application developer's ability to implement secure communication and can incur duplicate data streams. A more detailed analysis is provided in the appendix.

III. RELATED WORK

Numerous works suggest secure data sharing protocols in various scenarios [24], [25], [26], [27]. However, only a handful focus on secure data sharing for resource-constrained wearable devices [28], [29], and direct data communication between wearable devices was not the focus of such works.

Data sharing between wearable devices and cloud servers. Some works suggest data sharing frameworks for wearable devices and the cloud [28], [29]. These works share the common goal of designing a secure data sharing protocol for resource-constrained devices. However, their focus is on sharing data between user devices and a remote cloud server and enforcing access control for different parties involved with the data, such as patients supplying data and medical

staff. Meanwhile, SecuWear focuses on data sharing between devices owned by one user and enforcing access control for different applications running on the user devices.

Data sharing between IoT devices. Although wearable devices also fall into the category of IoT devices, existing IoT data sharing schemes are unsuitable for our target scenario that focuses on direct data sharing between wearable devices. For example, JEDI [30] proposes a many-to-many end-to-end encryption protocol for IoT devices. Our target scenario is similar to the one targeted by JEDI in that the protocol needs to support many-to-many communication between resource-constrained IoT devices. However, JEDI focuses on scenarios involving a hierarchical structure among devices, such as IoT devices spread across different rooms in a building, and includes decentralized delegation as a core component. Meanwhile, SecuWear focuses on scenarios involving data sharing between devices without any hierarchical structure. Mollah et al. [31] also suggest a secure data sharing framework for IoT devices, but the framework relies on nearby edge servers to offload computationally expensive security operations. Such assumptions cannot be made in our target scenario. Furthermore, some works focus on creating a shared database among IoT devices [32], [33] or among applications running IoT devices [34], [35], rather than maintaining a continuous stream of shared data.

IV. Adversarial Model

To address possible threats in wearable data sharing, we develop a realistic adversarial model considering relevant stakeholders and attacker goals.

A. Stakeholders

We list stakeholders in the wearable device ecosystem and our assumptions about their capabilities and intentions, which serve as the foundation for constructing the adversarial model.

Users. Users are individuals who regularly wear the devices on or in close proximity to their bodies, as well as interact with the applications on the devices. Users are not expected to possess advanced technical expertise, which makes them dependent on the underlying system for security assurances. The primary concern of users is the seamless operation of their devices without compromising personal data.

Application developers. Application developers are responsible for creating and deploying applications that run on wearable devices. These applications may request data from one or more devices. While developers generally have technical expertise, expecting them to implement secure data-sharing mechanisms in a multi-device environment is unrealistic. Moreover, some developers may be malicious, with the intention of collecting user data surreptitiously.

OS developers. Operating system (OS) developers are trusted entities and are expected to provide the necessary security features required for secure data sharing across devices. These developers play a crucial role in ensuring that the OS supports the necessary secure communication protocols.

Hardware manufacturers. Hardware manufacturers are trusted entities. They are not expected to incorporate specialised security hardware, such as Trusted Execution Environments (TEEs). Our protocol is designed to work with standard commodity wearable devices without relying on advanced security features at the hardware level.

B. Adversarial goals

AG1 – Unauthorised Data Access:. The adversary aims to access user data without consent by requesting data without necessary permissions or by intercepting and decrypting wireless transmissions. The adversary is successful only if they obtain unencrypted data or the necessary decryption keys.

AG2 – Data Injection:. The adversary attempts to introduce false or misleading data into a benign application, potentially by masquerading as a legitimate data provider or by intercepting and altering data in transit.

AG3 – Data Corruption:. The adversary aims to corrupt data in transit. Even minor alterations, such as flipping a single bit, constitute a successful attack. This can be achieved through packet injection or manipulation during transmission.

Denial-of-service (DoS) attacks are excluded from the scope of this model, as the focus is on preserving data integrity and confidentiality rather than availability.

C. Adversary model

We thus assume the following strong active local adversary:

1) The adversary can monitor and record all communication between user devices.
2) The adversary can inject packets in an attempt to override legitimate traffic, allowing the adversary to introduce malicious data into the communication stream.
3) The adversary can selectively alter packets during transmission, leading to data corruption or disruption of communication.
4) The adversary may gain temporary physical access to one of the user's devices, such as a wearable involved in data sharing or even the smartphone acting as the orchestrator, allowing direct manipulation without the user's knowledge. However, simultaneous access to multiple devices without user awareness is beyond the scope of this model.

This is a realistic threat model for an adversary that wishes to attack and disrupt the normal operation of a collaborative wearable environment of a user. A survey [36] indicated that wearables are becoming highly personal devices, with about 40% of respondents stating that they wear them all the time. Therefore, it is highly unlikely for an adversary to gain simultaneous access to multiple user devices without the user noticing. We use this model in §VI-A to show that in the absence of the mitigations proposed by SecuWear, the adversary succeeds in their goals to undermine the correct operation of wearable collaboration.

D. Security goals

SecuWear aims to achieve the following key security goals in the presence of such an adversary.

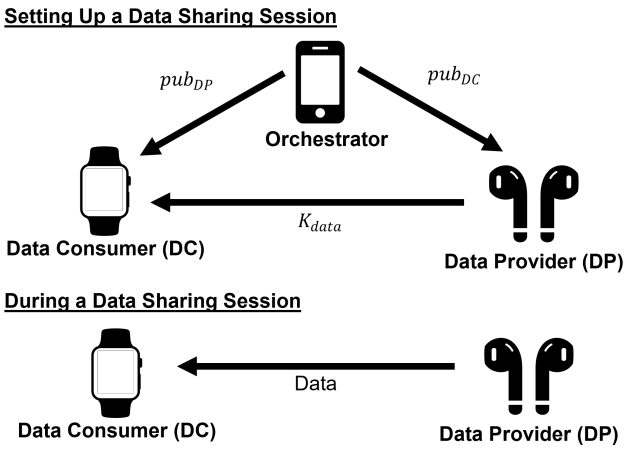

Fig. 2. SecuWear overview.

