# Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing using Different Sensor Types

Jun Han, Albert Jin Chung, Manal Kumar Sinha, Madhumitha Harishankar,
Shijia Pan, Hae Young Noh, Pei Zhang, and Patrick Tague
Carnegie Mellon University
{junhan, albertjc, manalkus, mharisha, shijiapan, noh, peizhang, tague}@cmu.edu

***Abstract*—*Context-based* pairing solutions increase the usability of IoT device pairing by eliminating any human involvement in the pairing process. This is possible by utilizing on-board sensors (with same sensing modalities) to capture a common physical context (e.g., ambient sound via each device's microphone). However, in a smart home scenario, it is impractical to assume that all devices will share a common sensing modality. For example, a motion detector is only equipped with an infrared sensor while Amazon Echo only has microphones. In this paper, we develop a new context-based pairing mechanism called *Perceptio* that uses time as the common factor across differing sensor types. By focusing on the event timing, rather than the specific event sensor data, *Perceptio* creates event fingerprints that can be matched across a variety of IoT devices. We propose *Perceptio* based on the idea that devices co-located within a physically secure boundary (e.g., single family house) can observe more events in common over time, as opposed to devices outside. Devices make use of the observed contextual information to provide entropy for *Perceptio*'s pairing protocol. We design and implement *Perceptio*, and evaluate its effectiveness as an autonomous secure pairing solution. Our implementation demonstrates the ability to sufficiently distinguish between legitimate devices (placed within the boundary) and attacker devices (placed outside) by imposing a threshold on fingerprint similarity. *Perceptio* demonstrates an average fingerprint similarity of 94.9% between legitimate devices while even a hypothetical impossibly well-performing attacker yields only 68.9% between itself and a valid device.**

## I. Introduction

While Internet-of-Things (IoT) devices provide significant value to smart home operations, the data they create often contains privacy-sensitive information about user activities within the home [74], [49], [31], [52], [33]. Securing the wireless communication among IoT devices is thus a critical capability for any home IoT deployment. In particular, newly deployed IoT devices must be able to securely pair with existing devices through cryptographic key establishment in a way that protects against man-in-the-middle (MitM) and protocol manipulation attacks [38], [29], [6], [48], [25], [71], [43]. Such protections unfortunately require users to be involved in the protocol (e.g., to type in a password), and such *human-in-the-loop* solutions are not feasible for many IoT systems. The first reason is that the number of IoT devices in a home is projected to increase from around ten to several hundred within the next decade [15], [53], increasing the complexity and burden to the homeowner. Second, most emerging IoT devices do not have a user interface, so direct password entry or management is challenging or impossible [40]. While it is possible to equip IoT devices with pre-loaded keys, user interfaces, or dedicated pairing hardware (e.g., NFC), such approaches would overburden manufacturers, limit interoperability, and slow IoT innovation.

Efforts to reduce or remove human involvement from the secure pairing process has brought the emergence of *context-based* pairing. This allows devices to rely on on-board sensors to extract entropy from the surrounding environment, using the principle that *co-present* devices will observe similar events. The common sensor measurements can be translated to common randomness, forming the basis of a symmetric key agreement protocol [51], [32], [65]. Intuitively, the unpredictability of the activities in the environment provides the entropy source and eliminates the need for a human participant.

While promising first steps have been taken toward developing *usable* and *secure* IoT device pairing, existing solutions rely on a common, properly calibrated sensing capability across all devices (e.g., a microphone or light sensor on each device). However, in reality, a wide diversity of hardware capabilities will be present in a smart home, so a usable pairing protocol must consider this device heterogeneity. We are particularly inspired by emerging IoT products that have a small number of embedded sensors (often only one) to optimize cost, power consumption, and form factor (e.g., motion detector with a single infrared sensor [11] or peel-and-stick sensors [60]). One of the major challenges in this heterogeneous device landscape is that the sensor measurements from different IoT devices will not be directly comparable. Aside from different sensing modalities, manufacturers may use different chipsets or calibration methods, so even sensors of the same type may measure an event in a different way. Heterogeneous sensor-based pairing protocols must therefore rely on a suitable invariant property that can be measured by devices with a wide variety of configurations.

Toward such goals, we need to gain a stronger understanding of the contextual content of sensory data as observed from different IoT devices. To do this, we can gain some insight from analogous human behavior through the following scenario. Suppose that one person with a hearing impairment and another with a visual impairment are both in a room. When the door to the room closes, both people can observe the event at the same time, but using different senses; the hearing impaired person can see the door closing, while the visually impaired person can hear

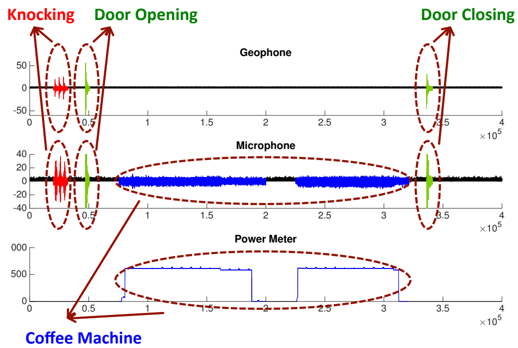



Fig. 1: We demonstrate how different types of sensors are capable of measuring aspects of the same events.

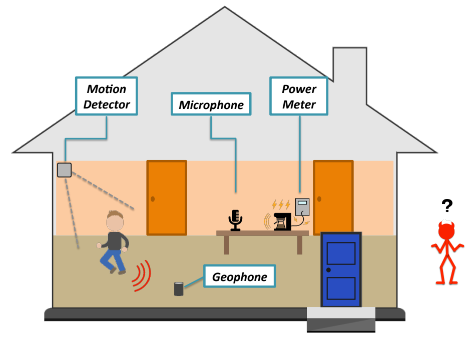



Fig. 2: A physical boundary (house) provides a perceptual separation between user's devices inside vs. other devices outside, enabling *context-based* pairing via observations of random events within the house.

the door closing. Because of the timing, both people could share their observations and determine they had witnessed the same event. This analogy can be further extended to include events that humans perceive in multiple ways. For example, we perceive rainfall through hearing, feeling, and seeing raindrops [8]. By applying this analogy to the IoT device space, we can similarly leverage timing information as an invariant property among heterogeneous devices. We thus develop our approach using a principle we refer to as "*numerically different yet contextually similar*" observation of events, exploiting commonly observed timing information. In the IoT device regime, we provide a more detailed example to demonstrate the ability for disparate sensing devices to measure common events. In this scenario, Bob knocks on his roommate Dan's door to invite him for coffee in the living room. Dan opens his bedroom door and walks into the living room, and Bob then makes two cups of coffee. After enjoying their coffee together, Dan goes back into his bedroom and closes the door. Suppose now that Bob and Dan have deployed IoT devices with a *geophone* and *microphone* and that the coffee machine is connected to a *power meter*. In this case, the corresponding sensor readings from these devices capture the events, as depicted in Figure 1. The different types of sensors are capable of perceiving some events in common. In particular, the geophone and the microphone both capture the knocks and door opening/closing events, while the microphone and power meter both capture the activity of the coffee machine.

