"Use It—No Need to Shake It!" Accurate Implicit Authentication for Everyday Objects with Smart Sensing

CHUXIONG WU*, University of South Carolina, USA
XIAOPENG LI*, University of South Carolina, USA
FEI ZUO, University of South Carolina, USA
LANNAN LUO, University of South Carolina, USA
XIAOJIANG DU, Stevens Institute of Technology, USA
JIA DI, University of Arkansas, USA
QIANG ZENG†, University of South Carolina, USA

Implicit authentication for traditional objects, such as doors and dumbbells, has rich applications but is rarely studied. An ongoing trend is that traditional objects are retrofitted to smart environments; for instance, a contact sensor is attached to a door to detect door opening (but cannot tell "*who is opening the door*"). We present the first accurate implicit-authentication system for retrofitted everyday objects, named `MoMatch`. It makes an authentication decision based on a single natural object use, unlike prior work that requires *shaking* objects. `MoMatch` is built on the observation that an object has a motion typically because a human hand moves it; thus, the object's motion and the legitimate user's hand movement should *correlate*. The main challenge is, given the *small* amount of data collected during one object use, how to measure the correlation accurately. We convert the correlation measurement problem into an image comparison problem and resolve it using neural networks successfully. `MoMatch` does not need to profile the user's biometric information and is resilient to mimicry attacks.

CCS Concepts: • **Security and privacy → Authentication**; • **Networks → Mobile and wireless security**.

Additional Key Words and Phrases: Implicit Authentication, Everyday Objects, Deep Learning, Smart Sensing

ACM Reference Format:
Chuxiong Wu, Xiaopeng Li, Fei Zuo, Lannan Luo, Xiaojiang Du, Jia Di, and Qiang Zeng. 2022. *"Use It—No Need to Shake It!"* Accurate Implicit Authentication for Everyday Objects with Smart Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 146 (September 2022), 25 pages. https://doi.org/10.1145/3550322

*The two authors contributed equally to this research.
†Corresponding author.

Authors' addresses: Chuxiong Wu, chuxiong@email.sc.edu, University of South Carolina, Columbia, South Carolina, USA; Xiaopeng Li, xl4@email.sc.edu, University of South Carolina, Columbia, South Carolina, USA; Fei Zuo, fzuo@email.sc.edu, University of South Carolina, Columbia, South Carolina, USA; Lannan Luo, lluo@cse.sc.edu, University of South Carolina, Columbia, South Carolina, USA; Xiaojiang Du, xdu16@stevens.edu, Stevens Institute of Technology, Hoboken, New Jersey, USA; Jia Di, jdi@uark.edu, University of Arkansas, Fayetteville, Arkansas, USA; Qiang Zeng, zeng1@cse.sc.edu, University of South Carolina, Columbia, South Carolina, USA.

2474-9567/2022/9-ART146 $15.00
https://doi.org/10.1145/3550322

Table 1. Comparison with other approaches.

Approach	F1: Implicit authentication	F2: No dependency on user interface	F3: Need only one natural use	F4: Accurate	F5: No user profiling	F6: Resilient to attacks	Need user-side device?
SenseTribute [38]	✓	✓	✓	✗	✗	✗	No
Bluetooth [3]	✓	✓	✓	✗	✓	✗	Yes
ZEBRA [55]	✓	✗	✗	✓	✓	✗	Yes
TouchAuth [93]	✗	✗	✗	✓	✓	✗	Yes
Pet-2-Auth [52]	✗	✗	✗	✓	✓	✓	Yes
ShakeUnlock [24]	✗	✓	✗	✓	✓	✓	Yes
`MoMatch`	✓	✓	✓	✓	✓	✓	Yes

1 INTRODUCTION

An ongoing trend is that traditional objects are retrofitted with on-object sensor nodes; for instance, a contact sensor is attached to a door to detect door opening (but cannot tell "*who is opening the door*"). Implicit authentication for such retrofitted traditional objects, which implicitly recognizes the object user, has rich applications. For example, a smart health center can make use of it to keep track of how many times a dumbbell has been lifted by a patient; when a user pushes a door, a smart home can trigger personalized automation.

Many implicit-authentication works use behavioral biometrics to distinguish users [18, 30, 38, 42, 60, 73, 74, 76]. For example, SenseTribute [38] infers users based on the motion data collected from on-object sensor nodes attached to objects. It has a very low accuracy and requires profiling each user's behavioral biometrics. Proximity-based approaches built on wireless signals can authenticate users implicitly (e.g., Bluetooth [3]) or explicitly (e.g., NFC [26]). But they are vulnerable to the very mature *mafia fraud* (i.e., radio relay) attacks [29, 39, 86]. ZEBRA [55] provides implicit authentication for desktops, requiring mice and keyboards, which do not exist on most retrofitted objects. (TouchAuth [93], Pet-2-Auth [52], and ShakeUnlock [24] conduct explicit authentication.)

We propose the first *accurate* implicit authentication system for retrofitted objects, named `MoMatch`. It has the following prominent features. (**F1**) ***implicit authentication***. (**F2**) It does ***not depend on specific interfaces***, such as keyboards and mice. (**F3**) It makes an authentication decision based on ***one single natural object use***, such as pushing a door or picking up a gun. This is unlike prior work (such as ShakeUnlock [24] and ShaVe/ShaCK [58]) that requires shaking objects. (**F4**) It is ***accurate***. (**F5**) It does ***not need user profiling***. (**F6**) It is ***resilient*** to attacks (such mimicry attacks and radio relay attacks). Table 1 summarizes the comparison with other work and is further interpreted in Related Work (Section 2.2).

Potential Applications of `MoMatch`. Because of these prominent features of `MoMatch`, a variety of *applications* can be enabled. (A1) ***Personalization for smart environments***. When a user pushes a door or picks up a TV remote, `MoMatch` can provide the user identity information, which can be used for, e.g., room temperature adjustment and TV program recommendation. (A2) ***Smart health***. Without hiring nurses for counting, a rehabilitation center can make use of `MoMatch` to collect important data, e.g., *"which dumbbell has been lifted by which patient for how many times."* (A3) ***Forensics***. In an Amazon warehouse or an airport, for example, `MoMatch` can provide evidence *"which package/suitcase has been moved by which employee."* (A4) ***Continuous monitoring***. Consider a refrigerator in a lab that contains sensitive samples or a cabinet storing medical records; when its door, which is left unlocked (ideally, it should be locked every time after the use, but humans make mistakes), is pulled by an attacker, `MoMatch` can recognize the attack and trigger a siren alarm. We clarify that `MoMatch` cannot prevent an adversary from pulling the door. However, given the fact that many sensitive objects are left unlocked (e.g., many accidental shootings by children are because of this [43]), `MoMatch` may play a unique role by providing accurate continuous monitoring and enabling early intervention.

Our Insights. The work closest to ours is SenseTribute [38], which also attaches an inertial measurement unit (IMU) sensor node to a traditional object to collect motion data in order to recognize the user identity implicitly. It assumes each user has a habit of using an object (while our work does not). It first profiles each user by having

her use an object multiple times to collect her biometric template. When an object is used, the model compares the captured motion data with all the templates and predicts the user identity. Without considering mimicry attackers or habit changes, given merely five users, the accuracy of SenseTribute [38] averages only 74%. (As illustrated in Table 1, SenseTribute has the advantage of not requiring users to wear wristbands. When accuracy and security are not the most important goals, SenseTribute might be more usable than MoMatch, but note that SenseTribute needs per-user profiling).

Our first insight is that the approach of SenseTribute [38] loses information significantly in two aspects. 1) The approach omits the ***realtimeness*** information completely. Essentially, it compares the *current* object motion data with biometric templates established in the *past*. Consequently, regardless of what the victim user is doing *now*, as long as an attacker mimics the victim's historical object uses well, the approach can be fooled. 2) The approach ignores the ***uniqueness*** of each use. Thus, it is easier for an attacker to succeed, since the attacker does not need to mimic a particular object use (but a template). On the other hand, if a user uses an object in an unusual way due to, e.g., a hurry or emotional change, the approach of SenseTribute [38] may fail.

Our second insight is that this behavioral biometrics based approach has to handle ***habit changes***. A user may change her habit over time. Without re-profiling (which hurts usability), the system accuracy will degrade.

Thus, to boost the accuracy and resilience to attacks, we need to recover and make use of the lost information. To increase the usability and robustness, we aim to eliminate the habit-change problem.

Main Idea. Instead of relying on behavioral habits, MoMatch is built on a solid causal relationship: an object has a motion usually because a human hand moves it. For example, to open a door one needs to push or pull it. Thus, the object's motion and the legitimate user's hand movement must *correlate* to validate a legitimate use. The authentication problem is then converted to a motion-data correlation evaluation problem.

Nowadays, both wearable devices and on-object sensor nodes are prevalent. The worldwide wearable market is estimated to increase by an average of 20% each year over the coming years and have 243 million unit sales by 2022 [25]. Meanwhile, there is an emerging trend in on-object sensing devices (i.e., detachable wireless sensor nodes), which retrofit objects such as doors, windows, and drawers to smart environments [10, 38]. These nodes are often equipped with accelerometers and/or gyroscopes to monitor the object status [65, 79]. We thus propose to (1) capture the owner's hand movement data via a wearable device, such as a smartwatch or fitness band, which usually has built-in inertial measurement units, and (2) capture the object's motion data via an on-object sensor node attached to the object. When a sensitive object is being used, the motion data from both the on-object sensor node and the wearable device of the owner is collected, and a correlation score between the two streams of data is calculated to determine whether the current use is legitimate.

MoMatch exploits the ***realtimeness*** information. Not only the motion data but also the ***timestamp*** of each piece of motion data participates in the correlation evaluation. An attacker who mimics a user's object use will fail, as the average human reaction time is larger than 200ms [31, 44, 59] and such a time difference is detected as attacks by our model. Second, MoMatch makes use of ***uniqueness*** of an object use, as any unique details will be reflected on both sides (sensor node and wristband) and used in correlation evaluation. An attacker simply mimics a user's habit but ignores the unique details of the object use being attacked will probably fail. Third, MoMatch does not have the ***habit-change*** issue, as our correlation evaluation does not rely on habits.

Challenges. (1) As the hand rotation around the wrist cannot be precisely captured by the wristband, the motion data of the wristband is not identical to that of the object, which complicates correlation evaluation. (2) MoMatch authenticates based on a single object interaction, such as pushing a door, which provides scarce data. (3) There are a variety of objects, e.g., *fixed-motion* objects like doors vs. *free-motion* objects like guns, which makes it challenging to come up with a uniform solution.

Technique. We propose "*Imagified Curve Comparison*" (ICC) to resolve the motion-data correlation evaluation problem. Specifically, given two sequences of motion data (one from a user's wristband and the other a sensor

node), they are converted to two curves. We plot each curve to an image, and *transform the correlation-evaluation problem into an image-comparison problem*. We then design a neural network with the Siamese architecture [85] to compare the two imagified curves, as the Siamese architecture performs well in comparison [8, 14, 97].

We build prototypes and evaluate MoMatch on 10 various objects (doors, shotguns, dumbbells, etc.). The results show that MoMatch (1) achieves high accuracies (an average AUC = 0.984 across 10 objects) using surprisingly *small* datasets, (2) keeps accurate for users never seen, (3) makes the authentication decision very fast (within 2.5s), (4) has low energy consumption, and (5) is resilient to mimicry attacks. We make the following contributions.

- **Approach.** We propose the *first* accurate implicit authentication approach for retrofitted objects. Unlike prior work (such as ShakeUnlock [24] and ShaVe/ShaCK [58]) that requires shaking objects, MoMatch conducts authentication based on a natural object use. Plus, profiling of user biometrics is not needed, meaning a trained model can be used for authenticating users that are never seen during training.
- **Technique.** (1) We propose the *Imagified Curve Comparison* technique that converts the motion-data correlation evaluation problem into an image comparison problem. We leverage recent advances in deep learning, and design a Siamese neural network to resolve the problem successfully. (2) While explainable AI is a challenging task, we use visualization to vividly demonstrate the effectiveness of the design. (3) We study how to exploit two kinds of motion data (acceleration and gyroscope) to increase accuracy. (4) We illustrate that one single unified model can be trained for the authentication of a variety of objects. (5) We extensively compare our design with prior designs on signal similarity calculation and some more straightforward designs, such as SVM and RNN (recurrent neural network).
- **Prototyping and Evaluation.** We build prototypes of MoMatch for 10 different objects and perform comprehensive evaluation in terms of accuracy, efficiency, robustness, security, and applications.