G1: Confidentiality. The adversary is unable to read user data sent between the wearable devices.
G2: Integrity. The adversary cannot modify data fed into user applications.
G3: Transparency. The protocol is transparent such that users can review which data is being shared with which application and between which devices.

V. SecuWear

To describe SecuWear, we provide an overview, list the underlying assumptions, introduce the roles assigned to each device, walk through the steps involved in sharing data, explain how access control policy is enforced, and detail how transparency is achieved.

A. Overview

The overall process of data sharing through SecuWear is described in Figure 2. When setting up a data sharing session, the orchestrator device helps establish trust between the data consumer and the data provider by providing the public key of the other party to both of them. With asymmetric key encryption, the data provider securely shares the symmetric data encryption key with the data consumer. In a data sharing session, the orchestrator device is no longer involved and the data provider directly shares data with the data consumer by using the data key shared earlier.

B. Assumptions

In designing SecuWear, we made the following assumptions.
The orchestrator maintains long-term authenticated channels with every participating device. We assume that the wearable devices have the capability to establish and maintain long-term authenticated channels with the orchestrator device. This is a realistic assumption of the wearables on the market, which authenticate and establish Wi-Fi or Bluetooth/Bluetooth Low Energy (BLE) connections with a smartphone when the user sets them up. Therefore, we assume that in a user's set of devices, there is one device powerful enough to establish and maintain authenticated and secure channels with every one of the other wearable devices.
All data sharing operations are exclusively handled by a system service, not delegated to applications. To facilitate secure data sharing between wearable devices, we suggest that wearable OSes implement a system service, namely a data manager, that handles inter-device data sharing. For instance, when an application requests data that is not available locally, the local data manager will communicate with the data manager processes of other devices to acquire and deliver the data to the application. This way, the burden on application developers to implement secure inter-device data transmission is reduced, providing a layer of abstraction and lowering the chance for implementation mistakes. Hence, in SecuWear we assume that each wearable device is running a data manager service to handle all inter-device data sharing. The tasks on the orchestrator device, the data consumer device, and the data provider device are carried out by the data manager service running on each device.

C. Protocol Components

In SecuWear, a device can assume three roles. One of the user's devices, most likely the smartphone, takes on the role of the *orchestrator* to facilitate the establishment of secure data sharing sessions between the user's wearable devices. Other devices can act as either a *data consumer* or a *data provider*. As the name suggests, a data provider is a device that shares data upon request from a data consumer who receives the shared data. We explain in further detail the tasks of the orchestrator, data consumer and data provider below.

1) Orchestrator: One of the user's devices assumes the role of the orchestrator to facilitate data sharing sessions between data consumers and data providers. The smartphone would be the most intuitive choice as it already maintains long-term connections with wearable devices. However, any device that can establish and maintain authenticated and secure communication channels with every other user device can assume the role of the orchestrator.
Necessity of the orchestrator. Wearable devices are resource-constrained, hence they cannot be expected to keep track of all other available devices and establish secure communication channels every time data needs to be shared. We propose the use of an *orchestrator*, the device that oversees all involved devices and facilitates data sharing sessions by establishing secure communication channels.
Mitigating the single point-of-failure. The "data hub" approaches suffer from a single point-of-failure by creating a central repository of user data and only allowing data sharing through the central repository. The design of having an orchestrator also introduces a single point-of-failure. However, unlike the "data hub" approach, SecuWear includes mechanisms that mitigate against the single point-of-failure. First, SecuWear decouples data transmission from trust establishment. Data is directly shared between data consumers and data providers, eliminating a central data repository. Second, with SecuWear, wearable devices can still share data after the orchestrator becomes unavailable by replacing the orchestrator with another device. Since we eliminate the role of the "data hub" from

TABLE I
KEYS GENERATED AND RECEIVED BY EACH ENTITY INVOLVED IN A DATA SHARING SESSION; DC AND DP REPRESENT A DATA CONSUMER AND A DATA PROVIDER, RESPECTIVELY.

Entity	Generated Keys	Received Keys
Orchestrator		pub_{DC}, pub_{DP}
Data Consumer	pub_{DC}, $priv_{DC}$	pub_{DP}, K_{data}
Data Provider	pub_{DP}, $priv_{DP}$, K_{data}	pub_{DC}

the orchestrator device, a device with smaller computation power and storage capacity, such as a smartwatch, can assume the role of the orchestrator. Upon realising that the original orchestrator device is unusable, the user can designate another device to assume the role of the orchestrator. This new orchestrator device then establishes secure and authenticated channels with every other wearable device.

Responsibilities of the orchestrator. The orchestrator listens to incoming data requests and connects data consumers to appropriate data providers. It is assumed that the orchestrator knows all the data available from each device, which allows it to connect data consumers to appropriate data providers. It also verifies access control policies, tracks existing sessions to avoid duplicates, and logs valid access requests for transparency.

2) Data Consumer: A device becomes a data consumer when the data manager running on the device receives a data request from a local application for (sensor) data not available locally. Then, the data manager running SecuWear forwards the request to the orchestrator device, which will establish a data sharing session with the appropriate data provider. In the established data sharing session, the data consumer device will receive data and share the data with the application.