In a more general scenario, sensing devices can detect relevant events, group them by event type (e.g., using unsupervised clustering), measure the time interval between subsequent occurrence of each event type, and map the time interval measurements to a collection of *fingerprints*. These fingerprints can then be incorporated into a verification algorithm to prove co-presence and contribute to the generation of a shared symmetric key. Different fingerprints can be incorporated to verification with other devices, depending on the sensing capabilities of each device.

While our use of inter-event timing removes some useful signal content, this comes in trade for support of many critical aspects of practical fingerprint verification that may not be possible using more descriptive signals. Most importantly, elimination of numerical signal features introduces a high degree of tolerance that addresses (1) differing hardware or sensor calibration methods, (2) signal attenuation due to variations in proximity between sensors and events, and (3) measurements from different sensor types. In addition, the use of time intervals eliminates the need for tight time synchronization across devices. Moreover, the devices do not need to recognize the events themselves to measure timing, but simply (as discussed later) group events by clusters using unsupervised learning.

Another key insight contributing to our approach is the idea that IoT devices deployed in a common environment are intended to collaborate as part of the same smart home system. Hence, there is an implicit human-driven intent for the devices to pair with each other as long as they can determine that they are deployed in the same environment. In the context of a single-family home, comprising the majority of housing units in the U.S [46], the building structure and composition provide a barrier for many types of activities that would be observed by sensors, including but not limited to vibration, sound, light, or electrical load. Through the combination of this physical sensing barrier and the typical physical security of a single-family home, the *secure pairing* problem reduces to verifying *co-presence* within the home.

Our approach to verifying device co-presence leverages the fact that devices deployed in the same room of a house will perceive most of the same events over time, while devices outside the home will observe different (or significantly attenuated) events. Random events induced by user activities (e.g., walking, making coffee) within the home thus provide the necessary entropy to enable co-presence verification. Because our approach is based on ***sensory perception*** of events in the surrounding environment, we name our autonomous device pairing technique ***Perceptio***. In Figure 2, we illustrate the high-level ideas of *Perceptio*, where multiple devices within the home observe physical events that cannot be clearly observed by the outside attacker.Building on this idea, *Perceptio* enables IoT devices to effectively *fingerprint* their surroundings with no human involvement, achieving maximum usability. *Perceptio* uses these fingerprints for symmetric key establishment, taking advantage of our findings that an outside attacker can neither accurately observe nor predict events within the home at the temporal granularity required for verification. From these findings, our *Perceptio* design includes a *Key Strengthening Process* that builds confidence in co-presence verification over time. Starting with an initial shared key (that may be insecure), *Perceptio* augments this key with subsequent fingerprint information until reaching a desired strength. This process is similar in spirit to a multi-round security protocol.

To evaluate the design of *Perceptio*, we perform experiments by equipping a room with a variety of sensors to represent existing and prevalent commercial IoT products. Our deployment includes a microphone (smart speakers [7], [27]), an accelerometer (on-object sensors [69], [23], [56]), a motion detector [55], [20], a power meter [37], [36], and a geophone (structure or footstep monitors [73], [58]). In addition, we deploy corresponding devices as well as higher quality microphone and accelerometer outside the room to represent the attacker's devices. Human participants perform a number of typical events in the room, providing the ambient inputs to the various sensors. As a proof of concept, our empirical evaluation demonstrates that fingerprints generated by devices within the room are far more likely to match (yielding an average of 94.9%), while the highest fingerprints generated by the attacker's devices outside the room have low similarity to those inside the room (only yielding an average of 68.9%). To support the proof of concept, we study existing data sets for activity within smart homes to quantify the available entropy and the corresponding amount of time for devices to establish keys with sufficient confidence.

Overall, our contributions in this paper are as follow.

- We develop an autonomous context-based pairing protocol, named *Perceptio*, for IoT devices with heterogeneous sensing types, using a fingerprint mechanism that is robust to signal variation across devices, requires no time synchronization across devices, and needs no prior training phase.
- We demonstrate through proof-of-concept implementation and experimentation that *Perceptio* can differentiate between devices inside and outside of the room, effectively protecting against attacking devices located just outside a user's home.
- We analyze existing data sets to quantify entropy extraction rates in real-world smart home scenarios, in support of quantifying the time to build sufficient confidence in device pairing.

The remainder of this paper is organized as follows. We discuss background and relevant related work in Section II, and present models and assumptions in Section III. In Section IV, we present the entropy extraction and fingerprinting techniques, and we then present the *Perceptio* protocol in Section V. We present our proof-of-concept implementation of *Perceptio* in Section VI and subsequent evaluation in Section VII. We discuss practical deployment considerations and limitations in Sections VIII and IX, respectively. We then conclude our work in Section X.

## II. Background and Related Work

We present background information on sensors equipped by smart home devices, and related work on secure pairing.

**Commercial Smart Home Sensors.** We witness many smart home IoT devices commercially available today. Each of these devices is equipped with a small number of on-board sensors (often one), with a specific sensing modality – e.g., smart speakers equipped with microphones and motion detectors equipped with PIR sensors. We present a more detailed overview of smart home IoT devices and their corresponding sensor types in Appendix A. We present *Perceptio* to enable these smart home devices of heterogeneous sensor modalities to prove that they are co-located within a physical boundary by experiencing similar events.

***Human-in-the-Loop*-based Pairing.** We first highlight some of the traditional secure pairing protocols using *human-in-the-loop* solutions. One of the work in this category is Seeing-is-Believing, which authenticates other device's public key by taking a picture of a 2D bar code which encodes the hash of the public key of the other device [47]. Furthermore, many industry standards such as Bluetooth Secure Simple Pairing [29] and Wi-Fi Protected Setup [6] requires humans to enter passwords on the devices intended for pairing. These solutions, however, are not applicable in smart home environments due to usability concerns.

***Context*-based Pairing.** Researchers also explore *context-based* pairing protocols to capture commonly observed context for pairing leveraging on-board sensors without requiring human involvement. Miettinen et al. propose recurring authentication when pairing devices at home by leveraging ambient sound or light [51]. Devices co-located at one household would experience similar context as opposed to devices in a neighbor's house. Schurmann et al. propose a similar idea, but leverage short audio as contextual information for secure pairing [65]. Rostami et al. propose a key agreement scheme between an implanted heart with its remote programmer [63]. They establish a shared key by extracting entropy bits from measuring the patients heart beat. Han et al. propose recurring authentication across trucks driving on a highway by sensing context from the road bumpiness using accelerometer [32]. While these approaches are promising first steps in the context-based pairing schemes, they all focus on leveraging identical sensor pairs across devices such as microphones, accelerometers, and other sensors using direct signal analysis. Unlike these homogeneous context-based pairing schemes, *Perceptio* addresses a difficult but interesting question of how to enable differing (i.e., heterogeneous) sensor modalities to capture the same contextual information.