2 BACKGROUND AND RELATED WORK

2.1 Background: Implicit vs. Explicit Authentication

Implicit authentication recognizes a user's identity without requiring explicit authentication efforts from the user. By contrast, *explicit authentication* requires users to explicitly conduct certain authentication operations, e.g., shaking an object, scanning fingerprints, and inputting passwords. Explicit authentication usually has a serious security issue: once unlocked, a device can be used illegally without raising any alarms. For example, an unlocked desktop, if left unattended, can be illegally accessed. ZEBRA [55], an implicit authentication approach for desktops, is proposed to mitigate the problem. Analogous to ZEBRA for desktops, MoMatch complements (rather than replaces) traditional explicit authentication, such as locks and passcode-protected safes, to provide continuous monitoring. In addition to continuous monitoring, MoMatch supports many other applications, such as smart-environment personalization, smart health, and forensics (see Section 1).

2.2 Related Work

A large body of work exploits behavioral biometrics to propose implicit authentication approaches, such as keystroke dynamics [63]/mouse movements [95] on computers, touch behaviors [12, 30, 51, 78] on smart phones, and SenseTribute [38] for retrofitted objects. They all need to profile a user's biometric information and deal with habit changes, while MoMatch does not. Besides, unlike SenseTribute, MoMatch captures ***realtimeness*** and ***uniqueness*** of each object use, and demonstrates a much higher accuracy and resilience to attacks.

Proximity detection techniques can be applied to implicit authentication, such as ZIA [16], Apple's Auto Unlock [3], Sound-Proof [45], Proximity-Proof [37], and NAuth [96]. (Some approaches use the common context in proximity for implicit pairing, such as audio [75], gait [92], CSI [91], and luminosity [61].) However, they can only confirm whether a user is nearby, but not whether a user is actually *using* the object [55], causing unintentional authentications if a user is just standing near or passing by the object. Moreover, when multiple users are near an object, it is unclear who is the one using it. Worse, techniques that use short-range Bluetooth or

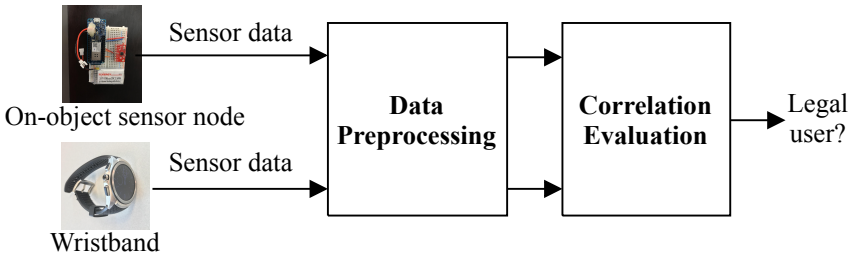


Fig. 1. System architecture.

Received Signal Strength Indicator (RSSI) to prove proximity are vulnerable to *radio relay attacks* [27–29, 39, 86], which relay radio signals to mislead the authentication system to believe that a user's token is nearby. For instance, the practicality of radio relay attacks against the keyless entry system of modern cars, without cracking crypto-keys, has been well demonstrated [27]. Car thefts applying relay attacks are not only real [83] but also cheap ($22) [89]. Readers are referred to [69] about the insecurity of applying RSSI, radio fingerprinting, etc.

Approaches, such as P2Auth [52], G2Auth [90], TouchAuth [93], SAW [56], ShaVe/ShaCK [58], T2Pair [53], and Touch-And-Guard [87], perform explicit authentication.[1] In contrast, `MoMatch` aims at implicit authentication. For example, both `MoMatch` and ShakeUnlock [24] have a user wearing a wristband interact with an object having an IMU sensor to perform authentication. The two differ much in the following three aspects. (1) *Implicit vs. explicit Authentication.* In ShakeUnlock, a user wearing a smartwatch **shakes** a smartphone intensively to authenticate herself to the smartphone, which is explicit authentication. `MoMatch` is an implicit authentication system. When a user uses an object naturally, e.g., pushing a door, drawing a drawer, picking up a gun, we collect the data generated from that object use to perform the authentication. (2) Due to the two types of very different interactions, they generate very different data. Because of the shaking operations, the motion data from both sides contain tens of sharp peaks and valleys (see Figure 2 of [24]). `MoMatch` does not have that rich data to make use of. (3) They use entirely different techniques for calculating correlation scores. ShakeUnlock relies on complex methods for extracting manually-selected attributes for computing coherence-based similarity, requiring tedious feature engineering. `MoMatch` converts the motion data correlation calculation problem into an image comparison problem, and resolves it using a Siamese neural network, which does not need feature engineering.

3 SYSTEM OVERVIEW

3.1 System Architecture

As shown in Figure 1, `MoMatch` comprises three types of entities: (a) an on-object sensor node attached to an object; (b) a wristband worn by the legitimate user; and (c) an authentication process, which is an app running in the wristband or a server.[2] When a motion of the object is detected by its sensor node, the sensor node notifies the authentication process, which collects acceleration and/or gyroscope data from the two devices. The process then pre-processes the data and calculates a correlation score, based on the motion data collected during *one* object interaction, such as pushing a door, pulling a drawer, or picking up a remote, which takes only seconds.

Categorization of objects. (1) *fixed-motion* objects, which can only be moved around some axis or along a glide, such as doors and drawers; and (2) *free-motion* objects that can be moved around freely, such as guns and dumbbells. Free-motion objects impose extra challenges for authentication since the motions are more flexible; *they are not considered by many prior works* [38, 71, 81]. Both are considered by `MoMatch`.

[1] ShaVe/ShaCK [58], T2Pair [53], and Touch-And-Guard [87] are proposed for pairing, but can be adapted to *explicit* authentication [52].
[2] The deployment of the authentication process mainly depends on the application scenario. For instance, for smart home personalization, it can be deployed in a local hub or cloud server. We focus on the authentication technique itself, regardless of the deployment choice.

3.2 Assumptions, Threat Model, and Limitations

3.2.1 Assumptions. (1) The sensor data transmission is via a secure communication channel protected by a key, which can be established using any secure pairing method [49]. This ensures data confidentiality, integrity, and freshness. (2) The clocks of the wristband and the sensor node are synchronized using a time synchronization method [20, 62]. (3) When the wristband is *taken off*, it can detect this and any subsequent object use will be considered as invalid (and reported to, for example, the user's smartphone). When the user *puts on* the wristband, she needs to authenticate herself to the wristband using a PIN (to activate the authentication app). Both techniques are available on smartwatches such as Apple [2] and Android Wear watches [15]. (4) The wristband and the sensor node have wireless communication capabilities, and embedded accelerometers and/or gyroscopes. (5) The authentication process is trusted, and does not leak privacy information (including the motion data). Similar assumptions are used in prior wristband-based authentication works [52, 55–58, 93].

3.2.2 Threat model. The adversary $\mathcal{A}$, who is illegally using an object $\mathcal{O}$, aims to fool the authentication system to believe that a victim user $\mathcal{V}$ is using $\mathcal{O}$. We assume $\mathcal{A}$ can clearly see $\mathcal{V}$'s operations and $\mathcal{A}$ launches attacks by mimicking $\mathcal{V}$. We consider two types of mimicry attacks. *Attack-I*: $\mathcal{V}$ is performing an activity that is different from $\mathcal{A}$'s activity on $\mathcal{O}$. E.g., $\mathcal{A}$ pushes a door while $\mathcal{V}$ is walking. *Attack-II*: $\mathcal{V}$ is performing an activity that is the same as $\mathcal{A}$'s. For instance, $\mathcal{V}$ pushes a door, and $\mathcal{A}$ mimics $\mathcal{V}$ to push another door. While Attack-II unsurprisingly has a higher success rate, Attack-I can be launched without waiting for the victim user to do the targeted operation and is thus also threatening. We thus examine both types of attacks. Our evaluation (Section 8.6) shows that `MoMatch` is resilient to both attacks above.

The following attacks are beyond scope. Denial-of-Service is possible by jamming the wireless traffic. There are mature solutions [6, 68] to jamming attacks. Plus, the authentication process can raise alarms if the communication channel is disrupted. Attackers may use a camera to capture the legitimate user's operation and use a robot to precisely recreate the operation on the target object. The recreating involves latency, which results from video analysis, data transmission, planning, and controlling actuators. Robotic imitation of human actions is actively studied, but still very limited. For example, *NAO*, one of the state-of-the-art humanoid robots, has a latency of 200ms to execute a prescribed motion [23], regardless of its high price ($9,000 [72]). Another study indicates that the end-to-end delay from human-waving to robot-waving is as large as 1.72 seconds [9]. The large latency of robotic imitation probably cannot be resolved in the near future, which makes the threat unrealistic. We thus consider robotic mimicry attacks out of scope.

Since neural networks are known to be vulnerable to adversarial examples (AEs), attackers may attempt to create AE [32] operations to fool our neural network based system. How to compute physical AE operations is an interesting separate research question. Unlike fooling an image classifier, AE attacks against `MoMatch` has extra constraints: an AE needs to be computed realtime based on how the victim user is using an object and then the attacker physically moves the victim object accordingly; note that each time `MoMatch` compares the information collected from two sides (i.e., wristband and sensor node), which reflects that specific object use.

Given a stolen wristband, the attacker cannot exploit it directly, since it is protected by a PIN as clarified in our assumptions. However, the attacker may use certain attacks (e.g., [36]) to steal the key stored in the wristband. Such attacks are beyond the scope of this work.

3.2.3 Deploying `MoMatch` *for multiple objects and users.* While the focus of this work is a technique that enables accurate implicit authentication for object uses, we briefly discuss the deployment of `MoMatch` for a smart environment that has multiple objects and users. It is very common that a smart environment uses a local or cloud IoT server to keep track of the device states, access control and automation. First, each new device (object or wristband) gets paired with the IoT server to establish a long-term key/token, which is also the current practice of IoT [1, 33]. This way, given a new wristband, the user only needs to pair it once (rather than pair it with each

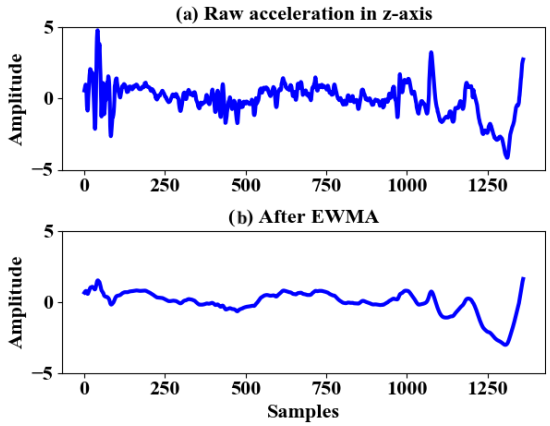


Fig. 2. Example of data smoothing.

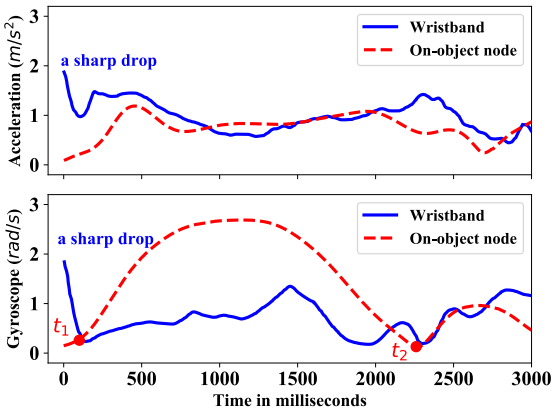


Fig. 3. Motion data due to a door opening. After t_2, the user's hand is off the door handle.

of the objects). Second, similar to an access control system, each object should have a list of legitimate users (e.g., when an office room is shared by two users). When an object is used, the IoT server only pulls data from the wristbands in the list. In a large organization, assuming the wristbands get connected to an WiFi access point near an object automatically when they approach the object, to further enhance the scalability, the IoT server only pulls data from the legitimate users' wristbands that are currently connected to that access point.