3) Data Provider: A device becomes a data provider when it receives a data sharing request from the orchestrator. Upon receiving the request, the data provider will aid in establishing the data sharing session and transmit the requested data. Furthermore, the orchestrator may add another data consumer to an existing data sharing session. The data provider should also listen to such requests. In addition, the data provider also logs data accesses for transparency.

D. Protocol flow

In this section, we describe the protocol flow of SecuWear.

1) Cryptographic Materials: SecuWear uses both symmetric and asymmetric key encryption to establish secure data sharing. The keys generated by each device and keys received by each device are shown in Table I. Both data consumer and data provider devices generate a private ($priv$) and public (pub) key pair to be used in establishing a data sharing session. The data provider also generates a symmetric key K_{data} to encrypt data before sharing.

Inspired by the standard in secure messaging approaches, SecuWear adopts a session-based approach. The keys involved with a data sharing session are only valid for a 15-minute session. Once the 15-minute session is over, a new session needs to be established with a new set of keys. We suggest a length of 15 minutes because the Bluetooth specification suggests the device MAC address to be randomised at least once every 15 minutes [37].

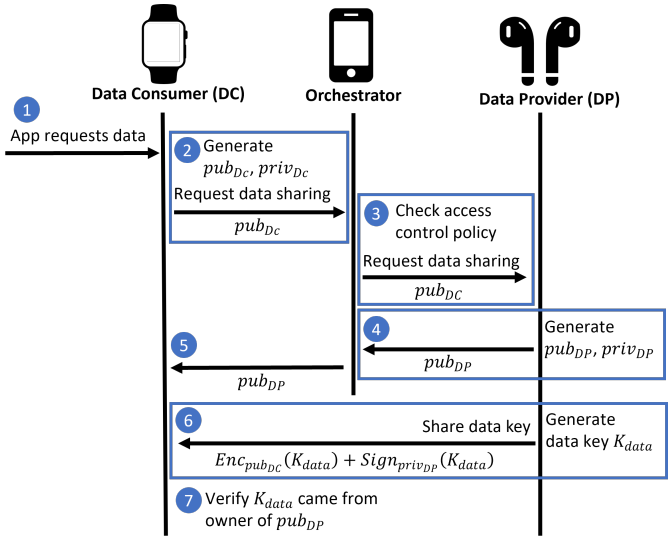

Fig. 3. Data consumer and data provider setting up a data sharing session through the orchestrator.

Key rotation rather than long-term keys is more suitable to our target scenario. If a user revokes a data sharing permission, the cryptographic keys have to reflect such change in the access control policy. Rotating keys per session makes it easy to revoke access as SecuWear can simply not share the keys in the next session. Furthermore, when a key is compromised, the impact of the compromise is contained to that specific session, providing forward and backward secrecy.

2) Setting Up a Data Sharing Session: The steps involved with setting up a data sharing session are shown in Figure 3. When an application requests for data not available in the local device (1), the data manager running on the device will initiate the process of establishing a data sharing session. First, it will generate an asymmetric key pair (pub_{DC} and $priv_{DC}$) and send a data sharing request to the orchestrator along with the public key, pub_{DC} (2). The orchestrator will then check whether the application has permission to access the data it requested and if valid, the orchestrator sends a data sharing request to a user device that can provide the requested data. Along with the data sharing request, the public key of the data consumer, pub_{DC} is also forwarded (3). The data provider device will generate an asymmetric key pair of its own (pub_{DP} and $priv_{DP}$) and share the public key, pub_{DP}, to the orchestrator (4). The orchestrator then shares pub_{DP} with the consumer device (5). Since the orchestrator device is trusted and the communication between each device and the orchestrator device is authenticated and secure, the public keys shared through the orchestrator are considered genuine.

Once the public keys are shared, the data provider generates a symmetric key for encrypting data, K_{data}. Then, the data provider encrypts K_{data} with pub_{DC} (i.e., $Enc_{pub_{DC}}(K_{data})$) and also signs K_{data} with $priv_{DP}$ (i.e., $Sign_{priv_{DP}}(K_{data})$) and shares it to the data consumer (6). The encryption is needed

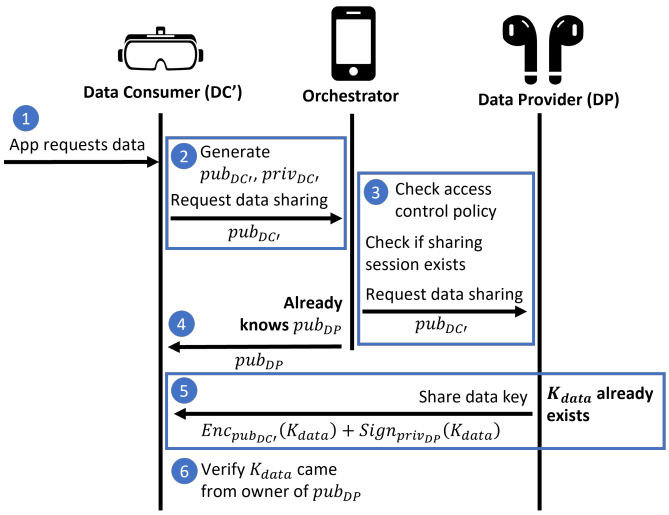

Fig. 4. Adding a data consumer to an existing data sharing session.