## III. Models and Assumptions

We now present our threat model describing the goals and capabilities of the attacker. Subsequently, we present the assumptions and constraints of *Perceptio*.

### A. Threat Model

The goal of the attacker is to *leak private information* of home occupants by eavesdropping on the communication between IoT devices. In order to achieve this goal, the attacker may launch (1) *Shamming attack* or (2) *Man-in-the-Middle attack*.

We define a Shamming attack where the attacker's device, $\mathcal{M}$ (placed outside of the house but within the wireless communication range), succeeds in fooling a legitimate device, $\mathcal{LD}$ (inside the house), to accept the pairing as another $\mathcal{LD}$. $\mathcal{M}$ may launch three types of Shamming attacks. First, it may launch an *(1-a) Eavesdropping attack* by attempting to sense (from outside) the events occurring inside. $\mathcal{M}$ may have following three levels of capabilities to launch this attack. $\mathcal{M}$ may have *(i) normal-level of resources* equipped with standard off-the-shelf IoT sensors that are comparable to $\mathcal{LD}$s inside the house. $\mathcal{M}$ may also have *(ii) medium-level of resources* equipped with higher-end off-the-shelf consumer electronic devices that are more powerful than (i). Furthermore, $\mathcal{M}$ may have *(iii) powerful-level of resources* equipped with asymmetric capabilities (e.g., military-grade thermal imaging and x-ray vision). As such, we focus on (i) and (ii) and disregard (iii) because such attackers could already visualize activities within the home and reveal private activities, independent of *Perceptio* and the IoT devices deployed within the home. Moreover, the attacker may launch other types of Shamming attack such as: *(1-b) Signal Injection attack* – by creating events with large noise or vibration from outside (e.g., using jack-jammer); or *(1-c) Sensor Spoofing attack* – by injecting spoofing signals to $\mathcal{LD}$s. The attacker launches either of these attacks again in an attempt to allow both $\mathcal{M}$ and $\mathcal{LD}$s to perceive simultaneous event signals and ultimately succeed in fooling $\mathcal{LD}$s to accept the pairing with $\mathcal{M}$.

Second, $\mathcal{M}$ may launch a man-in-the-middle (MitM) attack on key agreement messages between a pair of $\mathcal{LD}$s by simply intercepting messages transmitted over the wireless medium. Such an attacker is able to use a variety of primitives such as injection, replay, modification, and blocking/deleting messages in the communication channel.

### B. Assumptions and Constraints

We assume that the physical boundaries of a house draw a natural trust boundary for deployed devices, $\mathcal{LD}$s. This assumption reflects scenarios in which $\mathcal{LD}$s inside the boundary are owned and operated by a common entity (e.g., home owner). However, non-authorized personnel do not have access to the physical space, hence do not have control over the IoT devices. We also assume that the family members and authorized guests are not malicious. For example, if one's family members or authorized guests are the only people who have access to their house, and devices brought into the home for prolonged periods of time are assumed to be trustworthy, then a proof of deployment within the house is sufficient to bootstrap a trusted connection to the IoT network. We view the introduction of unauthorized devices into the home by malicious guests as a problem of the homeowner's physical security, not as a relevant problem of secure pairing. Hence, this issue is out of scope for our work.

In addition, we acknowledge that single-family homes are made up of a number of joined rooms, and the separating walls actually present numerous physical boundaries within the home. While sensors within the same home are likely to perceive some common events due to the common physical structure, the walls are bound to induce a *non-negligible attenuation factor*, with different propagation media causing distortion and attenuation of mechanical signals. More specifically, walls and joints are known to cause material damping, reflection and diffraction of acoustic and vibration signals [39], [26]. However, since interior walls tend to provide far less attenuation compared to exterior walls, we expect a fair amount of signal to propagate between nearby IoT devices, at least a sufficient amount to allow for IoT network connectivity, as full pairwise connectivity is likely unnecessary. As we will discuss later, it may also be possible to configure a small number of IoT devices to act as "bridging devices", if needed, to facilitate secure pairing across the internal walls of the home.

In either case, we design *Perceptio* to rely on the core observation that sensors outside the home cannot *consistently perceive* the relevant activities inside with *similar fidelity* as $\mathcal{LD}$s. While our design focuses on single-family detached housing (comprising 61.5% of U.S. housing [46]), we believe that future extensions of *Perceptio* could extend our work to other multi-tenant attached housing (e.g., apartments or townhouses) through rigorous engineering of thresholds and other protocol parameters.

## IV. Entropy Extraction and Fingerprinting

We first present different sources of shared entropy that can be used to bootstrap trust among the IoT devices. Subsequently, we explain how to extract the entropy via our context fingerprinting mechanism.

### A. Entropy Extraction

Analogous to a cryptographic key agreement protocol relying on a source of entropy to establish (pseudo-)random key bits, we propose approaches to enable devices to capture and extract shared entropy from the device's surroundings, which is later used to bootstrap trust as discussed in Section V.

One possible approach to help devices extract shared entropy is to deliberately inject randomness to the devices within the physical boundary. This may be realized by introducing a *signal injecting device* (e.g., device with vibration motor or speaker) that outputs signals such as vibration or sound that are encoded random bits. This is analogous to traditional key establishment schemes that provide "deliberate entropy" [9]. However, this solution poses many practical concerns regarding cost and usability, as well as scalability with respect to multiple sensing modalities.

To address the above concerns, we propose an approach that relies on the inherent randomness of events in a device's surroundings to establish a context fingerprint, i.e., "natural entropy". We leverage the inherent randomness of events occurring in a room (e.g., knocking, walking, talking, etc.) as its source of entropy for a cryptographic protocol. Specifically, *Perceptio* leverages the fact that it is infeasible for an attacker to predict the precise timing of events within the physical boundary at a millisecond-scale granularity. Using the randomness in event timing, the fundamental goal of the fingerprint generation mechanism is for two devices to generate “similar” fingerprints only if they meet the contextual requirements of the scenario. Unlike traditional secure pairing protocols, however, the nature of our problem requires that there is a degree of tolerance to capture the dissimilarities between sensing devices and their respective abilities of perception, namely relaxing the requirement that fingerprints $F_{Device_A}$ and $F_{Device_B}$ are numerically equal to instead satisfy $d(F_{Device_A}, F_{Device_B}) < \epsilon$ for a suitable distance metric $d$ and small tolerance $\epsilon > 0$ only when the two devices “match”. For now, we leave the specifics of fingerprint matching to the later sections and focus on the fingerprinting mechanism.