3.2.4 Limitations. MoMatch exploits motion data for authentication. If a server is used in the deployment and it gets compromised, the user privacy leaked by motion data will become a concern. This limitation is shared by many prior works [24, 52, 55] that make use of motion data for authentication. The concern may be mitigated by deploying the trained neural network in the wristband, if the computation resources and application scenarios allow. How to run deep learning models in a resource-constrained computer is an actively studied topic [5]. Another possible solution is to leverage techniques proposed for privacy-preserving authentication [4].

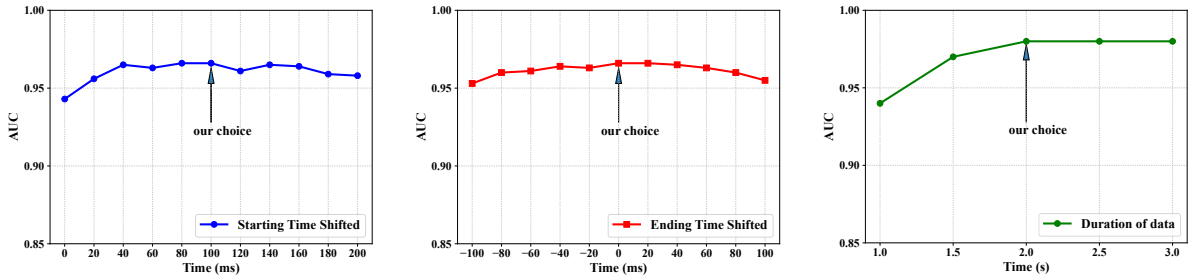
4 DATA PRE-PROCESSING

4.1 Data Representation and Smoothing

Accelerometers and gyroscopes are among the most common sensors. An accelerometer measures positional acceleration, and a gyroscope angular velocity. Both provide three-axis measurements. For example, the acceleration data $[a_x, a_y, a_z]$ indicates the accelerations along the axes x, y, and z, respectively.

People may operate an object from different orientations. Rather than comparing the data on each axis separately, we propose to use the square root values of the data: the acceleration is calculated as $a = \sqrt{a_x^2 + a_y^2 + a_z^2}$, and the gyroscope as $g = \sqrt{g_x^2 + g_y^2 + g_z^2}$. The resulting data can reflect the current status of a motion in a reliable way regardless of the orientation of the object user.

The raw sensor data usually contain high-frequency noises. To reduce the effect of noises, we apply a low pass filter, called *Exponentially Weighted Moving Average* (EWMA) [54]. As an example, Figure 2 shows the effect of EWMA when applied to the z-axis raw acceleration data from a smartwatch. After smoothing, most high-frequency noises are removed and the important features such as significant valleys and crests are preserved.



(a) Selecting the start time t_1.

(b) Shifting the end time t_2 (*fixed-motion* objects).

(c) Selecting the time duration t_2 - t_1 (*free-motion* objects).

Fig. 4. Evaluating the selections of the start and end time.

4.2 Data Interval Selection

An object use involves a sequence of actions. For instance, when opening a door, a user first places her hand on the door handle, turns it, and then pushes the door. After that, the action will be various—some may hold the door shortly, while others may leave the door alone and, in the latter case, the movements of the hand and the door become uncorrelated. It will confuse the machine learning model if the data of the *uncorrelated* duration is used for training and, more critically, require more training data to get a converged model. We thus conduct *data interval selection.*

4.2.1 Selecting the start time t_1. Our observation is that, for a very short duration after an object motion is detected, the sensor data of the wristband and that of the sensor node are usually different. E.g., when a user *raises* her hand and *moves* it towards an object with a certain speed, the first few milliseconds of both the acceleration and gyroscope values of the wristband are *not* zero, while the data of the on-object node starts from near zero. In Figure 3, starting at Time=0 the acceleration from the wristband drops sharply from $1.9m/s^2$ and gyroscope from $1.8rad/s$, while the data from the on-object node increases from near zero. Such differences may mislead the model training. To determine how much uncorrelated data needs to be discarded at the beginning of an operation, we evaluate the performance by changing the start time t_1 while fixing the end time t_2. We compute the average AUC using the proposed S-CNN model (see Section 5.1). Figure 4 (a) shows that the AUC increases significantly when the first $40ms$ data is discarded and starts to drop when the discarded data is longer than $140ms$. While our design adopts $100ms$, the AUC keeps relatively steady when the shift time is between $40ms$ and $140ms$.

4.2.2 Selecting the end time t_2. The methods differ for fixed-motion objects and free-motion ones.

- For fixed-motion objects, the end time of an operation can be determined according to the property of the motion. Specifically, (1) For fixed-motion objects that *rotate along a fixed pivot* (i. e., doors), we use the gyroscope data of the on-object node to determine the end time, which is the *second valley point* in the *gyroscope* data; according to our observation, it consistently means that the user has completed the pushing/pulling operation. Figure 3 shows an example of this class, where t_2 is the end time. (2) For fixed-motion objects that *do not have a fixed pivot* (i. e., drawers), the gyroscope data is close to zero all the time. We instead use the acceleration data of the on-object node to determine the end time, which should satisfy the following constraints. First, it should correspond to a valley; second, the value at the valley should be very small (≤ 0.4). Our experiments show that this corresponds to the end of an activity on such objects. We further conduct an experiment to validate the determination of the end time. We investigate how the performance changes as we shift the end time t_2 selected using the above strategies. Figure 4 (b) shows that the average AUC slightly decreases if we shift t_2 more than 40ms. The experiment confirms that

the above strategies are able to select an optimal t_2, although minor shifting does not result in an obvious change in the model performance.

- For free-motion objects, such as guns and dumbbells, a user may hold it for a while; we thus need to determine the time length for authentication. We test the performance of our model on the first 1, 1.5, 2, 2.5 and 3 seconds sensor data, respectively. Figure 4 (c) shows the AUC. The largest improvement is observed when the time is increased from 1 to 2 seconds, and there is no noticeable improvement when it is increased from 2 to 3 seconds. Thus, we adopt 2 seconds to save the authentication time.

5 CORRELATION CALCULATION

5.1 Design Choices

To calculate the correlation accurately (low false acceptance rate and low false rejection rate), our design started with the following three techniques, which are frequently used for calculating the correlation of two signals.

(1) *Correlation.* The correlation function is used to assess the temporal similarities of two sequences [35]. Pearson Product-Moment Correlation Coefficient [13] is effective in summarizing the linear relation of sequences.

(2) *Coherence.* It examines the similarity of two signals in the frequency domain [77]. For example, Welch's method [88] computes the magnitude squared coherence of the power spectral density of two signals.

(3) *Dynamic Time Warping (DTW).* Via dynamic programming, DTW [64] finds an optimal non-linear alignment that minimizes the distance of two sequences, and calculates the similarity score based on the distance.

We implemented and evaluated the three designs, but the performance was poor (see Section 8.2). They each work well in capturing a certain correlation: specifically, the Pearson ***Correlation*** function performs well if one signal can be converted to the other via linear transformation, the ***Coherence*** outputs a high score if two signal are similar in the frequency domain, and ***DTW*** works well if one signal can be *"warped"* to get another. However, data in our system, such as the example shown in Figure 3, does not fall in any of the cases.

(4) *SAPHE.* The approach proposed in SAPHE [34] was used for device pairing. We adapt it to compare two sequences of sensor data. Specifically, it first generates a sequence of random values and compares them to the corresponding values from the two pieces of sensor data. Then, the hamming distance is computed using the two sequences of comparing results. A closer hamming distance indicates a higher similarity. SAPHE also considers similar motion data, e.g., by shaking two devices together.

(5) *ShakeUnlock.* We implement the approach proposed in ShakeUnlock [24]. It requires the user to wear a smartwatch and hold another mobile device in the same hand, and to shake them together. In their case, the two devices have almost the same motions; thus, their acceleration data looks very similar. Their approach is based on coherence and obtains coherence vectors to determine a similarity score.

We then leveraged various machine learning techniques to resolve the problem.

(6) *SVM.* Given two motion data sequences, we construct a feature vector for each, and then compute the L1 distance of the two feature vectors to train an SVM, which detects whether the two sequences correlate. We consider totally 43 features: 42 features from the famous work [50], which are used for activity recognition and are able to describe key characteristics of motion data due to activities, and the DTW distance as the 43rd one.

(7) *S-LSTM.* Recurrent Neural Networks (RNNs) are known to be good at handling time series. We employ long short-term memory (LSTM), a type of RNNs specialized in learning long-term dependencies. As shown in Figure 5, we use the Siamese architecture [8], which performs very well in comparison, and the design is denoted as *S-LSTM*. The inputs are two data sequences, $X^{(a)} = (x_1^{(a)}, ..., x_n^{(a)})$, and $X^{(b)} = (x_1^{(b)}, ..., x_n^{(b)})$. An LSTM cell analyzes an input value coming from either the input sequence or the precedent step and updates its hidden state at each time step (the reader is referred to [41] for more details). The outputs, $h^{(a)}$ and $h^{(b)}$, at the last time step

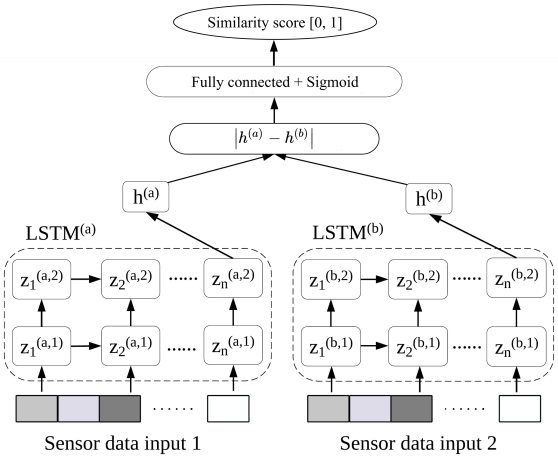


Fig. 5. The network architecture of S-LSTM.

in the second hidden layer are used to calculate the element-wise absolute difference $\left|h^{(a)} - h^{(b)}\right|$, which is fed to a fully-connected layer plus a sigmoid unit to calculate the similarity score.

(8) *S-CNN.* Researchers have found that CNNs work well for resolving time series problems [94]. Inspired by it, we propose to combine CNNs and the Siamese architecture, denoted as ***S-CNN***, to compare two time series. We evaluate both 1D CNNs and 2D CNNs for this purpose, denoted as S-CNN (1D) and S-CNN (2D), respectively. The key difference is that the convolutional kernel in 1D CNN moves in one direction, while that in 2D CNN moves in two directions, allowing it to extract rich features from the data. While it is straightforward to feed a time series into a 1D CNN [46, 47], how to convert time series into 2D images is described in Section 5.2.

5.2 Imagified Curve Comparison

We convert the correlation calculation problem into a comparison problem: if two motion-data sequences *correlate*, they are regarded as *similar*; otherwise, *dissimilar*. Furthermore, we propose *Imagified Curve Comparison* (ICC), which converts motion data sequences into images and employs Convolutional Neural Networks (CNNs) to automatically learn important features from those images for subsequent comparison. We use the Siamese architecture [8], which performs very well in comparison, and the design is denoted as *S-CNN*. Instead of classifying the motion data according to users' templates as in SenseTribute [38], the comparison-oriented idea is able to capture the ***uniqueness*** in each object use; that is, any unique detail in an object use is reflected by motion data on both sides. Moreover, it does not need per-user profiling, meaning once trained the model can be used to authenticate users that are not seen during training. Figure 6 shows the architecture, which contains twin CNNs each with three convolutional layers followed by a fully connected layer.

5.2.1 Data plotting and input layer. For each data sequence, we plot it with x-axis representing the time and y-axis the sensor value, and convert the sampling points into a *curve*. Thus the data from the sensor node (or wristband) is converted into two curves—one for acceleration data and another gyroscope data (if both acceleration and gyroscope data are used; Section 8.2 examines the model performance if only acceleration data is used). It is worth emphasizing that the ***realtimeness*** information is captured by the plotting along x-axis. Next, the two curves are combined to form an *image* (Section 8.3 explores two ways to combine them). We set the image size to be 100×75 (width $\times$ height) pixels. The width is chosen based on the sampling rate 50Hz (**P2** in Section 8.7) and the length of sensor data $\leq$ 2 seconds (**Data Interval Selection** in Section 4.2). Thus, the width is 100 ($= 50 \times 2$) pixels, and the height is chosen based on a common image aspect ratio (4:3).