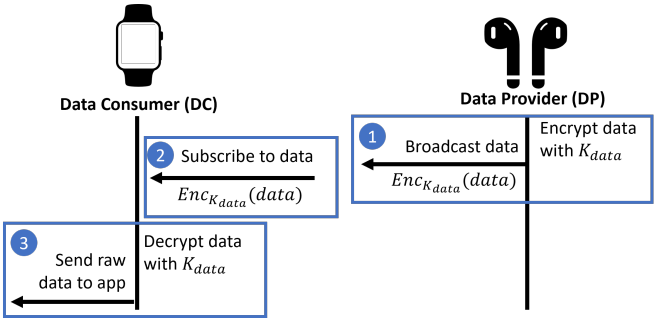

Fig. 5. Data consumer and data provider sharing data in a data sharing session.

to ensure that only the intended data consumer, the owner of $priv_{DC}$ can access K_{data}. Upon receiving the encrypted data key and the signature, the data consumer verifies that the key has been shared from the intended data provider device, the owner of $priv_{DP}$ by verifying the appended signature with pub_{DP} 7. As a result, the data consumer and the data provider have exchanged a symmetric encryption key by relying on a trusted third party–the orchestrator– but not sharing the key itself with the trusted third party.

Furthermore, when an application requests data for which a data sharing session already exists, the orchestrator guides the additional data consumer to join the existing data session. The steps involved with allowing an additional data consumer to join an existing data sharing session are shown in Figure 4. The additional data consumer goes through a similar process as Figure 3, except that step 4 in Figure 3 is skipped. This is because the orchestrator already knows pub_{DP}.

3) Sharing Data in a Session: Once the data sharing session is set up, the data provider device and data consumer device share data through the procedure shown in Figure 5. The data provider encrypts the data to be shared with K_{data} and broadcasts the encrypted packets 1. The data consumer subscribes to the encrypted broadcast 2. The data manager running on the data consumer device receives the data, decrypts it with K_{data}, and feeds the raw data to the correct application 3. All the data packets that are broadcast include a checksum for the data consumer to verify the integrity of received data.

SecuWear employs a pub-sub approach where multiple data consumers subscribe to the same encrypted data broadcast. This design choice makes SecuWear efficient and scalable as additional data consumers do not place any additional burden on the data provider in a data sharing session. The only additional burden is sharing the existing data key with the additional data consumer. However, most common broadcast protocols, such as UDP and BLE broadcast, do not guarantee reliable packet delivery. We leave for future work the investigation into reliable broadcast protocols.

E. Access Control

Wearable OSes should implement a cross-device access control policy to allow users to have control over what data is shared to which applications. In our context, access control refers to the user selectively granting permission for an application to access one type of data in each device. For example, if an application running on a smartwatch wants to access microphone data from another device, the data manager running on the smartwatch will send a data sharing request to the orchestrator, and the orchestrator will see that microphone data from earbuds is available and ask the user whether it should grant the application access to microphone data from earbuds. Through SecuWear, users can have centralised control for inter-device data sharing as all the inter-device data sharing requests are sent to and organised by the orchestrator device.

F. Transparency

To support the goal of providing transparency, SecuWear logs all data accesses in the orchestrator device and each data provider device. When a data sharing session is established, the orchestrator records the following information in the access log: access time, application name, data provider device and data type. When the data provider starts broadcasting data in a data sharing session, the data manager running on the data provider device records the following information in the access log: access time, application name, and data type. To prevent any tampering with access logs, the access logs should be only accessible to the data manager and other privileged processes, not applications. We duplicate access logs in both the orchestrator and data provider devices so users can trace leaked data via the orchestrator log or the data provider logs if one of them is compromised.

VI. Evaluation

Our evaluation of SecuWear is twofold. First, we perform an informal security evaluation to validate that SecuWear achieves the security goals outlined in §IV-D against an adversary described in §IV-C. Second, we implement a prototype and measure the performance overhead introduced by SecuWear.

A. Security Analysis

In this section, we assess whether SecuWear meets the security goals described in §IV-D. We discuss each case with reference to an adversary which tries to defeat each security goal and justify why they fail.

Setup. We assume an adversary with capabilities outlined in §IV-C. *Alice* has four wearable devices and a smartphone. Alice's devices employ SecuWear to share data to provide her with insights into her health. We assume that one wearable device acts as a data consumer *DC*, requesting access to sensor data that another wearable device has, acting as a data provider *DP*. The smartphone acts as the orchestrator *Orc*. *Adv* can attack this data sharing transaction at different key times in the transaction: `Ta` when *DC* sends the request to *Orc*, `Tb` when the request is forwarded from *Orc* to *DP*, `Tc` when *DP* shares data key K with *DC*, and lastly, `Td` during the data sharing session. At any point in time, `Ta`, `Tb`, `Tc`, `Td`, *Adv* can gain possession of any one of the devices.

1) Defeating SecuWear security goals: First, **G1 goal–confidentiality** for the messages sent at `Ta` and `Tb` are guaranteed because of the assumption that the orchestrator device maintains secure and authenticated channels with every participating device. SecuWear provides confidentiality at `Tc` using public key encryption. The authenticity of the public key is guaranteed by the orchestrator device. Furthermore, the confidentiality of the data stream (i.e., at time `Td`) is protected by SecuWear using symmetric key encryption. The participant devices generate ephemeral keys that get refreshed every 15 minutes, limiting any possible compromise to the vulnerability window. Second, SecuWear provides the **G2 goal–integrity** for messages in flight by using a checksum to detect if any errors were introduced. Furthermore, at `Tc` the protocol uses cryptographic signatures to ensure the authenticity of the message carrying the the data stream key. Last, SecuWear provides the **G3 goal–transparency** by having *Orc* and *DP* log data accesses. At `Ta`, when *Orc* receives the data request, *Orc* logs the access, provided it is valid and allowed by the access control policy. Similarly, at `Tb`, when *DP* receives the data request, *DP* logs the request being granted and data shared with *DC*. By protecting the integrity of the messages, the transparency of the protocol is also protected as *Adv* cannot trick either *Orc* or *DP* to write the wrong entry in the transparency logs.