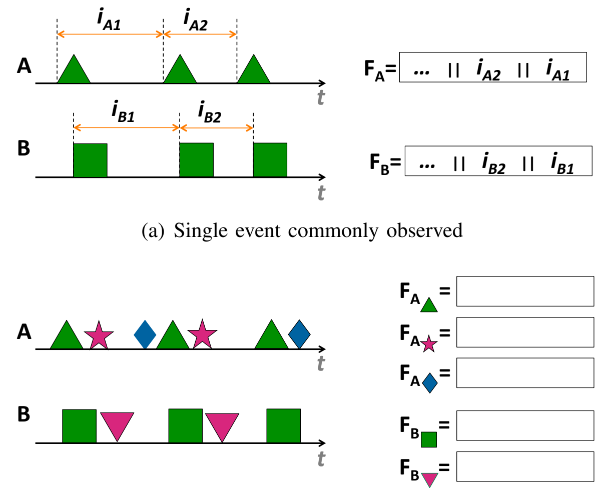
### B. *Context Fingerprinting*

We present the fingerprint extraction algorithm and how multiple event types affect *Perceptio* context fingerprinting. We also explain how *Perceptio* guarantees sufficient entropy needed for key agreement protocol.

*1) Fingerprint Extraction Algorithm:* The main idea behind *Perceptio*’s fingerprinting mechanism is based on three primary insights: (1) raw signals obtained by different devices and sensor types will have different characteristics; (2) sensors on different devices will perceive the same event in roughly the same way; and (3) inter-event timing measured by different sensors will be roughly the same. When we combine these three properties, we arrive at an approach that combines event detection, event clustering, and per-cluster inter-event timing. Specifically, each device will generate a set of fingerprints, one for each cluster, that collectively represent the observable context. Note that devices do not need to know what specific types of events are occurring. From these core ideas, it is clear that the context fingerprinting approach is general, and we will further describe specific use cases and experimental evaluations in later sections.

To illustrate how the start times and corresponding inter-event intervals (time between start of subsequent events of the same type) are used to create the fingerprints, we provide Figure 3(a). The figure highlights the fact that the two sensors do not need to have a common representation of the event detected (one device labels the clusters with ▲ and the other uses ■), but the inter-event timings match. Note also that the event detection does not need to be perfectly synchronized. In general, each device measures an event sequence $S$ yielding the inter-event times, $i_S$, and the resulting fingerprint, $F$, is computed by concatenating bit-representations of intervals as $F_A = \{i_{S_n} || i_{S_{n-1}} || ... || i_{S_1}\}$.

We further take into account that a sensor is capable of detecting multiple events. Consider one device $A$ with a microphone and another device $B$ with a geophone. Microphone will be more sensitive to talking, and geophone will be more sensitive to vibrations caused by walking, but both will sense aspects of a running coffee machine, since it vibrates and emits sound. In this case, the two devices can each detect




(a) Single event commonly observed

(b) Events observed by each sensor

Fig. 3: Figure depicts fingerprint ($F$) extraction from starting point intervals of same event type (i.e., cluster). (a) depicts how two sensors with different modalities, $A$ and $B$, represents commonly observed event differently as ▲ and ■, respectively. Despite differences in representation, inter-event timings, $i_A$ and $i_B$, are similar. (b) depicts how $A$ and $B$ extracts various fingerprints from many event types.

multiple event types (including but not limited to talking, walking, and making coffee). Each device will collect its time-series data, perform a sequence of signal processing to detect events, cluster the events based on various signal properties, and create a fingerprint for each event cluster. The microphone’s event sequence, $S_A$, may involve three event types – talking, walking, and making coffee – while the geophone’s event sequence, $S_B$, may involve two event types – walking and making coffee. From Figure 3(b), we see that the microphone labeled its three event clusters with {▲, ★, ♦} and the geophone labeled its two event clusters with {■, ▼}. The embedded devices creates sets of per-cluster fingerprints {$F_▲$, $F_★$, $F_♦$} and {$F_■$, $F_▼$}, exchange them with each other, and perform a pairwise search to see if any of the fingerprints match (Section V).

*2) Fingerprint Entropy:* *Perceptio* bootstraps its trust from the entropy of event timings in the environment. Intervals between starting points of subsequent event observations are translated into the bits of the fingerprint. Hence, the entropy of the fingerprint depends on the number of similar events observed and the bit resolution of each interval. This is depicted in (Equation 1). $F$ depicts the concatenation of bit values of intervals $i_{A_k}$, for $k = 1, \ldots, n$. If the length of $F$ is less than a minimum acceptable fingerprint length $l_F$, the fingerprint is discarded due to insufficient entropy, otherwise $F$ is truncated to $l_F$ bits. We explain the requirement of $l_F$ in Appendix C in order to provide sufficient entropy.

$$F_{final} = \begin{cases} \Big[F\Big]_{l_F}, & \text{if } |F| \geq l_F \\ \emptyset, & \text{otherwise} \end{cases} \quad (1)$$

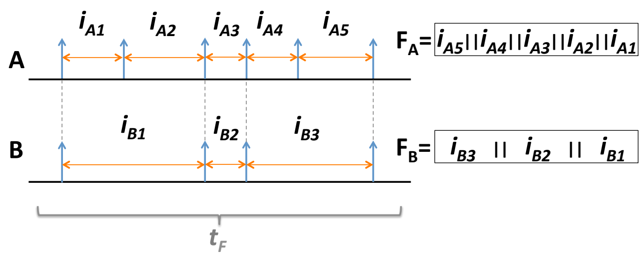



Fig. 4: Figure depicts how incorrect event detection affects error accumulation when extracting $F$ over $t_F$. $B$ mistakenly misses two observations, but the resulting $F_B$ accumulates error, analogous to framing error in serial communications.

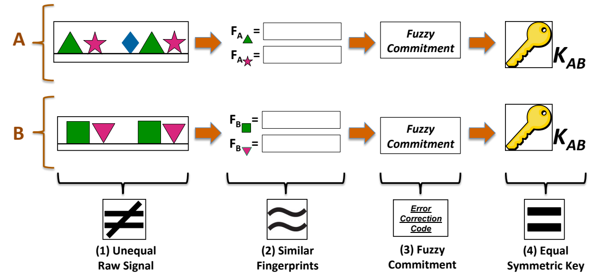



Fig. 5: Figure depicts *Perceptio* protocol overview. Unequal heterogeneous sensors data from $A$ and $B$ are eventually converted to numerically equivalent symmetric key.