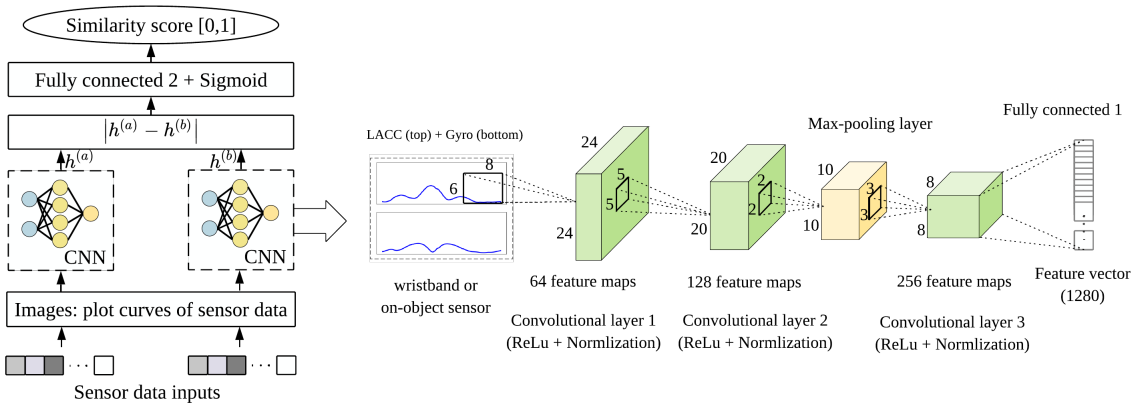


Fig. 6. The network architecture of S-CNN (2D).

5.2.2 Convolutional layers. We choose three layers considering the trade-off between performance and computational efficiency (**P3** in Section 8.7). The first layer applies 64 filters with a shape 8×6 and a stride of (4, 3), the second 128 filters with a shape 5×5 and a stride of 1, and the third 256 filters with a shape 3×3 and a stride of 1, respectively. The filter shape in the first layer is designed based on the aspect ratio (4:3) of input images. We opt for a larger stride of (4, 3) for the first layer filters as our input images are relatively sparse compared to complicated images. The filters with a shape 5×5 and 3×3 are commonly used and efficient in AlexNet [48] and GoogLeNet [80]. For the number of feature maps in each layer, we examine it by looping through the powers of 2 from 16 to 512 and choose the ones that perform the best. ReLUs are employed as activation functions.

5.2.3 Pooling layer. A max-pooling layer with a shape 2×2 and a stride of 2, is added after the second convolutional layer for downsampling the spatial dimensions and extracting shift-invariant features. We do not add a pooling layer after the first convolutional layer as its large filter stride already performs a sparse sampling of the data.

5.2.4 Fully connected layers. The fully connected layer flattens the feature maps from the convolutional layer into a feature vector. Our experiment shows an output of 1280 dimensions works best. The element-wise absolute difference of the two feature vectors (of the two Siamese sub-networks) is then converted into a similarity score $s \in [0, 1]$ using a fully connected layer plus a single sigmoid output unit.

5.2.5 Training and loss function. To learn the network parameters, we use standard backpropagation procedure to minimize the binary cross-entropy *loss* function:

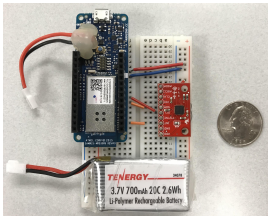
$$\mathcal{L} = -\sum_{i \in \mathcal{S}} \log s(M_1, M_2) - \sum_{j \in \mathcal{D}} \log(1 - s(M_1^{'}, M_2^{'})) \quad (1)$$

where M_1 and M_2 denote a pair of images converted from correlated sensor data sequences, and $M_1^{'}$ and $M_2^{'}$ an pair of images from uncorrelated data sequences. $\mathcal{S}$ and $\mathcal{D}$ represent datasets of correlated pairs and uncorrelated pairs, respectively. How to build the datasets $\mathcal{S}$ and $\mathcal{D}$ is detailed in Section 7.2.

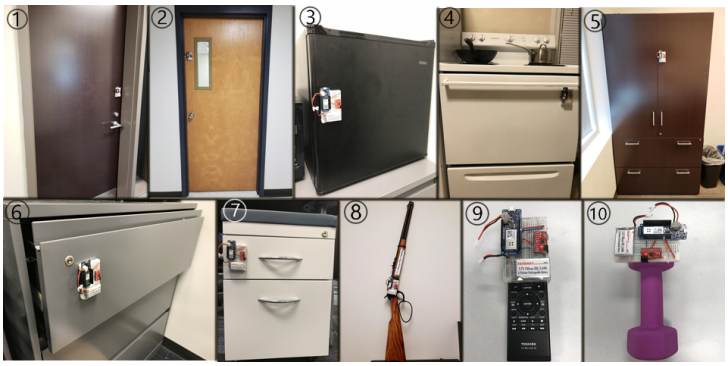
6 OTHER IMPLEMENTATION DETAILS

Wristband. We implement an application for collecting the sensor data on the LG W200 smartwatch that runs the latest Android Wear 2.0. The smartwatch has a Bosch BMI160 inertial measurement unit embedded with a triple-axis accelerometer and a triple-axis gyroscope.

On-object sensor nodes. As shown in Figure 7a, the on-object node uses an InvenSense MPU-6050, which integrates an accelerometer and a gyroscope. We use an Arduino MKR1000 or Arduino Pro Mini 3.3V to interface with MPU-6050= via I^2C bus. Mass production cost for such a unit is around $10.



(a) A sensor node compared to a US quarter coin



(b) Ten objects used in our experiments

Fig. 7. Ten objects in (b) include *seven fixed-motion* objects (two room doors labeled as 1 and 2, where Door I is a right swing door and Door II a left swing door that automatically closes the door after opening; a fridge door labeled as 3; an oven door labeled as 4; a cabinet door labeled as 5; and two drawers labeled as 6 and 7, where Drawer I does not have handles and Drawer II does), and *three free-motion* objects (a replica shotgun labeled as 8; a remote labeled as 9; and a 5lb dumbbell labeled as 10).

Table 2. Statistics about participants and activities.

Object category	Name	User #	Activity #
Fixed-motion objects	Room door I	13	1284
	Room door II	17	1788
	Fridge door	20	2129
	Oven door	12	1295
	Cabinet door	13	1408
	Drawer I	10	1062
	Drawer II	20	2108
Free-motion objects	Shotgun	15	1596
	Remote	19	2107
	Dumbbell	15	1651

Authentication process. To facilitate large-scale data collection and experiments, we run the authentication process in a desktop with an Intel i7-6700 processor and 12GB RAM memory. The process performs these main functions: 1) maintaining secured connection with wristbands and on-object nodes; 2) pre-processing sensor data and conducting the authentication; and 3) notifying the results.

7 DATA COLLECTION

7.1 Objects and Participants

Totally ten (10) objects are used in the experiments, as shown in Figure 7b. We obtained the IRB approval for the study. We recruited 27 volunteers (15 male and 12 female) by advertising about the study on our university campus through emails and posters. Out of the 27 subjects, 19 are graduate students with ages ranging from 20 to 35 years, 6 are undergraduate students with ages ranging from 20 to 25 years, and 2 are local residents with ages ranging from 40 to 55 years. None of the subjects have a computer security background. Note that most prior works invited no more than 10 participants [38, 42, 71]. More importantly, our evaluation (**P1** in Section 8.7) shows 8 users are sufficient to train the neural network model. We randomly assign 5 objects to each participant

and ask them to operate on the objects using the hand wearing the smartwatch, e. g., opening a door, picking up a gun, etc. For each object, they are asked to perform the activity for 100+ times, and are free to choose their ways of performing the activities. Table 2 shows the number of participants and activities for each object.

7.2 Building Dataset

We follow a conventional way of building positive and negative pairs for training a Siamese network [67, 97].

Correlated sensor data pairs. Whenever a participant performs an operation on an object, we collect one correlated data pair from the smartwatch and the on-object sensor node. Our dataset contains 16,428 correlated pairs, each with a label $s = 1$.

Uncorrelated sensor data pairs. Given any two *different* randomly selected activity instances, where u_1 operates on o_1, and u_2 on o_2, we denote the sensor data collected from u_1's and u_2's smartwatches as s_{u_1} and s_{u_2}, and the data from the on-object nodes as s_{o_1} and s_{o_2}. We can construct two uncorrelated pairs, $<s_{u_1}, s_{o_2}>$ and $<s_{u_2}, s_{o_1}>$. However, the time intervals of the two sequences in an uncorrelated pair may be different. We thus truncate the longer one, such that the two sequences have the same interval. To make them balanced with the correlated ones, we generate 16,428 uncorrelated pairs, each with a label $s = 0$.

8 EVALUATION

We first compare the design choices listed in Section 5.1 in terms of ***accuracy*** and ***generalization***. Specifically, we seek to understand: (**C1**) which method achieves the *highest accuracy*, and (**C2**) which one is able to train a *unified model* effective across the various objects.

We next evaluate our model in terms of ***accuracy***, ***adaptability***, ***efficiency*** and ***resilience to mimicry attacks***. We first examine (**Q1**) whether the *accuracy* can be improved by combining *both* the acceleration and gyroscope data. Regarding ***adaptability***, we seek to understand (**Q2**) whether it keeps a high accuracy when users are tested again after a long time; and (**Q3**) whether it keeps accurate for ***unseen users***, that is, users that are not seen during training. To evaluate the ***efficiency***, we measure (**Q4**) the response time for `MoMatch` to make a decision, and the computation resource for running our neural network model; (**Q5**) the time used for training our neural network model; and (**Q6**) the energy consumption of our application on the smartwatch. Moreover, we evaluate its ***resilience to mimicry attacks*** (**Q7**).

We finally study the parameter choices and the effect of hyperparameters. (**P1**) How large should the training dataset be in order to attain high accuracy? (**P2**) What is the appropriate sampling rate that yields the best trade-off between accuracy and efficiency? (**P3**) How do the hyperparameters affect the model performance?

8.1 Evaluation Methodology

For the five methods—correlation, coherence, DTW, SAPHE [34], and ShakeUnlock [24]—we calculate their statistic values for each pair of sensor data in our dataset. We use *Receiver Operating Characteristic* (ROC) curve which plots the True Acceptance Rate (TAR) against the False Acceptance Rate (FAR) as the threshold value varies, to determine the optimal threshold that leads to *Equal Error Rate* (EER), and use *Area Under the Curve* (AUC) to evaluate their performance. As our correlated dataset and uncorrelated dataset have the same size, the overall accuracy is derived as ***Accuracy* = 1 - EER**. An ideal authentication system can correctly distinguish legal uses from illegal uses all the time, and thus possesses EER = 0 and AUC = 1.0.

To evaluate SVM, S-LSTM and S-CNN, we adopt the strict methodology of *Leave-One-Subject-Out (LOSO) cross validation*, which is widely used to evaluate authentication systems [21]. We run LOSO where each time one subject is used for testing only (that is, ***unseen users***) and the other participants for training. We iterate the process to test our models over all participants and report the average AUC and EER, which can demonstrate

Table 3. Hyper-parameters of S-LSTM and S-CNN, and the ranges of their values explored in our experiments.

Model	**S-LSTM**	**S-CNN**
Layer #	$1 \sim 5$	$1 \sim 5$
Hidden dimension or # of units in FC	$16 \sim 512$	$128 \sim 2048$
Learning rate	$10^{-4} \sim 10^{-1}$	$10^{-4} \sim 10^{-1}$
Decay rate	$10^{-5} \sim 10^{-3}$	$10^{-5} \sim 10^{-3}$
Max epoch #	200	200
Optimizer	SGD / Adam	SGD / Adam
Loss function	MSE / BCE	MSE / BCE

Table 4. Top 5 selected features with the highest fisher score. (F_2: average acceleration in z axis; F_3: standard deviation in x axis; F_4: standard deviation in y axis; F_6: average absolute difference in x axis; F_7: average absolute difference in y axis; F_8: average absolute difference in z axis; F_9: average resultant acceleration; F_{43}: DTW distance)

Drawer I		**Drawer II**	
Features	Fisher score	Features	Fisher score
F_9	0.107	F_9	0.207
F_6	0.084	F_{43}	0.203
F_3	0.083	F_2	0.146
F_{43}	0.081	F_7	0.144
F_8	0.073	F_4	0.119

how it works for *unseen users* (so it can answer **Q3**). Some prior authentication systems, such as ZEBRA [55], SAW [56], and P2Auth [52], also use the LOSO methodology.