2) Person-in-the-middle: Adv can attempt person-in-the-middle attacks by pretending to *Orc* to be a legitimate device in Alice's set of devices. However, *Orc* maintains secure and authenticated communication channels with all of Alice's devices as per our assumptions in §V-B, which guarantees the authenticity of the devices participating in SecuWear.

3) Physical compromise: Adv can gain temporary physical access to one of Alice's devices. When a device is physically compromised, *Adv* has access to data stored on the compromised device. In this analysis, we consider the additional exposure Alice has during this type of attack due to data sharing. If *Adv* gains access to *DC*, they have access for a limited amount of time to the data being shared by *DP* either until Alice discovers the device is missing and takes measures to revoke permission, or in the worst case until the sharing session expires. Similarly, if *Adv* can gain access to *DP*, then they can manipulate the active data streams, or delete the transparency log. In the former case, the tampering is limited to the 15 minute window, and if Alice notices the compromise before the session expires, she can prompt *DC* to request data again and have *Orc* direct it to a different data provider device. In the latter case, should the transparency log be deleted, there is a complementary log being stored on *Orc*, which can be used by the user upon request to analyse the status of data sharing. Last, if *Adv* gains access to *Orc*, *Adv* can compromise **G1 goal–confidentiality** and **G2 goal–integrity** by arbitrarily establishing data sharing sessions. However, the user can recover from such compromises by designating another device as *Orc*. Furthermore, **G3 goal–transparency** is not compromised. Even if *Adv* deletes the log on *Orc*, the user can reconstruct it by referencing the logs from other devices.

B. Overhead Analysis

1) Protocol Implementation: We implemented a prototype of SecuWear in Python on Raspberry Pi platforms. For encryption algorithms, we used a 2048-bit key RSA for asymmetric encryption and a 256-bit key AES for symmetric encryption.

For wireless communication, we used Wi-Fi to cover a wide range of wearable devices. Nonetheless, SecuWear is independent of the communication medium and can be easily extended to other communication channels such as Bluetooth and Bluetooth Low Energy (BLE).

2) Experimental Setup: We evaluate SecuWear using a Raspberry Pi 4 Model B as the orchestrator and two Raspberry Pi Zero 2 as participant devices. Due to the difficulty of covering all wearable device heterogeneity, these devices were chosen as representatives: the Raspberry Pi 4 for its processing power comparable to smartphones, and the Raspberry Pi Zero 2 for its limited computational capacity and energy efficiency similar to wearables. We measured system overhead in a scenario where an orchestrator sets up a data sharing session between two devices, which then securely share data.

3) Latency: For the latency analysis, we run the aforementioned scenario five times and report the average values. For each run, we measured the latency of all operations involved in SecuWear. The latency for data transmission over Wi-Fi varied greatly for many reasons irrelevant to the design of SecuWear, such as signal interference and network congestion. For a standardized view of SecuWear's latency, we replaced all data transmission latency values with an average Wi-Fi latency measured over multiple runs. Wi-Fi latency was measured 20 times in a lab environment at different times of day, yielding an average of $58.61ms$ ($\sigma = 97.64ms$).

Setting up a data sharing session. Overall, SecuWear takes $1687ms$ ($\sigma = 639.41ms$)[1] to set up a data sharing session. Considering that this operation needs to be performed only once for each 15 minute-long data sharing session, SecuWear imposes less than 0.2% latency overhead to set up a data sharing session. The latency breakdown graph is shown in Figure 6. The exact measurement values are available in the appendix. The results show that the data consumer and

[1]This standard deviation does not account for the standard deviation of data transmission latency.

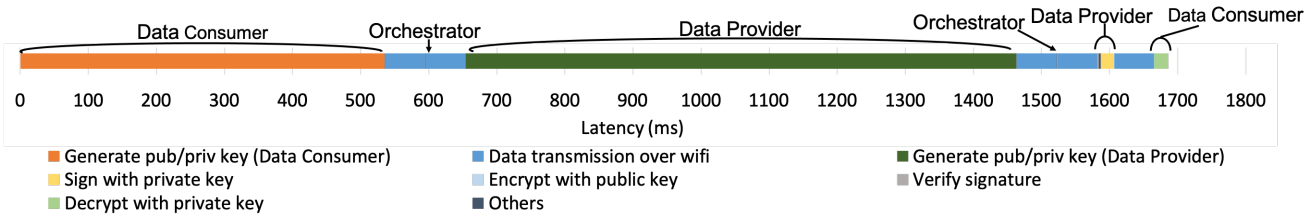

Fig. 6. Latency breakdown for setting up a data sharing session.

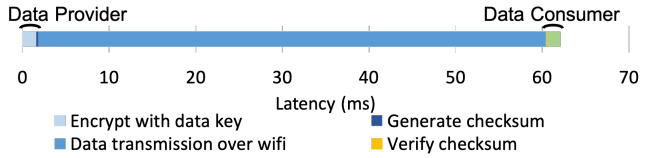

Fig. 7. Latency breakdown for sharing a data packet in a data sharing session.

TABLE II
POWER AND ENERGY CONSUMPTION OF ENCRYPTION ALGORITHMS USED.

	Power (mW)	Time (ms)	Energy (mJ)
RSA Key Generation	400.55	749.47	300.20
RSA Public Key Encryption	434.42	3.79	1.65
RSA Private Key Decryption	401.95	17.98	7.23
AES Key Generation	509.57	0.04	0.02
AES Encryption	566.74	1.84	1.04
AES Decryption	557.42	1.50	0.84

the data provider each generating a public and private key pair take up the majority of the latency overhead, 31.73% and 47.95%, respectively. The next dominant factor is data communication over Wi-Fi, which is the sum of five data transmissions between devices (17.37%). All the other tasks involved with setting up a data session, such as asymmetric key encryption, enforcing access control policy, and checksum, take up less than $50ms$ or 3% of the overall protocol overhead.