*3) Advantages of Fingerprint Extraction Algorithm:* We present a series of important advantages inherent to the *Perceptio* fingerprint extraction algorithm. First, the devices are not required to be time synchronized as (1) fingerprint extraction can be triggered by event occurrences and (2) fingerprints are generated based on event intervals rather than specific event occurrence times. As long as the clock rates are consistent across the devices, the generated fingerprints will be similar regardless of time synchronization. Second, the generated fingerprints are independent of the varying amplitudes of the captured signal depending on the location of the sensors relative to the source of the event. This is also because the algorithm makes use of the starting point intervals rather than the signals themselves.

The fingerprint algorithm inherently provides robustness against malicious adversaries launching Shamming attack. First, the algorithm makes it increasingly difficult for an attacker to predict events at a fine granularity. While some of the daily activities in a house seems rather predictable (e.g., opening a door around 9 a.m.), it is extremely difficult to predict it at the millisecond granularity, making it possible to extract entropy from the context. Second, the algorithm inherently protects against an attacker's device capturing some of the events from outside the physical boundary, as the attenuation factor of the physical boundary (e.g., walls) is assumed to be non-trivial. However, capturing only some of the events by an attacker's device is insufficient to create a fingerprint that is similar enough. This is because the errors accumulate as the attacker's device misses certain events, as illustrated in Figure 4. In this example, sensors $A$ inside and $B$ outside the boundary generate different fingerprints because of $B$'s inability to sense everything that $A$ senses. Even such non-consecutive event misses are detrimental to the attacker because the error accumulates, analogous to framing errors in serial communications. Hence, in order for the attacker's device to succeed in pretending to be a device within the physical boundary, it needs to consistently capture most of the events occurring in the room. We further analyze this difficulty with empirical data in Section VII-B.

## V. Protocol Design

*Perceptio*'s fingerprint verification incorporates the fingerprint, $F$, into a cryptographic protocol to yield a verifiable shared symmetric key between the two parties. Figure 5 depicts the high-level overview of *Perceptio* protocol. (1) Initially two devices with disparate sensor modalities captures numerically unequal time series data streams. (2) While co-located devices observe similar events, the extracted pair of fingerprints will not be exactly the same due to sensitivity and different modalities. (3) We treat such subtle differences in fingerprints as errors and tolerate them using a fuzzy commitment scheme [41], [18] building on error correcting codes. (4) Finally two devices share a master symmetric key, $k$, and can subsequently generate shared session key, $k_{AB}$. Similar to the related work [51], [32], we design a *Key Strengthening Process*, which gradually *strengthens* the initially shared (but potentially insecure) key. This is made possible by gradually increasing the authenticity confidence over time through repeated execution of the fuzzy commitment using different fingerprints (Steps (1) through (4)), until a minimum confidence score is attained, inherently making it extremely difficult for Shamming attacker devices (located outside of the physical boundary) to sustain the shared key.

**Protocol Details.** *Perceptio*'s fuzzy commitment protocol is composed of four main phases – (1) *Initialization*: devices discover each other and determine through exchange of identifiers that they wish to pair with each other; (2) *Key Agreement*: devices compute, exchange, and verify context fingerprints to establish a symmetric key; (3) *Key Confirmation*: devices verify the correctness of the symmetric key and increment the confidence score if the key is validated; and (4) *Confidence Score Check*: devices either declare pairing success if the confidence score is above a certain threshold or repeat from the key agreement phase. These phases are depicted in Figure 6 and described in more detail as follows. We intentionally omit the underlying cryptographic protocol details in this section, but present an in-depth description in Appendix B.

In the ***Initialization Phase***, device $A$ initiates a broadcast message containing its identifier (e.g., device ID or pseudonym). A nearby device $B$ that receives the message and wishes to "pair" with $A$ responds with a $RQST\_TO\_PAIR$, including its identifier in the request. If $A$ also wishes to pair with $B$, it responds with a $RSP\_TO\_PAIR$ message, at which point both devices continue to the ***Key Agreement Phase***. $A$ and $B$ follow

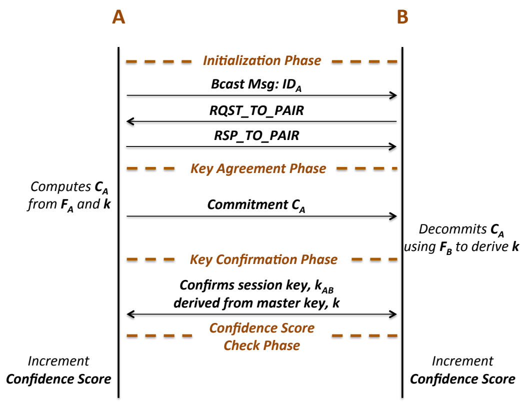



Fig. 6: Figure depicts *Perceptio* protocol flowchart diagram. Details of this protocol is presented in Appendix B.

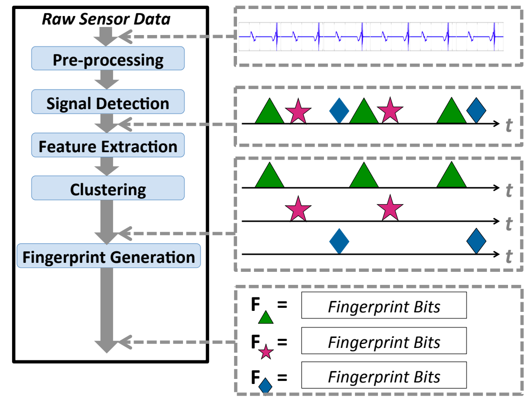



Fig. 7: Figure depicts *Perceptio* fingerprint generation flow chart, taking as input the raw sensor data, performs clustering to distinguish group of events (e.g., walking, door opening, knocking) represented as ▲, ★, ♦, and computes corresponding fingerprints per event cluster (i.e., $F_\blacktriangle$, $F_\bigstar$, and $F_\blacklozenge$).

the previously described fingerprint generation process to create respective sets of fingerprints $\{F_{A_i}, i = 1, \ldots, p\}$ and $\{F_{B_j}, j = 1, \ldots, q\}$ for the $p$ and $q$ observed event clusters. Using the fingerprints, $A$ is able to compute a set of commitments, $C_A$, that it hides a set of secrets generated by $A$, $k_i$, by effectively encrypting it using an extracted fingerprint, $F_{A_i}$. Another device, $B$, can decode the message to acquire $k_i$ only if it has a fingerprint $\hat{F_{B_J}}$ that is "close enough" to $F_{A_i}$. The fuzzy commitment primitive is similar in spirit to encryption of $k_i$ with $F_{A_i}$ using a one-time pad. Once $B$ successfully derives the secret with its $F_{B_j}$ that is similar to $F_{A_i}$, $A$ and $B$ has now established a shared symmetric master key, $k$. Then the two devices continue to the ***Key Confirmation Phase***, creating a shared session key, $k_{AB}$, derived from $k$. Each time a pair of devices $A$ and $B$ successfully execute the key generation and confirmation phases for a round, they increase their respective confidence by a small amount. Upon each increment, each device verifies $ConfScore > Thr_{Conf}$. In the final ***Confidence Score Check Phase***, each device can finally decide that the other is contextually verified once the overall confidence score is above a threshold. The confidence score check mechanism is thus similar to a reputation system.