To train the SVM models, we try two different kernels: *Gaussian radial basis function* (RBF) and linear kernel. We vary the penalty parameter C from 0.001 to 1000, and the kernel parameter gamma in RBF from 0.0001 to 10. The values with the best accuracy are selected. We perform tedious feature selection for each object. We use fisher score [19] to find the most effective features from the total 43 features (Section 5.1). A higher fisher score indicates that the features have a better ability to separate data from different classes. Based on the score ranking, we add the feature that increases the accuracy until no feature improves the performance any more. In the end, we obtain different feature sets for different objects to achieve the best performance. As an example, Table 4 shows the top 5 selected features for the two objects, Drawer I and Drawer II. The result shows that a feature, which is important for one object, has little effect on the other even if the two objects seem similar. This shows that *much tedious effort is required to select the best features for each object when applying SVM.* We normalize the feature distances, and use the library Scikit-learn [66] to build an SVM that implements the Sequential Minimal Optimization algorithm [22].

To train models for S-LSTM and S-CNN, we investigate the hyperparameters showed in Table 3. We initialize the network weights for S-CNN and S-LSTM using He's initialization approach [40] and small random Guassian entries, respectively. We choose the batch size of 128 for S-CNN, and 64 for S-LSTM. A smaller batch size for LSTM can help reduce the length variance of inputs. After each epoch, we measure the AUC and loss on the validation dataset, and save the model that achieves the best AUC as the best model. Early stopping is applied once the AUC has no improvement in 10 consecutive epochs.

Table 5. Performance comparison for models *using acceleration data*. O_1: Door I; O_2: Door II; O_3: Fridge; O_4: Oven; O_5: Cabinet; O_6: Drawer I; O_7: Drawer II; O_8: Gun; O_9: Remote; O_{10}: Dumbbell. ("**Separate models**" means that, for each design, a separate model is trained for each type of objects; "**Unified models**" means that a unified model is trained for all the 10 objects.)

Model	Metric	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	**Avg.**
Separate Models												
Coherence	EER	0.45	0.39	0.24	0.53	0.35	0.43	0.28	0.47	0.22	0.34	0.370
	AUC	0.56	0.64	0.84	0.46	0.69	0.60	0.78	0.53	0.85	0.73	0.668
Correlation	EER	0.38	0.33	0.25	0.48	0.24	0.37	0.22	0.48	0.23	0.29	0.327
	AUC	0.68	0.73	0.83	0.53	0.83	0.67	0.85	0.55	0.85	0.78	0.730
DTW	EER	0.15	0.31	0.23	0.31	0.15	0.24	0.19	0.35	0.27	0.48	0.268
	AUC	0.91	0.74	0.86	0.74	0.92	0.84	0.90	0.69	0.78	0.62	0.800
SAPHE [34]	EER	0.29	0.27	0.26	0.40	0.28	0.30	0.19	0.32	0.22	0.32	0.285
	AUC	0.78	0.79	0.81	0.63	0.79	0.76	0.87	0.73	0.86	0.74	0.776
ShakeUnlock [24]	EER	0.25	0.28	0.23	0.32	0.24	0.23	0.19	0.36	0.21	0.28	0.259
	AUC	0.83	0.78	0.87	0.71	0.85	0.86	0.91	0.67	0.88	0.78	0.814
SVM	EER	0.15	0.26	0.13	0.29	0.14	0.24	0.14	0.25	0.13	0.27	0.200
	AUC	0.92	0.82	0.94	0.76	0.94	0.85	0.93	0.83	0.94	0.79	0.872
S-LSTM	EER	0.11	0.17	0.15	0.17	0.23	0.24	0.11	0.21	0.16	0.19	0.174
	AUC	0.95	0.91	0.93	0.91	0.87	0.85	**0.96**	0.89	0.90	0.87	0.904
S-CNN (1D)	EER	0.10	0.08	0.19	0.21	0.24	0.22	0.09	0.15	0.17	0.33	0.178
	AUC	**0.96**	**0.96**	0.89	**0.92**	0.83	**0.90**	**0.96**	**0.93**	0.90	0.70	0.895
S-CNN (2D)	EER	0.16	0.15	0.12	0.17	0.10	0.18	0.11	0.15	0.12	0.15	0.141
	AUC	0.90	0.93	**0.95**	0.91	**0.97**	**0.90**	**0.96**	**0.93**	**0.95**	**0.92**	**0.932**
Unified Models												
SVM	EER	0.24	0.27	0.20	0.39	0.19	0.30	0.22	0.33	0.16	0.31	0.261
	AUC	0.84	0.80	0.87	0.64	0.89	0.76	0.85	0.72	0.90	0.76	0.803
S-LSTM	EER	0.12	0.17	0.20	0.17	0.29	0.21	0.12	0.24	0.20	0.25	0.197
	AUC	**0.94**	**0.91**	0.89	**0.91**	0.81	0.86	0.94	0.84	0.88	0.83	0.881
S-CNN (1D)	EER	0.15	0.32	0.15	0.21	0.20	0.22	0.04	0.13	0.13	0.36	0.191
	AUC	0.92	0.74	0.92	0.83	0.89	0.85	**0.98**	0.92	**0.95**	0.75	0.875
S-CNN (2D)	EER	0.14	0.20	0.15	0.18	0.13	0.18	0.12	0.16	0.14	0.15	0.155
	AUC	0.93	0.89	**0.92**	0.90	**0.95**	**0.90**	0.94	**0.93**	0.94	**0.93**	**0.923**

8.2 Performance Comparison between Design Choices

8.2.1 Answer to C1 (which method achieves the highest accuracy). We first train a *separate* model for each object using the *acceleration* data alone. The AUC and EER of each model for each method are showed in the rows 3-9 in Table 5. We have the following observations. (1) ***Coherence*** performs the worst on average. (2) ***Correlation*** is slightly better but poor overall. (3) The performance of ***DTW*** is not stable—it has good accuracies on some objects, but not on others, such as oven, dumbbell, and gun. (4) The average AUC achieved by ***SAPHE*** [34] is 0.776, worse than DTW. (5) ***ShakeUnlock*** [24] has slightly better performance than DTW on average, but it is significantly worse than any of our machine learning based approaches. ShakeUnlock needs to carefully select the coherence frequencies for different tasks. The performance is also susceptible to coherence frequency weighting and the choices of coherence thresholds. While SAPHE and ShakeUnlock attain high accuracies for handling very similar motion data from two devices that are shaken together [24, 34], they do not perform well for data due to natural object uses in our case. (6) ***SVM*** attains the average AUC=0.872, but its performance is still

Table 6. Performance comparison for separate models **with sensor data fusion**. "Fusion I" and "Fusion II" are interpreted in Section 8.3. The calculation of "Opt. AUC" assumes the best model for each object is used.

Model	Metric	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	**Avg**	**Opt. AUC**
S-LSTM	EER	0.05	0.01	0.05	0.12	0.18	0.05	0.10	0.08	0.20	0.16	0.100	
	AUC	0.97	0.99	0.96	0.95	0.90	**0.97**	0.95	0.97	0.88	0.92	0.946	
S-CNN (1D)	EER	0.02	0.00	0.08	0.13	0.15	0.22	0.01	0.15	0.20	0.23	0.119	
	AUC	**0.99**	**1.00**	0.96	0.95	0.91	0.88	**0.99**	0.94	0.87	0.85	0.934	
S-CNN (2D)	EER	0.10	0.08	0.07	0.11	0.15	0.10	0.07	0.11	0.08	0.04	0.091	**0.984**
(Fusion I)	AUC	0.96	0.97	0.97	0.96	0.94	0.95	0.97	0.95	0.96	0.98	0.961	
S-CNN (2D)	EER	0.06	0.12	0.09	0.04	0.11	0.12	0.02	0.03	0.08	0.01	0.068	
(Fusion II)	AUC	0.98	0.97	**0.98**	**0.99**	**0.97**	0.95	**0.99**	**0.99**	**0.97**	**0.99**	**0.978**	

Table 7. Performance comparison for unified models **with sensor data fusion**.

Model	Metric	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	**Avg.**
S-LSTM	EER	0.06	0.11	0.08	0.16	0.20	0.15	0.10	0.12	0.22	0.22	0.142
	AUC	**0.97**	0.94	0.95	0.91	0.87	0.92	0.95	0.93	0.87	0.88	0.919
S-CNN (1D)	EER	0.14	0.04	0.13	0.20	0.18	0.07	0.08	0.15	0.16	0.33	0.148
	AUC	0.92	**0.99**	0.94	0.83	0.90	**0.99**	**0.97**	0.92	0.92	0.79	0.917
S-CNN (2D)	EER	0.12	0.09	0.10	0.14	0.11	0.14	0.10	0.18	0.12	0.13	0.123
(Fusion I)	AUC	0.94	0.97	0.96	0.94	0.95	0.93	0.96	0.91	0.94	0.94	0.944
S-CNN (2D)	EER	0.11	0.13	0.08	0.07	0.06	0.12	0.09	0.08	0.09	0.03	0.086
(Fusion II)	AUC	0.96	0.95	**0.97**	**0.98**	**0.98**	0.95	**0.97**	**0.98**	**0.97**	**0.99**	**0.970**

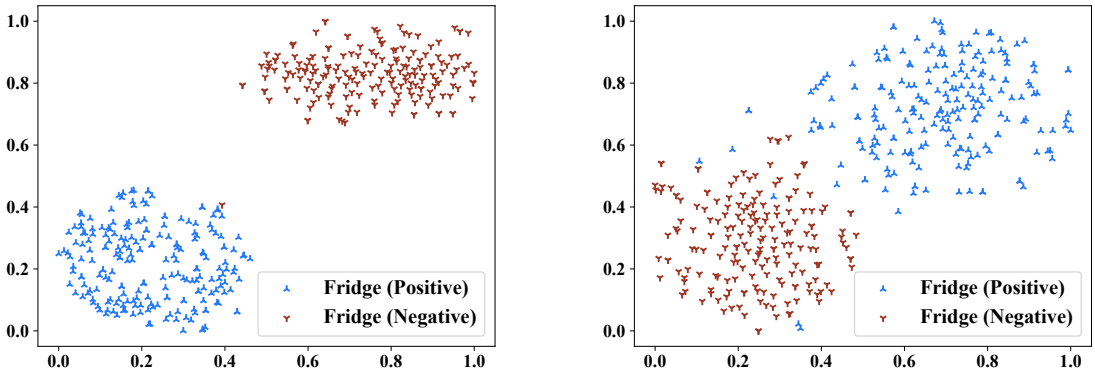
unstable across the ten objects. (7) ***S-LSTM*** achieves better performance than SVM on average. (8) ***S-CNN (2D)*** outperforms all the other methods regarding the ***average AUC***, which is 0.932, significantly higher than 0.904 with S-LSTM and 0.895 with S-CNN (1D). Its performance keeps relatively stable across all the objects. This can be attributed to the ICC technique, which captures rich information for correlation evaluation. (9) For objects, such as Door I, Door II and Oven, the S-CNN (1D) models show the highest accuracy. Thus, if a separate model is deployed for each type of objects, we can select the better model between S-CNN (1D) and S-CNN (2D). (10) While S-CNN performs better than SVM in our case, we do not imply that deep learning is inherently better than traditional machine learning. However, compared to *SVM*, *S-CNN* does not need feature engineering.

For S-LSTM and S-CNN, we also train a *separate* model for each object with sensor fusion (explained in Section 8.3). As shown in Table 6, the average performance of all the models gets improved, and S-CNN (2D) still outperforms others. By selecting the best model for each object, `MoMatch` achieves an average AUC=**0.984**.

8.2.2 Answer to C2 (which method can train a unified model effective across all the objects). If many different types of objects have to be considered, deploying a *separate* model for each type of objects in a resource-constrained environment, such as an IoT hub, is a burden. Thus, a *unified* model is desirable. This experiment compares the performance of SVM, S-LSTM and S-CNN by training such a unified model for each design. This time, we perform feature selection for SVM by considering the best average accuracy for all the objects.