Within a data sharing session. In a data sharing session, SecuWear incurs 0.32 ms of latency overhead, which is 0.52% of end-to-end latency with $62.10ms$ ($\sigma = 0.32$)[1] to transmit 1024 bytes of a data packet from the data provider device to the data consumer device. The latency breakdown graph is shown in Figure 7. The exact measurement values are available in the appendix. The results demonstrate that SecuWear imposes negligible latency overhead for data exchange.

4) Power: To demonstrate that SecuWear imposes acceptable power overhead, we measure the power consumption of encryption algorithms used in SecuWear (i.e., 256-bit key AES and 2048-bit key RSA) on a Raspberry Pi Zero 2 device with a Monsoon power monitor. Please note that these encryption algorithms may be replaced with other more lightweight ones.

Table II shows the power and energy overhead of encryption tasks. For power measurements, we report the additional power consumption from running the encryption algorithms. This is because the base (idle) power of a Raspberry Pi Zero device is around 600 mW, which is relatively high compared to a typical wearable device due to the lack of support for a low power mode. Establishing a data session, which involves RSA key generation, encryption, decryption, and AES key generation, consumes around 309.10 mJ of energy per session. With 15-minute intervals, the energy overhead seems negligible. Considering a typical wearable device operating with a battery capacity of 100 mAh and voltage of 3.7 V, 309.10 mJ is around 0.02% [2] of the battery capacity, which is 1.9% [3] of the battery capacity per a full day's operation of SecuWear. During a secure data sharing session, AES encryption and decryption tasks are constantly required. However, it only consumes 1.88 mJ per packet and any data sharing mechanism with encryption will require the same or similar workload.

5) Storage: As described in §V-F, SecuWear generates logs of data access in the orchestrator device and each data provider device. The prototype implementation records the logs in a CSV format. For each data access, 57 bytes were added to the orchestrator log and 49 bytes were added to the data provider log. Considering a typical wearable device with 16MB storage space, it can store around 30,000 entries of data access logs, which covers data sharing for more than 300 days.

VII. CONCLUSION AND FUTURE WORKS

In the near future, we anticipate wearables to dynamically share data to support applications running across multiple devices simultaneously. To facilitate this collaboration, a secure and efficient data sharing protocol is essential. To address this need, we developed a comprehensive threat model and introduced a secure data sharing protocol for wearable devices named SecuWear. The key innovation of SecuWear lies in decoupling trust establishment from data flow, thereby eliminating the need for a centralized "data hub" found in many existing data sharing schemes. Furthermore, by allowing multiple data consumers to subscribe to the same encrypted broadcast from a data provider, SecuWear enables more scalable and resource-efficient data sharing. Our informal security analysis demonstrates that SecuWear is resilient against an active local adversary. We also evaluated the resource overhead with a prototype implementation, showing that SecuWear imposes acceptable overhead for wearable devices in terms of latency, power, and storage.

SecuWear has several limitations. First, it regenerates fresh keys for every session rather than using key ratcheting. While this approach simplifies access revocation, key ratcheting could be more efficient if we incorporate an effective method for managing access revocation. Second, SecuWear utilizes a pub-sub model for efficiency, but popular broadcast protocols

[2] 0.02% = 309.10 mJ / (100 mAh × 3.7 V × 3600 seconds)
[3] 0.19% = 0.02% × (24 hours × 60 minutes / 15 minutes)

may lead to packet loss. Integrating reliable broadcast protocols without compromising efficiency would be beneficial. Lastly, SecuWear requires users to make all access control decisions on the orchestrator device, which may negatively impact usability. We hope to address these limitations for a more robust data sharing protocol for wearable devices.

REFERENCES

[1] A. Ferlini, A. Montanari, C. Min, H. Li, U. Sassi, and F. Kawsar, "In-ear ppg for vital signs," *IEEE Pervasive Computing*, vol. 21, no. 1, pp. 65–74, 2021.

[2] S. M. Phillips, L. Cadmus-Bertram, D. Rosenberg, M. P. Buman, and B. M. Lynch, "Wearable technology and physical activity in chronic disease: opportunities and challenges," *American journal of preventive medicine*, vol. 54, no. 1, pp. 144–150, 2018.

[3] Samsung, "Galaxy ring titanium black - size 10," https://www.samsung.com/uk/rings/galaxy-ring/galaxy-ring-titanium-black-size-10-sm-q500nzkaeub/, 2024, accessed: 2024-08-30.

[4] N. B. Labs, "Omnibuds," https://www.omnibuds.tech/, 2024, accessed: 2024-08-30.

[5] C. Min, U. G. Acer, S. Jang, S. Choi, D. A. Vasile, T. Gong, J. Yi, and F. Kawsar, "An ai-native runtime for multi-wearable environments," *arXiv preprint arXiv:2403.17863*, 2024.

[6] M. Kalanadhabhatta, C. Min, A. Montanari, and F. Kawsar, "Fatigueset: A multi-modal dataset for modeling mental fatigue and fatigability," in *International Conference on Pervasive Computing Technologies for Healthcare*. Springer, 2021, pp. 204–217.

[7] S. Gashi, C. Min, A. Montanari, S. Santini, and F. Kawsar, "A multidevice and multimodal dataset for human energy expenditure estimation using wearable devices," *Scientific Data*, vol. 9, no. 1, p. 537, 2022.