In addition to the four main phases, *Perceptio* protocol includes an optional extension to allow a notion of *transitive verification* for cases where two devices want to verify each other but their sensing equipment does not allow for generation of matching fingerprints (e.g., the accelerometer and the power meter who perceive no event in common). We call this extension ***Transitivity of Trust (ToT)***. If the two devices $A$ and $C$ have each performed the fingerprint verification with a third device $B$, meaning $A$ and $B$ share key $k_{AB}$ and $B$ and $C$ share key $k_{BC}$, $A$ and $C$ can rely on $ToT$ to expand the "pairing" operation to a "grouping" operation by leveraging authenticated encryption scheme [64] to exchange public parameters for Diffie-Hellman key exchange [17]. Furthermore, this approach enables devices located in different rooms within a house to pair, leveraging *bridging devices*. We discuss this extension further in Section IX.

## VI. Implementation

We now present our implementation of *Perceptio* fingerprint generation. Figure 7 depicts the flow chart diagram. First, each sensor perceives the contextual information by measuring its sensor data for a fingerprinting time period, $t_F$. Measured data is input to *Pre-processing* module for noise reduction. The pre-processed signals are then input to *Signal Detection* module, which distinguishes event signals (e.g., walking, door opening, knocking.) against the rest of the signal and outputs the corresponding signal time indices. Subsequently, the the indices, along with detected signals are input to *Feature Extraction* and *Event Clustering* module, which performs unsupervised learning to cluster signals of similar events via $K$-Means clustering. This is analogous to categorizing detected event signals into clusters of ▲, ★, ♦. The *Fingerprint Extraction* module then converts the resulting cluster indices into corresponding fingerprints per cluster (i.e., $F_\blacktriangle$, $F_\bigstar$, and $F_\blacklozenge$). We present the implementation details of the *Signal Detection* and *Event Clustering* modules.

### A. Signal Detection

Signal detection module identifies events of interest by (1) signal smoothing and (2) threshold-based detection.

*1) Pre-processing: Signal Smoothing for Noise Reduction:* We first compute a moving average to smooth the signal for noise reduction, specifically applying an exponentially weighted moving average (EWMA) filter to discrete time series $x$ as $y[k] = \alpha * x[k] + (1-\alpha) * y[k-1]$, where $\alpha$, $k$, $x[k]$, and $y[k]$ denote the weight, sample index, sensor data and moving average data, respectively. Hence, EWMA smooths the signal while retaining significant changes. Figure 8(a) depicts the original geophone signal of the event of a person walking. Figures 8(b) and (c) depict the absolute values of the original and EWMA-filtered values, respectively.

*2) Thresholding and Signal Detection:* We then perform *thresholding* for signal detection, including both a lower-bound ($Thr_{lower}$) and an upper-bound ($Thr_{upper}$) threshold.

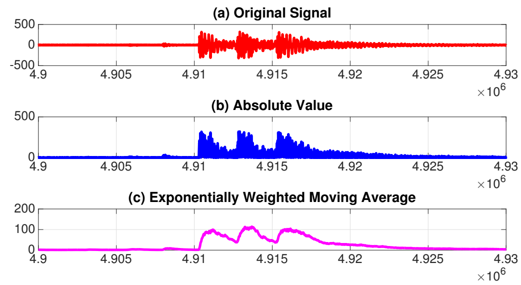



Fig. 8: Figure depicts an example of *Pre-processing* module where the (a) raw sensor signal is first converted to (b) corresponding absolute value, and ultimately converted to (c) subsequent pre-processed signal.

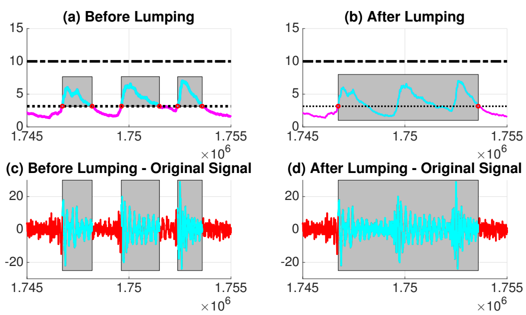



Fig. 9: Figure depicts an example of the *Signal Detection* module, and demonstrate the effects of thresholding and signal isolation.

We leverage $Thr_{lower}$ to distinguish event signals to ambient noise. On the other hand, we also leverage $Thr_{upper}$ to remove any signals of high amplitude, in order to thwart Shamming attack. We note that $Thr_{upper}$ can be a function of $Thr_{lower}$ after certain calibration phase.

This is depicted in Figure 9(a), where we apply a lower-bound thresholding to the *EWMA* signal using the lower dotted line (i.e., $Thr_{lower} = 3$). The signal above the threshold are highlighted with a gray box. Also, we apply an upper-bound thresholding as well using the upper dotted line (i.e., $Thr_{upper} = 10$). For more accurate event clustering, however, we implement a signal *lumping* technique to group segmented parts of the event signal into a single event signal, as shown in Figure 9(b). Specifically, we disregard short discontinuities between adjacent segmented signals above threshold to "lump" the signals into one continuous group of signal event. From the indices returned by these steps, we determine the signal of interest in the original signals as presented in Figures 9(c) and (d), depicting before and after lumping technique, respectively. Finally, this module outputs the corresponding indices of detected signal to the *Event Clustering* module.

### B. Event Clustering

We implement event clustering to appropriately group observed events. Though some additional work may increase the accuracy and efficiency of the clustering results, we detail a preliminary proof-of-concept implementation.

*1) Feature Extraction:* We select a set of features per sensor to reliably separate perceived events via clustering. We select common time-domain features for analysis (e.g., maximum amplitude, duration, and area under the curve and its variants) and evaluate them using principle component analysis. We choose final set of features based on their capacities to maximize visibilities across events. We choose maximum amplitudes and lengths for geophone, microphone, and accelerometer. Motion and power meter did not require clustering as these sensors only perceive one specific event in our experiments. Hence, we performed dimensionality reduction via feature extraction process while retaining essential features for differentiating events.

*2) K-Means Clustering and Elbow Method:* We leverage K-Means clustering to eliminate the need for a training phase. K-Means takes as input $k$ cluster groups and outputs data points to similar clusters. K-Means algorithm computes the Euclidean distances between data points and then selects cluster centroids that minimizes the distances.