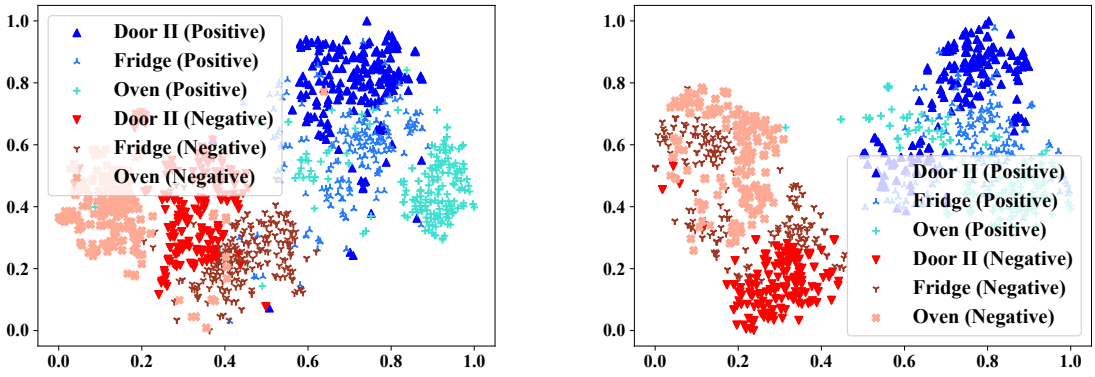
The AUC and EER of the unified model for each method are shown in Table 5. We can observe that S-CNN and S-LSTM outperform SVM significantly. Note for SVM, it is challenging to find a unified set of feature that keeps working well across objects. S-CNN (2D) degrades slightly from the average AUC= 0.932 (separate models) to AUC = 0.923 (unified model), which is significantly higher than 0.881 with S-LSTM and 0.875 with S-CNN (1D). Note that the final accuracy of a unified S-CNN (2D) model with sensor fusion is AUC = 0.970 (Section 8.3).

We use t-Distributed Stochastic Neighbor Embedding (t-SNE) [84], a technique for visualizing high dimensional features, to plot feature embeddings of the separate model trained for the fridge in Figure 8 (a) and the unified



(a) The separate model. (b) The unified model.

Fig. 8. Feature visualization of the separate model and unified model; the data of the *fridge* is used as an example.



(a) Using acceleration data only. (b) Sensor fusion with acceleration and gyroscope data.

Fig. 9. Feature visualization of models with acceleration data alone vs. sensor fusion.

model in Figure 8 (b). The positive and negative pairs in Figure 8 (a) are more distinguishable than those shown in Figure 8 (b), which explains the slight drop in accuracy when a unified model is used.

In short, S-CNN (2D) outperforms the other designs in terms of both accuracy and generalization. Thus, in the following, we use a *unified* S-CNN (2D) model for all the 10 objects unless otherwise specified.

8.3 Sensor Data Fusion

8.3.1 Answer to Q1 (whether the accuracy can be improved by using both the acceleration and gyroscope data (i.e., sensor fusion)). We explore two ways to apply sensor fusion to on S-CNN (2D): (Fusion I) the two curves are plotted overlapped using the same x axis; (Fusion II) the two curves are plotted in one image but placed at the top half and bottom half, separately (as shown in Figure 6).

As shown in Table 6, for separate S-CNN(2D) models, compared with using acceleration data alone (AUC = 0.932), both Fusion I (AUC = 0.961) and Fusion II (**AUC = 0.978**) help improve the performance. Table 7 also show that AUCs of Fusion II (**AUC = 0.970**) and Fusion I (AUC = 0.944) are improved over using acceleration data

Table 8. Four parts of response time.

Name	Value
Activity duration	around 2 s
Transferring 2s data	smartwatch: 159.7±26.3ms sensor node: 96.9±5.1ms
Pre-processing 2s data	113.8±8.0 ms
Time for running model on an image pair	Intel i7-6700: 19.2±4.1ms; or GeForce GTX 1080: 8.8±0.8ms

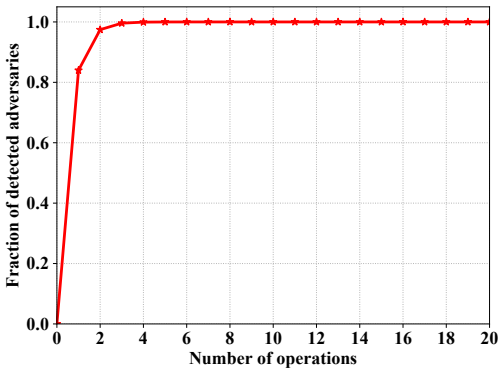


Fig. 10. Attacker detection rate.

alone. For both separate models and unified models, Fusion II outperforms Fusion I slightly. A possible reason is that when two curves are plotted separately, the model learns better how to weigh different regions of an image. We show the t-SNE embeddings of features generated by S-CNN (2D) with sensor data fusion II in Figure 9 (b). Compared to the features shown in Figure 9 (a), it is more distinguishable to separate positive and negative pairs in Figure 9 (b), which coincides with the results in Table 7.

We also apply sensor fusion to S-CNN (1D) by feeding the acceleration and gyroscope data to separate channels. The results shown in Table 6 indicate that the model S-CNN (1D) improves its performance through sensor fusion for separate models (acceleration alone: AUC = 0.895 vs. sensor fusion: AUC=0.934). Similarly, the results of unified models in Table 7 (acceleration alone: AUC = 0.875 vs. sensor fusion: AUC=0.917) also show the effectiveness of sensor fusion. Note S-CNN (2D) significantly outperforms S-CNN (1D) regarding the average AUC (separate models: 0.978 vs. 0.934; unified models: 0.970 vs. 0.917), demonstrating the advantage of using ICC.

8.4 Adaptability

8.4.1 Answer to Q2 (whether `MoMatch` *keeps accurate when users are tested again after a long time).* After two months, we asked each volunteer again to perform operations on each object for 10+ times, and create a new testing dataset. *Without retraining*, we directly measure the accuracy using the new testing dataset, and the accuracy has no observable changes.

8.4.2 Answer to Q3 (whether `MoMatch` *keeps accurate for unseen users).* This is already examined by the LOSO (Leave-One-Subject-Out) cross validation inherently (see Section 8.1), which always uses one participant not seen during training for testing.

8.5 Efficiency

8.5.1 Answer to Q4 (response time). We then evaluate the response time, which is defined as the time interval beginning at the time a user starts an operation and ending at the time `MoMatch` makes a decision. It mainly consists of the following four parts: (a) the duration of an activity; (b) the time used to transmit sensor data to the desktop; (c) the data pre-processing time; and (d) the time that the neural network model takes to compute the correlation score. Table 8 shows the measured time for each part. The total response time is less than 2.5 seconds. Thus, `MoMatch` makes a decision fast, in contrast to 50 seconds in ZEBRA [55].

Another important question is what computation resources are required for running our model. As shown in Table 8, when CPU (Intel i7-6700) is used, it takes around 19.2ms given an image pair. Although GPU (GeForce GTX 1080) can reduce the time to 8.8ms, the small difference can hardly be noticed by users, and thus does not validate the necessity of using GPUs.

Table 9. Power consumption of each component of the on-object sensor node.

Component	**Arduino Pro Mini 3.3 V**	**IMU**	**Bluetooth**	**Total**
Power Consumption (mA)	1.58	1.24	8.33 **(Active Mode)**	11.15
			0.17 **(Sleep Mode)**	2.99

8.5.2 Answer to Q5 (model training time). We use GPU to train the model. Our S-CNN (2D) model can achieve the best performance after 5-10 epochs. Each epoch contains 250 batches, and training each batch of images takes around 1.06 ± 0.4s on GPU. Thus, the training time is 44 min ($= 10 \times 250 \times 1.06$) at most.

By contrast, S-LSTM requires at least 80 epochs to achieve reasonable performance, each taking around 130s. So it totally requires about 173 min for training. The S-CNN (1D) model achieves its best performance after 20-30 epochs, each taking around 120s, which results in at least 40 min for training.

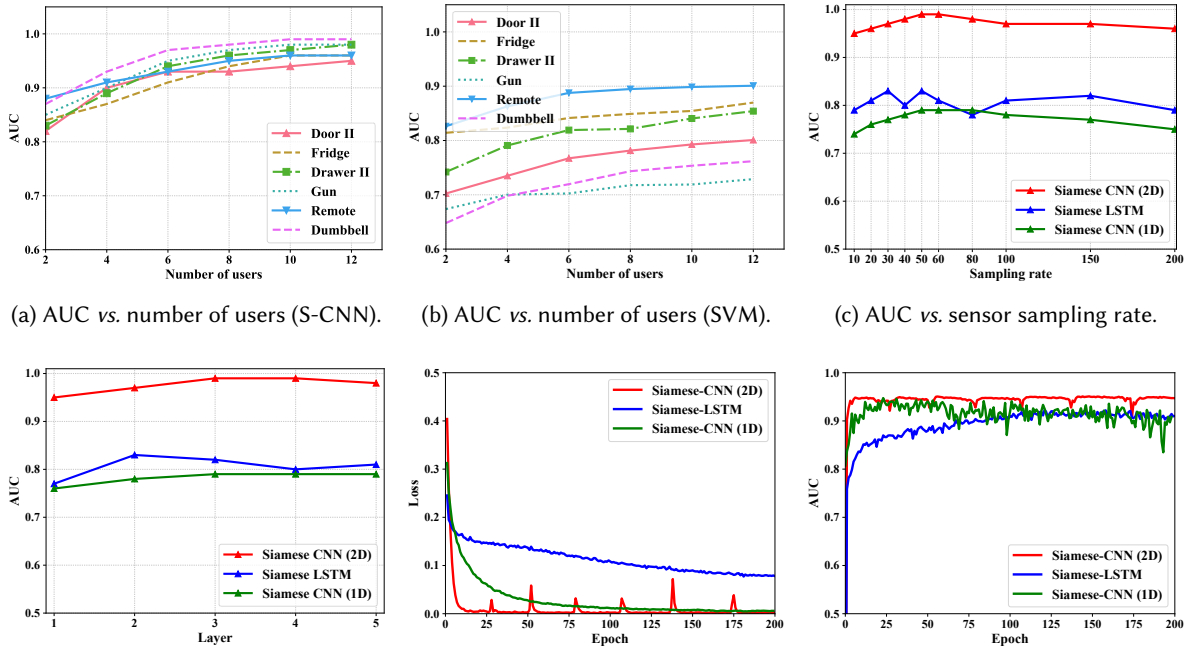
8.5.3 Answer to Q6 (energy consumption). We measure the energy consumption of our application on the smartwatch and the on-object sensor node. We let the smartwatch read 2s sensor data with a sampling rate of 50Hz and send it out once every minute for one hour to simulate the object use frequency of an average user. For comparison, we first measure the energy consumption when the screen is always on, which is 37.3mA per hour. The consumption increases to 47.4mA per hour when running our application while keeping the screen on. Thus, our application only consumes about 10.1mA per hour (<2% of the battery capacity). We let the on-object sensor node read 2s data from IMU MPU-6050 with a sampling rate of 50 Hz and send it out via a serial BLE module HM-10 continuously. We use a power monitor INA-219 to measure the power consumption. The average power consumption with Arduino MKR 1000 is 26.36 mA when the BLE module is in active mode and 18.20 mA in sleep mode. If our system is triggered for one minute each hour (20 operations if it takes 3s per operation), with a 700 mAh battery the on-object sensor node has a battery life of 38.2 hours. (Note Arduino MKR 1000 draws 16.79 mA and takes 92.25% of the power even when the Bluetooth is in sleep mode.) By comparison, as shown in Table 9, Arduino Pro Mini 3.3V draws 1.58 mA [17], and the total power consumption of the on-object sensor node is roughly 11.15 mA when the BLE module is in active mode and 2.99 mA in sleep mode, which extends the battery life to 223.9 hours. There are commercial low-power platforms which can further reduce power consumption of `MoMatch`. For exampel, the TI CC2652R [82] consumes 0.94 μA in standby mode, 3.4 mA in active mode, and 7.3 mA to transmit radio signals. The current consumption of Bosch IMU BMI270 [7] is just 30 μA. By leveraging these designs, the estimated battery life of a 700 mAh battery would be 129 days.

8.6 Security Analysis and Evaluation

To answer **Q7: resilience to mimicry attacks** described in our threat model (Section 3.2), we first perform security analysis and then present our empirical study results.