[8] R. C. King, E. Villeneuve, R. J. White, R. S. Sherratt, W. Holderbaum, and W. S. Harwin, "Application of data fusion techniques and technologies for wearable health monitoring," *Medical engineering & physics*, vol. 42, pp. 1–12, 2017.

[9] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Information Fusion*, vol. 35, pp. 68–80, 2017.

[10] I. Analog Devices, "Max78000 - features and benefits," https://www.analog.com/en/products/max78000.html, 2024, accessed: 2024-08-30.

[11] Google, "Coral dev board micro," https://coral.ai/products/dev-board-micro/, 2024, accessed: 2024-08-30.

[12] T. Gong, S. Y. Jang, U. G. Acer, F. Kawsar, and C. Min, "Collaborative inference via dynamic composition of tiny ai accelerators on mcus," *arXiv preprint arXiv:2401.08637*, 2023.

[13] D. A. Vasile, F. Kawsar, and C. Min, "Emerging paradigms in wearable security: Adaptable and secure sandboxing for on-the-fly collaboration among wearables," *IEEE Security & Privacy*, pp. 2–11, 2024.

[14] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys & Tutorials*, vol. 20, pp. 3453–3495, 2018.

[15] F. Blow, Y.-H. Hu, and M. Hoppa, "A study on vulnerabilities and threats to wearable devices," in *Journal of The Colloquium for Information Systems Security Education*, vol. 7, no. 1, 2020, pp. 7–7.

[16] J. Xin, V. V. Phoha, and A. Salekin, "Combating false data injection attacks on human-centric sensing applications," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, pp. 1–22, 2022.

[17] Cyberior, "Wearable devices and data security," https://cyberior.com/hk/blog/2019/06/17/wearable-devices-and-data-security/, 2019, accessed: 2024-08-30.

[18] S. Developer, "Samsung health android - overview," https://developer.samsung.com/health/android/overview.html, 2024, accessed: 2024-08-30.

[19] A. Inc., "Healthkit documentation," https://developer.apple.com/documentation/healthkit/, 2024, accessed: 2024-08-30.

[20] A. Orzikulova, D. A. Vasile, F. Kawsar, and C. Min, "Time-bound contextual bio-id generation for minimalist wearables," *arXiv preprint arXiv:2403.00889*, 2024.

[21] Garmin, "Garmin connect," https://connect.garmin.com/, 2024, accessed: 2024-08-30.

[22] M. Fitness, "Mi fitness," https://statics.teams.cdn.office.net/evergreen-assets/safelinks/1/atp-safelinks.html, 2024, accessed: 2024-08-30.

[23] D. Yeke, M. Ibrahim, G. S. Tuncay, H. Farrukh, A. Imran, A. Bianchi, and Z. B. Celik, "Wear's my data? understanding the cross-device runtime permission model in wearables," *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 80–80, 10 2023.

[24] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," *Journal of Cryptology*, vol. 33, pp. 1914–1983, 10 2020.

[25] M. Alatawi and N. Saxena, "Sok: An analysis of end-to-end encryption and authentication ceremonies in secure messaging systems," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 187–201.

[26] M. Ahmed-Rengers, D. A. Vasile, D. Hugenroth, A. R. Beresford, and R. Anderson, "Coverdrop: Blowing the whistle through a news app," *Proceedings on Privacy Enhancing Technologies*, no. 2, pp. 47–67, 2022.

[27] W. He, N. Reitinger, A. Almogbil, Y.-S. Chiang, T. J. Pierson, and D. Kotz, "Contextualizing interpersonal data sharing in smart homes," *Proceedings on Privacy Enhancing Technologies*, 2024.

[28] M. Tanveer, A. U. Khan, M. Ahmad, T. N. Nguyen, and A. A. El-Latif, "Resource-efficient authenticated data sharing mechanism for smart wearable systems," *IEEE Transactions on Network Science and Engineering*, vol. 10, pp. 2525–2536, 9 2023.

[29] H. Amintoosi, M. Nikooghadam, S. Kumari, F. Jun, H. Xiong, S. Kumar, and J. J. Rodrigues, "Secure and authenticated data access and sharing model for smart wearable systems," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5368–5379, 2021.

[30] S. Kumar, Y. Hu, M. P. Andersen, R. A. Popa, and D. E. Culler, "JEDI: Many-to-Many End-to-End encryption and key delegation for IoT," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1519–1536. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-sam

[31] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted internet of things," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 34–42, 2017.

[32] R. Li, C. Shen, H. He, X. Gu, Z. Xu, and C.-Z. Xu, "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 344–357, 2017.

[33] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2171–2180, 2013.

[34] C. Stach and B. Mitschang, "The secure data container: An approach to harmonize data sharing with information security," in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, vol. 1. IEEE, 2016, pp. 292–297.

[35] ——, "Curator—a secure shared object store: design, implementation, and evaluation of a manageable, secure, and performant data exchange mechanism for smart devices," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 533–540.

[36] Ericsson ConsumerLab, "Wearable technology and the internet of things," Ericsson, Tech. Rep., 2016, accessed: 2024-12-05. [Online]. Available: https://www.ericsson.com/4ac5b3/assets/local/reports-papers/consumerlab/reports/2016/wearable-technology-and-the-internet-of-things-ericsson-consumerlab-2016.pdf

[37] B. S. I. Group, "Specification of the bluetooth system, core v4.0," Bluetooth SIG, Tech. Rep. Core v4.0, 2010, available at: https://www.bluetooth.com/specifications/.

[38] H. Sync, "Health sync," https://healthsync.app/, 2024, accessed: 2024-08-30.