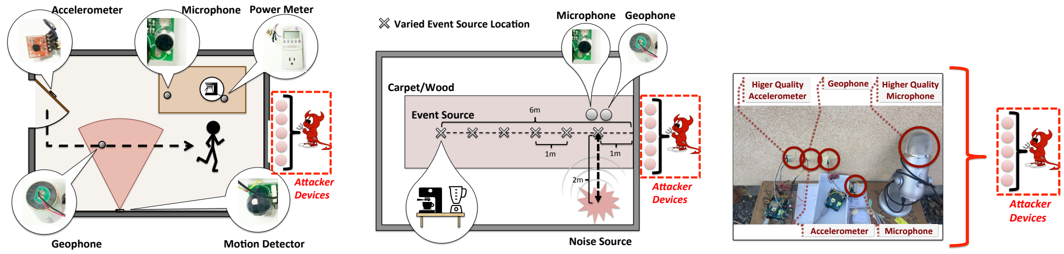
The number of cluster groups is unknown in *Perceptio*, as the devices do not know how many types of events will occur. To address this issue, we leverage Elbow method to infer the optimum value of $K$ [42]. Elbow method tests several $K$-cluster hypotheses to output the optimum $K$ value. Specifically, this method evaluates the rate at which data variances captured by the clusters increase when varying $K$. By leveraging K-Means and Elbow method, *Perceptio* increases its practicality by eliminating the burden of the user or device manufacturer to train specific event types.

## VII. Evaluation

We implement the *Perceptio* protocol and evaluate its effectiveness in different settings. After detailing the apparatus used, we present an end-to-end study of *Perceptio*'s various aspects, including sensors' event detection abilities and robustness of fingerprint similarity and key establishment.

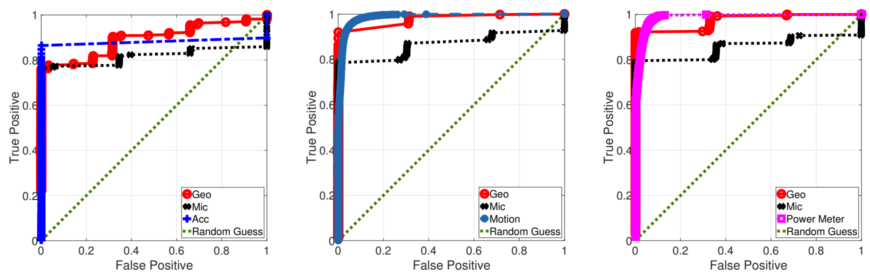
### A. Experiment Apparatus

We describe the nature of legitimate devices, $\mathcal{LD}$s, placed inside the environment and attacker devices, $\mathcal{M}$s, placed outside attempting to launch Shamming–Eavesdropping attack. The $\mathcal{LD}$s include a SM-24 geophone [13], an MD9745APA-1 microphone [3], an ADXL335 accelerometer [16], an MP Motion Sensor NaPiOn passive infrared motion detector [59], and a Kill-A-Watt P4400 power meter [37]. Each of the sensors is interfaced to an Arduino Uno board [5] with a Wireless SD Shield [4] and microSD card for data logging at 5 kHz sampling rate. The sensors were placed between 2.5-5.5m apart from each other. The $\mathcal{M}$s also include a SM-24 geophone, MD9745APA-1 microphone, and an ADXL335 accelerometer, as well as a higher-quality MMA1270KEG accelerometer [66] and a higher-quality Blue Yeti microphone [50] as depicted in Figure 10(c). The higher-quality accelerometer and microphone cost an estimated \$10 and \$100 respectively, which is roughly one and two orders more expensive than the normal-quality IoT accelerometer and microphone.

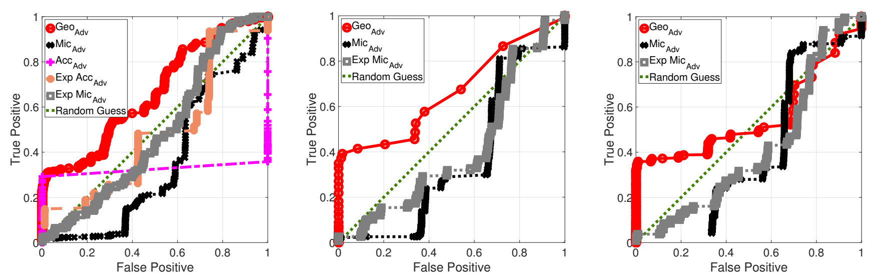



(a) Evaluating event detection accuracy for various sensing modalities (b) Evaluating the impact of different environmental factors (c) Attacker devices

Fig. 10: To study event detection accuracy for $\mathcal{LD}$s and $\mathcal{M}$s of different sensor modalities, we have human subjects conduct the following actions shown in (a): knock on a door hosting an accelerometer, walk across a motion detector, around a microphone and geophone on the ground, and brew coffee from a machine attached to a power meter. The attacker sensors are placed outside the wall opposite to the door. We study the effect of environmental factors in (b): a coffee machine and blender are used successively while varying the distance between them and the sensors, the floor type and the noise level inside the room. We illustrate the five $\mathcal{M}$s in (c) including higher quality accelerometer and microphone.




(a) Legitimate Devices – Knock (b) Legitimate Devices – Walk (c) Legitimate Devices – Coffee




(d) Attacker Devices – Knock (e) Attacker Devices – Walk (f) Attacker Devices – Coffee

Fig. 11: We study the ROC of $\mathcal{LD}$s and $\mathcal{M}$s for accuracy of event detection. Across all events, the $\mathcal{LD}$s have a high detection rate while the $\mathcal{M}$s (even the higher-quality microphone and accelerometer) hardly perform better than a random guess. (Note: For each event type, we only show sensors whose modalities have the ability to detect that event. For example, the accelerometers cannot detect the coffee machine, hence are ignored in (c) and (f)).

## B. Event Detection

*1) Detection Abilities of Legitimate and Attacker Devices:* We now evaluate the performance of each sensor in distinguishing event signals from ambient noise. Recall from Section VI-A the three variables of interest are a lower-bound threshold $Thr_{lower}$ to separate the signal from noise; an upper-bound threshold $Thr_{upper}$ to discard distinct signals with high amplitude to thwart Shamming–Eavesdropping attacks; and the weight $\alpha$ used in the exponential moving average. In this experiment, we vary $Thr_{lower}$, which is important in signal detection, while fixing $Thr_{upper}$ and $\alpha$ to empirically optimized values.

We illustrate the study setup in Figure 10(a). The experiment is conducted in a squash court wherein the $\mathcal{LD}$s are arranged with the geophone on the floor, the microphone on a table, the accelerometer on the door, the motion detector aimed at the center of the room, and the power meter supporting a single serving coffee machine (Nespresso Pixie Carmine [54]). The $\mathcal{M}$s deployed just outside the room (as illustrated in Figure 10(a)) include the accelerometer, the higher-quality accelerometer and the geophone attached to the outside of one of the walls of the squash court and the microphone and higher-quality microphone placed on the ground adjacent.