8.6.1 Security Analysis. A mimicry attacker $\mathcal{A}$ who wants to succeed has to make sure (1) his hand movement is similar to that of the victim $\mathcal{V}$ and (2) the movement should be synchronous with that of $\mathcal{V}$. However, studies have demonstrated that the average human reaction time is larger than 200ms [31, 44, 59]; such a time difference can be detected by our model, which captures the ***realtimeness*** of motion data. On the other hand, assuming an attacker *predicts* (rather than observing and mimicking) the action of $\mathcal{V}$ to launch an attack, although $\mathcal{A}$ may be able to align his attack-start time better with $\mathcal{V}$, it is difficult to ensure his hand movement is similar with that of $\mathcal{V}$ and meanwhile keeps synchronous throughout the object use.

8.6.2 Empirical Study. We have 10 participants act as victims and another 10 as attackers. We tell attackers the internals of `MoMatch` that it works based on motion correlation. We use two object types—the door and the gun—as the representative of fixed-motion objects and that of free-motion ones, respectively.



(a) AUC *vs.* number of users (S-CNN). (b) AUC *vs.* number of users (SVM). (c) AUC *vs.* sensor sampling rate.

(d) AUC *vs.* number of network layers. (e) Loss *vs.* number of epochs. (f) AUC *vs.* number of epochs.

Fig. 11. Impact of different hyperparameters and sensor sampling rate.

Our experiment contains two attack settings, as discussed in our **Threat Model** in Section 3.2. *Attack-I*: $\mathcal{V}$ keeps walking, while $\mathcal{A}$ watches and mimics $\mathcal{V}$'s hand motions to operate on an object. *Attack-II*: $\mathcal{V}$ operates on an object, and meanwhile $\mathcal{A}$ mimics $\mathcal{V}$ on another object of the same type. Each attacker performs attacks for 30 times on each object. For both *Attack I* and *Attack II*, we provide $\mathcal{A}$ with an ideal attack environment: (1) $\mathcal{A}$ has a clear view of $\mathcal{V}$; (2) $\mathcal{A}$ has enough time to observe $\mathcal{V}$'s operations; and (3) $\mathcal{V}$ *notifies* $\mathcal{A}$ *when starting an operation.*

Given *Attack-I* attacks, `MoMatch` can identify 99% of attacks based on one single operation of door II and 93% the gun. Given *Attack-II* attacks, `MoMatch` can detect 84% and 86% of attacks based on *one single* operation of door II and the gun, respectively. When an attacker uses multiple objects to achieve his attack purpose, it is straightforward to authenticate multiple operations to detect attackers more accurately by launching `MoMatch` for multiple times. This strategy is adopted by Zebra [55] as well. Fig. 10 shows the detection rate by considering multiple operations of objects, and the attacker detection rate reaches 96% after only 3 operations and near 100% after 4. In contrast, Zebra needs 84 interactions to recognize all attackers (with a grace period of 2).

8.7 Parameter and Hyperparameter Study

8.7.1 Answer to P1 (the size of training dataset needed). We examine the amount of data needed for training the S-CNN (2D) model and the SVM model. We randomly choose 2, 4, 6, 8, 10 and 12 users' data for each object to train both models, and use the rest data to test each trained model. This experiment uses the objects of door II, fridge, drawer II, gun, remote and dumbbell as examples.

Figure 11 (a) shows that S-CNN (2D) can obtain a reasonable accuracy even if the training dataset contains only 4 users' data; a dataset containing 8 users trains the S-CNN (2D) with satisfactory performance. Figure 11 (b)

shows that, even when the amount of data (e.g., 4 users' data) is small, SVM does not outperform S-CNN (2D) but actually shows a much lower accuracy. (Section 8.2 presents the accuracy comparison when all the users' data is used, also showing S-CNN significantly outperforms SVM.) While DNNs usually need more training data than traditional ML methods, our experiments show a different case. It can be attributed to the Siamese architecture, which is known to perform well in comparison tasks without needing a huge training dataset [8, 14, 97].

8.7.2 Answer to P2 (sampling rate). To study **P2** and **P3**, we take the dumbbell as an example (other objects show similar results). To find the optimal sampling rate, we examine it ranging from 5Hz to 200Hz. Figure 11c shows the impact of sampling rate on S-CNN (1D), S-CNN (2D) and S-LSTM. It can be observed that S-CNN (2D) performs well at 50Hz, and does not improve significantly when the rate increases further. We thus set the sampling rate as 50Hz. The fluctuation issue is mitigated in S-CNN compared to S-LSTM, showing *another advantage* of S-CNN.

8.7.3 Answer to P3 (hyperparameters). We next evaluate the impact of hyperparameters on S-CNN and S-LSTM (see Table 3). We vary (1) the number of convolutional layers in S-CNN and the number of hidden layers in S-LSTM. Figure 11d shows AUC reaches a high value when the number is increased to 3 for S-CNN and 2 for S-LSTM; no noticeable improvement arises when the number increases. We thus choose 3 for S-CNN and 2 for S-LSTM.

To examine the impact of (2) the number of training epochs, we train the three models for 200 epochs and record the loss and AUC after each epoch. The results are showed in Figure 11e and Figure 11f. For S-CNN (2D), the loss drops quickly and AUC reaches to the highest value after *only 6 epochs*, compared to 25 epochs in S-CNN (1D) over 100 epochs in S-LSTM.

We next investigate (3) the learning rate, (4) the decay rate, (5) the hidden dimension in S-LSTM and the number of units in the fully-connected layer in S-CNN, (6) two different optimizers, and (7) two different loss functions. We vary the learning rate from 0.0001 to 0.1 at logarithmic intervals, and find that a rate of 0.001 for both S-CNN and S-LSTM yields the best performance. The learning decay rate has a slight influence on S-CNN. The hidden dimension of 64 is the best for S-LSTM. The unit number between 1200 and 1400 in the fully-connected layer in S-CNN can obtain good performance, and we choose 1280. We also find *Adam* optimizer learns faster and works better than *SGD* for both S-CNN and S-LSTM. For the loss function, *Cross-Entropy* (CE) performs better than *Mean Squared Error* (MSE) for S-CNN, which is opposite for S-LSTM.

8.8 Comparison with Other Approaches

Our work achieves much higher accuracies than SenseTribute [38] and Hallmarks [71], which are based on behavioural biometrics. The accuracy of SenseTribute averages 74% with only 5 participants, and Hallmarks 63.8%. They both only consider fixed-motion objects, and *do not demonstrate whether their approaches can work on free-motion objects*. Moreover, *neither* examines its resilience to mimicry attacks.

However, we acknowledge that, in SenseTribute [38], sensor nodes are only attached to the objects, i.e., without requiring users to wear wristbands. Actually, a major reason why `MoMatch` outperforms existing systems, such as SenseTribute, is that it collects information from both sides. In cases when accuracies are not the most important goal, SenseTribute might be more usable than `MoMatch` (but note that SenseTribute needs per-user profiling).

9 USE CASES

To demonstrate the applications of `MoMatch`, we conduct two use cases in a real-world environment. In the first use case, `MoMatch` is used to record the uses of a dumbbell, showing an application to smart health. The other is to monitor the fridge door openings, which can be used for splitting energy bills in an apartment; this is an example frequently used in prior work [11, 70, 71].

Table 10. Performance for identifying users of a fridge and a dumbbell using our unified S-CNN (2D) model. (Legend: "GT" stands for Ground Truth based on the videos; "CA" indicates the number of object uses that are Correctly Assigned to the real user; "IA" indicates the number of uses that are Incorrectly Assigned to a user.)

User	Fridge door			Dumbbell		
	GT (times)	CA (times)	IA (times)	GT (times)	CA (times)	IA (times)
A	56	53	0	95	92	0
B	48	45	0	104	101	1
C	63	59	1	127	122	1

Both use cases are conducted in a three-bedroom apartment shared by three participants (all males with ages ranging from 25 to 35 years: $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$). The three volunteers are graduate students recruited from our university. Sensor nodes are attached to the dumbbell and the fridge door, and we ask participants to wear smartwatches. In order to establish the ground truth, we install a motion-activated camera in the public area that can monitor the object uses. Each time an object is operated, `MoMatch` collects the acceleration and gyroscope data from the three smartwatches and the on-object sensor node. Then, `MoMatch` computes the similarity score between each smartwatch and the sensor node. The smartwatch that achieves the highest similarity score that is above the threshold (Section 8.2) is identified as the object user.

The experiment lasts for one week. Table 10 shows the statistics collected. In total, `MoMatch` assigns 157 out of 167 fridge door openings (157/167=94.0%) and 315 out of 326 dumbbell lifts (315/326=96.6%) correctly to real users. Only 1 fridge door opening and 2 lift-offs are incorrectly assigned. For a very small number of object uses (9 for both the fridge and the dumbbell), they are not assigned to any of the participants due to the low correlation scores. The high performance shows that `MoMatch` is promising for enabling many applications, such as smart health and smart environment enhancement.

After the experiment, we invited the participants to give us feedback regarding `MoMatch`. Most of the comments show positive attitudes towards `MoMatch`. The advantages are mainly in the convenience because of implicit authentication, while a concern is about the large size of on-object sensors, which can be mitigated by using customized sensors. Representative comments include: "*The system is very useful to me as I don't need to count the number of lift-offs*" (Participant $\mathcal{A}$); "*The system is so natural that I forgot I was participating in an experiment for most of the time*" ($\mathcal{B}$); "*It would be better if the sensors are small enough to be attached to the objects*" ($\mathcal{B}$); "*It took me only a couple of days to get used to wearing a smartwatch, and I enjoyed its rich functions*" ($\mathcal{C}$).

10 CONCLUSION

It is an emerging trend that many traditional everyday objects, such as doors and windows, are retrofitted to smart environments by attaching inexpensive sensor nodes. We presented the *first* accurate implicit authentication approach for such retrofitted objects. Unlike prior work that requires intensive explicit object shaking, `MoMatch` conducts implicit authentication based on a single natural object use. An Imagified Curve Comparison (ICC) technique was proposed to convert motion data correlation measurement into an image comparison problem, which was well resolved using deep learning without profiling user biometrics. The average AUC across 10 various objects is 0.984. In addition to continuous monitoring, `MoMatch` can enable many other applications, such as personalization for smart environments, smart health, and forensics.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-1856380, CNS-2016415, CNS-2107093, CNS-2144669, CNS-1953073, CNS-1850278, CNS-2204785, and CNS-2205868. The authors would like to thank the anonymous reviewers.

REFERENCES

[1] Apple. 2021. Add a HomeKit accessory to the Home app. https://support.apple.com/en-us/HT204893.

[2] Apple Inc. 2019. Apple Watch. https://www.apple.com/watch/.

[3] Apple Inc. 2019. How to unlock your Mac with your Apple Watch. https://support.apple.com/en-us/HT206995.

[4] Abhilasha Bhargav-Spantzel, Anna C Squicciarini, Shimon Modi, Matthew Young, Elisa Bertino, and Stephen J Elliott. 2007. Privacy preserving multi-factor authentication with biometrics. *Journal of Computer Security* 15, 5 (2007), 529–560.

[5] Sourav Bhattacharya and Nicholas D Lane. 2016. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 1–6.

[6] Kemal Bicakci and Bulent Tavli. 2009. Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. *Computer Standards & Interfaces* 31, 5 (2009).

[7] Bosch Sensortec. 2022. Bosch launches smart ultra-low power IMU BMI270 optimized for wearables. https://www.bosch-sensortec.com/news/bosch-launches-smart-ultra-low-power-imu-bmi270.html.

[8] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "Siamese" time delay neural network. In *NIPS*.

[9] Gerard Canal, Sergio Escalera, and Cecilio Angulo. 2016. A real-time human-robot interaction system based on gestures for assistive scenarios. *Computer Vision and Image Understanding* 149 (2016).

[10] LLC Cao Gadgets. 2017. Wireless Sensor Tag System: Monitor Everything from the Internet.

[11] Yun Cheng, Kaifei Chen, Ben Zhang, Chieh-Jan Mike Liang, Xiaofan Jiang, and Feng Zhao. 2012. Accurate Real-Time Occupant Energy-Footprinting in Commercial Buildings. In *BuildSys*.