[39] A. Inc., "Protecting user privacy in healthkit," https://developer.apple.com/documentation/healthkit/protecting_user_privacy, 2024, accessed: 2024-08-30.

[40] Samsung, "Security and privacy - samsung sustainability," https://www.samsung.com/uk/sustainability/security-and-privacy/privacy/, 2024, accessed: 2024-08-30.

TABLE III
METHODS FOR AN APPLICATION TO SHARE DATA WITH ANOTHER DEVICE.

		Type 1			Type 2		Type 3	Ours
Security Goals	Features	Apple HealthKit [19]	Samsung Health SDK [18]	Health Sync [38]	Garmin Connect [21]	Mi Fitness [22]	Direct Data Sharing Between Apps	SecuWear
Confidentiality	Data encrypted in transit	Yes	Yes	Yes	Yes	Yes	Depends*	Yes
	Does not create a central data repository	No	No	No	No	No	Yes	Yes
	Data stored locally in user devices	Yes	Yes	Yes	No	No	Yes	Yes
Integrity	Resilience to random bit flip attack	Yes	Yes	Unknown	Unknown	Unknown	Depends*	Yes
	Resilience to Man-In-The-Middle Attack	Yes	Yes	Unknown	Unknown	Unknown	Depends*	Yes
Transparency	App data access log available	No	No	No	No	No	No	Yes
	Centralised control of app data access	Yes	Yes	Yes	Yes	Yes	No	Yes

*Type 3 refers to each application implementing its own inter-device data sharing mechanism. Hence, whether or not some features are implemented cannot be generalised and is dependent on the specific implementation.

APPENDIX

A. Existing Data Sharing Methods for Wearable Devices

Commercial wearable devices typically come with a counterpart smartphone application component through which useful features and insights are provided to users. In these we observe three types of data sharing approaches, based on the sharing mechanism: (Type 1) collecting and sharing all data in a smartphone, (Type 2) collecting and sharing all data in a cloud server, and (Type 3) direct data sharing between applications [23]. We analyse each commercial approach from information widely available on the company websites, relevant papers, as well as app analysis and provide a best-effort comparison of SecuWear against each of the data sharing methods in Table III.

Type 1 sharing mechanism refers to utilising a platform or an application that collects all user data from wearable devices in the smartphone and allow other applications to access data from the smartphone. Native health platforms, such as Apple HealthKit [19] and Samsung Health SDK [18], are examples of such platforms. Since such native platforms only support a closed ecosystem of devices from the Apple and Samsung manufacturers respectively, third-party applications, such as Health Sync [38], have been developed to share user data to the native health platforms and other applications when an unsupported device is worn by the user. The native health platforms employ necessary cryptographic primitives to provide secure data communication and stored user data is encrypted when the smartphone is locked [39], [40]. However, these data sharing schemes introduce a powerful single point-of-failure. If the adversary gains access to an unlocked smartphone, the adversary can gain access to all user data collected from all devices. Furthermore, the data sharing schemes do not function without a smartphone.

Alternatively, in Type 2 approaches, all the user data is centrally collected in the cloud and other applications are granted access to the data directly from the cloud. Garmin Connect [21] and Mi Fitness [22] are examples of such applications that enable data sharing through the cloud. This method not only suffers from the similar single point-of-failure issue, but also introduces additional concerns regarding sensitive data being stored in cloud storage.

Last, Type 3 approaches refer to having the application run on both the device collecting data and the other device requesting access to the collected data. The application can then directly share data between the two devices. Such a data sharing scheme is not ideal for several reasons: it relies on the application developers to accurately implement secure data transmission, which is impractical; it can confuse users in terms of what permissions they granted to each application [23]; or there may be redundant data streams due to a lack of an orchestrator device organising all data sharing schemes.

With SecuWear, all raw user data can continue to be stored locally in user devices, which enables private processing on device, without accumulating all the user data in one device. While SecuWear does not create a central data repository, SecuWear does enable centralised control over user data collected with various user devices. Hence, unlike Type 3 approaches, SecuWear can take into account that different devices have access to unique vantage points on the body, as well as different types of sensor data, and facilitate more efficient and collaborative data sharing. Since SecuWear is a protocol developed to be integrated at the OS layer, we do not rely on individual application developers to accurately implement secure inter-device communication. In addition, to allow users to be more knowledgeable about how their data is being shared, we create transparency logs for data accesses from applications.

B. Evaluation Results

TABLE IV
TOTAL TIME TAKEN TO CARRY OUT DIFFERENT TASKS WHILE SETTING UP A DATA SHARING SESSION.

Task	Latency (ms)	Proportion (%)
Generate pub/priv key (Data Producer)	808.80	47.95
Generating pub/priv key (Data Consumer)	535.18	31.73
Data transmission over Wi-Fi	293.05	17.37
Decrypt with private key	20.36	1.21
Sign with private key	18.60	1.10
Others	5.72	0.34
Log data access	1.88	0.11
Verify signature	1.31	0.08
Encrypt with public key	0.98	0.06
Generate checksum	0.47	0.03
Verify checksum	0.51	0.03
Generate data key	0.06	0.003
Total latency	1686.91	100.00

TABLE V
TOTAL TIME TAKEN TO CARRY OUT DIFFERENT TASKS WHILE SHARING A DATA PACKET IN A DATA SHARING SESSION.

Task	Latency (ms)	Proportion (%)
Data transmission over Wi-Fi	58.61	94.37
Encrypt with data key	1.59	2.55
Decrypt with data key	1.59	2.55
Generate checksum	0.22	0.36
Verify checksum	0.10	0.16
Total latency	62.10	100.00