[12] Yushi Cheng, Xiaoyu Ji, Xiaopeng Li, Tianchen Zhang, Sharaf Malebary, Xianshan Qu, and Wenyuan Xu. 2020. Identifying child users via touchscreen interactions. *ACM Transactions on Sensor Networks (TOSN)* 16, 4 (2020), 1–25.

[13] Nian Shong Chok. 2010. Pearson's Versus Spearman's and Kendall's Correlation Coefficients for Continuous Data. (2010). http://d-scholarship.pitt.edu/8056/

[14] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*.

[15] Jason Cipriani. 2018. 13 new things you can do with your Android Wear smartwatch. https://www.cnet.com/how-to/tips-and-tricks-for-android-wear-2-0/.

[16] Mark D. Corner and Brian D. Noble. 2002. Zero-interaction Authentication. In *MobiCom*.

[17] Christopher David. 2020. Guide to reduce the Arduino Power Consumption. https://diyi0t.com/arduino-reduce-power-consumption/

[18] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. 2012. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *CHI*.

[19] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. *Pattern Classification (2Nd Edition)*. New York, NY, USA.

[20] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained Network Time Synchronization Using Reference Broadcasts. *SIGOPS Oper. Syst. Rev.* 36, SI (2002).

[21] Michael Esterman, Benjamin J Tamber-Rosenau, Yu-Chin Chiu, and Steven Yantis. 2010. Avoiding non-independence in fMRI data analysis: leave one subject out. *Neuroimage* 50, 2 (2010).

[22] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. 2005. Working Set Selection Using Second Order Information for Training Support Vector Machines. *J. Mach. Learn. Res.* 6 (2005).

[23] Sylvain Filiatrault and Ana-Maria Cretu. 2014. Human arm motion imitation by a humanoid robot. In *IEEE ROSE*.

[24] R. D. Findling, M. Muaaz, D. Hintze, and R. Mayrhofer. 2017. ShakeUnlock: Securely Transfer Authentication States Between Mobile Devices. *TMC* 16, 4 (2017).

[25] Forbes. 2018. Smartwatch Popularity Booms With Fitness Trackers On The Slide. https://www.forbes.com/sites/paullamkin/2018/02/22/smartwatch-popularity-booms-with-fitness-trackers-on-the-slide/#6929d3087d96.

[26] NFC Forum. 2018. NFC and Contactless Technologies. https://nfc-forum.org/what-is-nfc/about-the-technology/.

[27] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2011. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*.

[28] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 2010. Practical NFC Peer-to-peer Relay Attack Using Mobile Phones. In *RFIDSec*.

[29] Lishoy Francis, Gerhard P. Hancke, Keith Mayes, and Konstantinos Markantonakis. 2011. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *IACR Cryptology ePrint Archive* 2011 (2011).

[30] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. 2013. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *TIFS* 8, 1 (2013).

[31] T. P. Ghuntla, H. B. Mehta, P. A. Gokhale, and C. J. Shah. 2012. A Comparative Study of Visual Reaction Time in Basketball Players and Healthy Controls. *National Journal of Integrated Research in Medicine* 3, 1 (2012).

[32] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[33] Google. 2021. Connect smart home devices in the Google Home app. https://support.google.com/googlenest/answer/9159862?hl=en.
[34] Bogdan Groza and Rene Mayrhofer. 2012. SAPHE: Simple Accelerometer Based Wireless Pairing with Heuristic Trees. In *MoMM*.
[35] Miguel Angel Guevara and María Corsi-Cabrera. 1996. EEG coherence or EEG correlation? *International Journal of Psychophysiology* 23, 3 (1996).
[36] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. 2009. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM* 52, 5 (2009), 91–98.
[37] Dianqi Han, Yimin Chen, Tao Li, Rui Zhang, Yaochao Zhang, and Terri Hedgpeth. 2018. Proximity-Proof: Secure and Usable Mobile Two-Factor Authentication. In *MobiCom*.
[38] Jun Han, Shijia Pan, Manal Kumar Sinha, Hae Young Noh, Pei Zhang, and Patrick Tague. 2017. SenseTribute: Smart Home Occupant Identification via Fusion Across On-Object Sensing Devices. In *BuildSys*.
[39] Gerhard Hancke. 2005. *A practical relay attack on ISO 14443 proximity cards*. Technical Report.
[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*.
[41] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997).
[42] Mark R. Hodges and Martha E. Pollack. 2007. An 'Object-Use Fingerprint': The Use of Electronic Sensors for Human Identification. In *UbiComp*.
[43] JACLYN DIAZ. 2021. High Gun Sales And More Time At Home Have Led To More Accidental Shootings By Kids. https://www.npr.org/2021/08/31/1032725392/guns-death-children.
[44] Aditya Jain, Ramta Bansal, Avnish Kumar, and K. D. Singh. 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International Journal of Applied & Basic Medical Research* 5, 2 (2015).
[45] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. In *USENIX Security*.
[46] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. 2015. Real-time patient-specific ECG classification by 1-D convolutional neural networks. *IEEE Transactions on Biomedical Engineering* 63, 3 (2015), 664–675.
[47] Serkan Kiranyaz, Turker Ince, Ridha Hamila, and Moncef Gabbouj. 2015. Convolutional neural networks for patient-specific ECG classification. In *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*.
[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.
[49] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. 2009. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing* 5, 6 (2009).
[50] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. 2011. Activity Recognition Using Cell Phone Accelerometers. *SIGKDD Explor. Newsl.* 12, 2 (2011).
[51] Lingjun Li, Xinxin Zhao, and Guoliang Xue. 2013. Unobservable Re-authentication for Smartphones. In *NDSS*.
[52] Xiaopeng Li, Fengyao Yan, Fei Zuo, Qiang Zeng, and Lannan Luo. 2019. Touch Well Before Use: Intuitive and Secure Authentication for IoT Devices. In *MobiCom*.
[53] Xiaopeng Li, Qiang Zeng, Lannan Luo, and Tongbo Luo. 2020. T2pair: Secure and usable pairing for heterogeneous iot devices. In *CCS*.
[54] James M Lucas and Michael S Saccucci. 1990. Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics* 32, 1 (1990).
[55] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz. 2014. ZEBRA: Zero-Effort Bilateral Recurring Authentication. In *S&P*.
[56] Shrirang Mare, Reza Rawassizadeh, Ronald Peterson, and David Kotz. 2018. SAW: Wristband-based Authentication for Desktop Computers. *IMWUT* 2, 3 (2018).
[57] Shrirang Mare, Reza Rawassizadeh, Ronald Peterson, and David Kotz. 2019. Continuous Smartphone Authentication using Wristbands. In *Workshop on Usable Security*.
[58] R. Mayrhofer and H. Gellersen. 2009. Shake Well Before Use: Intuitive and Secure Pairing of Mobile Devices. *TMC* 8, 6 (2009).
[59] Daniel V. McGehee, Elizabeth N. Mazzae, and G.H. Scott Baldwin. 2000. Driver Reaction Time in Crash Avoidance Research: Validation of a Driving Simulator Study on a Test Track. *HFES Annual Meeting* 44, 20 (2000).
[60] Yuxin Meng, Duncan S Wong, Roman Schlegel, et al. 2012. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In *ICISC*.
[61] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. 2014. Context-Based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices. In *CCS*.
[62] D. L. Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications* 39, 10 (1991).
[63] Fabian Monrose and Aviel Rubin. 1997. Authentication via Keystroke Dynamics. In *CCS*.
[64] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007).

[65] Notion. 2019. Monitor your home wherever you are. https://getnotion.com/.
[66] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *JMLR* 12 (2011), 2825–2830.
[67] Wenjie Pei, David MJ Tax, and Laurens van der Maaten. 2016. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713* (2016).
[68] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V Krishnamurthy. 2010. Denial of service attacks in wireless networks: The case of jammers. *IEEE Communications Surveys & Tutorials* 13, 2 (2010).
[69] Aanjhan Ranganathan and Srdjan Capkun. 2017. Are we really close? verifying proximity in wireless systems. In *S&P*.
[70] Juhi Ranjan, Erin Griffiths, and Kamin Whitehouse. 2014. Discerning Electrical and Water Usage by Individuals in Homes. In *BuildSys*.
[71] Juhi Ranjan and Kamin Whitehouse. 2015. Object Hallmarks: Identifying Object Users Using Wearable Wrist Sensors. In *UbiComp*.
[72] RobotoLab. 2020. NAO V6 price is $9000. https://www.robotlab.com/store/nao-power-v6-educator-pack.
[73] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, and Nasir Memon. 2012. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *CHI*.
[74] Hataichanok Saevanee and Pattarasinee Bhatarakosol. 2008. User authentication using combination of behavioral biometrics over the touchpad acting like touch screen of mobile device. In *ICCEE*.
[75] D. Schürmann and S. Sigg. 2013. Secure Communication Based on Ambient Audio. *TMC* 12, 2 (Feb 2013).
[76] Mohamed Shahin, Ahmed Badawi, and Mohamed Kamel. 2007. Biometric authentication using fast correlation of near infrared hand vein patterns. *International Journal of Biological and Medical Sciences* 2, 3 (2007).
[77] John C. Shaw. 1984. Correlation and coherence analysis of the EEG: A selective tutorial review. *International Journal of Psychophysiology* 1, 3 (1984).
[78] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran S Balagani. 2016. HMOG: New behavioral biometric features for continuous authentication of smartphone users. *TIFS* 11, 5 (2016).
[79] Samsung SmartThings. 2017. SmartThings Multipurpose Sensor. https://support.smartthings.com/hc/en-us/articles/205382174-Samsung-SmartThings-Multipurpose-Sensor.
[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper With Convolutions. In *CVPR*.
[81] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. 2004. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Pervasive Computing*.
[82] Texas Instruments. 2021. *CC2652R SimpleLinkTM Multiprotocol 2.4 GHz Wireless MCU*. Technical Report.
[83] TripWire. 2017. Relay Attack against Keyless Vehicle Entry Systems Caught on Film. https://www.tripwire.com/state-of-security/security-awareness/relay-attack-keyless-vehicle-entry-systems-caught-film/.
[84] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[85] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. 2016. Gated Siamese Convolutional Neural Network Architecture for Human Re-identification. In *ECCV*.
[86] José Vila and Ricardo J. Rodríguez. 2015. Practical Experiences on NFC Relay Attacks with Android. In *Radio Frequency Identification*.
[87] Wei Wang, Lin Yang, and Qian Zhang. 2016. Touch-and-guard: secure pairing through hand resonance. In *UbiComp*.
[88] P. Welch. 1967. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics* 15, 2 (1967).
[89] Wired. 2017. Just a Pair of These $11 Radio Gadgets Can Steal a Car. https://www.wired.com/2017/04/just-pair-11-radio-gadgets-can-steal-car/.
[90] Chuxiong Wu, Xiaopeng Li, Lannan Luo, and Qiang Zeng. 2022. G2Auth: secure mutual authentication for drone delivery without special user-side hardware. In *MobiSys*.
[91] Wei Xi, Chen Qian, Jinsong Han, Kun Zhao, Sheng Zhong, Xiang-Yang Li, and Jizhong Zhao. 2016. Instant and Robust Authentication and Key Agreement Among Mobile Devices. In *CCS*.
[92] Weitao Xu, Chitra Javali, Girish Revadigar, Chengwen Luo, Neil Bergmann, and Wen Hu. 2017. Gait-Key: A Gait-Based Shared Secret Key Generation Protocol for Wearable Devices. *ACM Trans. Sen. Netw.* 13, 1, Article 6 (2017).
[93] Zhenyu Yan, Qun Song, Rui Tan, Yang Li, and Adams Wai Kin Kong. 2019. Towards Touch-to-Access Device Authentication Using Induced Body Electric Potentials. In *MobiComp*.
[94] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *IJCAI*.
[95] Nan Zheng, Aaron Paloski, and Haining Wang. 2011. An Efficient User Verification System via Mouse Movements. In *CCS*.
[96] Xinyan Zhou, Xiaoyu Ji, Chen Yan, Jiangyi Deng, and Wenyuan Xu. 2019. Nauth: Secure face-to-face device authentication via nonlinearity. In *Infocom*.
[97] Fei Zuo, Xiaopeng Li, Patrick Young, Lannan Luo, Qiang Zeng, and Zhexin Zhang. 2019. Neural machine translation inspired binary code similarity comparison beyond function pairs. In *NDSS*.