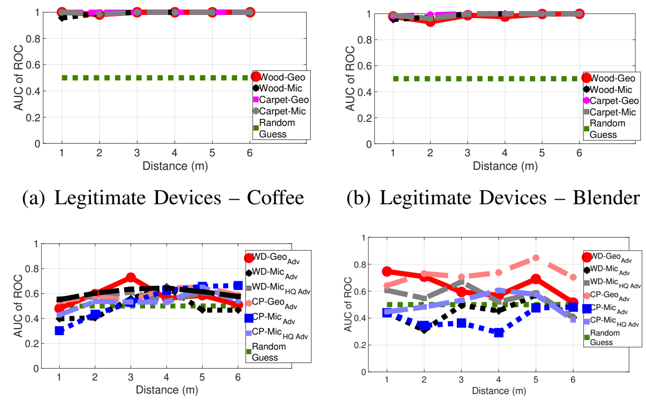
We have ten human subjects perform the following tasks:

knock on the door hosting the accelerometer, walk across the court (across the motion detector and the geophone) and around the table, brew coffee from the espresso machine on the table two times, one after another, walk back across the court, and knock on the door again before exiting. Hence, participants performed each activity of *knock*, *walk* and *coffee* twice per trial over ten different trials, providing a total of 600 activity traces. To evaluate sensor accuracy in event detection, we present Receiver Operating Characteristic (ROC) curves for each sensor used in this setup. The ROC curves plot the *true positive rate* ($TP_{rate}$) against *false positive rate* ($FP_{rate}$) and depict the ability of the different sensor modalities to detect events at varying threshold levels of signal amplitude.

Figure 11 depicts the resulting ROCs by event type. For each event, we depict the ROC of only those sensors whose modalities would allow them to possibly detect it. For example, a motion detector cannot detect a *coffee* event, and hence is ommitted from the *coffee* ROC. We find that all legitimate sensors have a high signal detection accuracy as most $Thr_{lower}$ yield a high $TP_{rate}$ with relatively low $FP_{rate}$. For example, *knock* ROC depicts good detection abilities for the inside geophone, microphone, and accelerometer, yielding large area under the curve (AUC), while the motion detector and power meter do not produce any signal for this event as expected (hence not shown). On the other hand, ROC curves for the $\mathcal{M}$s show relatively poor detection ability. We note that while all three events indicate that the higher quality attacker accelerometer and microphone generally perform better than their lower-quality counterparts, they are nevertheless unable to generate high $TP_{rate}$ without generating equally high $FP_{rate}$. At best, their curves follow a random guess trend. Some of the ROCs, especially for the attacker, appear to be increasing in a piecewise step fashion rather than a smooth concave trend. This is due to the nature of ambient noise in the system. As the signal detection threshold is lowered, noise is detected as true positive until the threshold is lowered enough such that other (lower) ambient noise is detected as false positives.

*2) Effect of Floor Types and Distances:* We next study the effect of the floor type on the detection accuracy of $\mathcal{LD}$s vs. $\mathcal{M}$s. We vary the floor type between wood and carpet (most common variations found in homes) as depicted in Figure 10(b). For each floor type, we trigger two events sufficiently spaced apart with no overlap in signal detection: a coffee maker brewing (the same machine used from Section VII-B1) and a blender (Cuisinart SPB-650 [14]) grinding. Since the accelerometer and motion detector cannot detect either, and the floor type does not affect the power meter, we study the sensing accuracy of the legitimate and attacker geophones and microphones. For each event type, the distance between the attacker/legitimate nodes and the event source (coffee maker/blender) is varied from 1-6m.

We show the resulting area under the ROC curve (AUC) for each sensor in Figure 12. Since the ambient noise inside the room is low (as is typical in homes), the legitimate geophone and microphone detect both the coffee and blender events




(a) Legitimate Devices – Coffee (b) Legitimate Devices – Blender

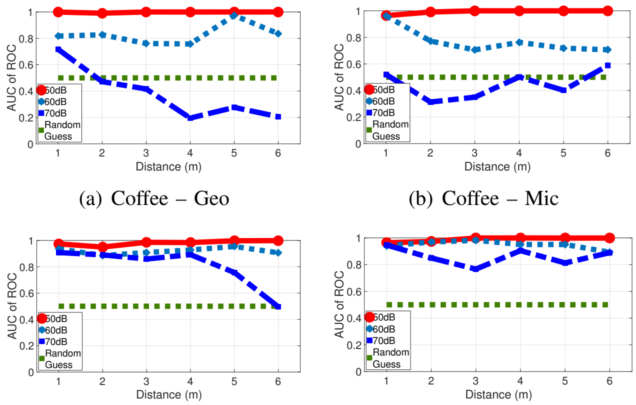
(c) Attacker Devices – Coffee (d) Attacker Devices – Blender

Fig. 12: As the distance between sensor devices and event source varies from 1-6m, the $\mathcal{LD}$s are consistently able to detect the event (with high AUC) while the $\mathcal{M}$s have a detection rate fluctuating around a random guess for carpet and wood alike. Since the blender is significantly louder with higher vibration than coffee brewing, the attacker's AUC is correspondingly higher.

with high accuracy for both floor types and across distances. The latter occurs due to the high signal to noise ratio inside the room even at longer distances from event source. On the other hand, the attacker's AUC fluctuates around 50% for carpet and wood alike across all distances for coffee events. Essentially, the attacker outside is contending with fluctuating noise levels due to the noisy surrounding, and is unable to detect these signals with accuracy any better than a random guess. For the blender event, the attacker geophone does show a slightly higher AUC, indicating better than random guess. This is as expected with the consistently higher sound and vibration caused by the blender as compared to the coffee machine. However, the attacker's AUC for blender, even for the geophone, barely exceeds 80% at best, and is significantly lower than the legitimate node's AUC.

*3) Effect of Background Noise and Distances:* While our analysis in Sections VII-B1 and VII-B2 show that $\mathcal{LD}$s consistently have high detection accuracy, the prevailing ambient noise inside the court was indeed low. We now study the degradation in event detection accuracy for the legitimate sensors with increasing background noise. Hence the background noise is varied between 50, 60 and 70 dB.

We show the resulting AUC for the legitimate microphone and geophone inside the room across distances of 1m to 6m from the event source in Figure 13. We see a clear trend of decreasing AUC across noise levels for all sensor types. As the ambient noise floor rises, the signal to noise ratio for the events degrades, incurring higher false positives for a given threshold of signal amplitude. At 50dB both the geophone and microphone are able to detect the coffee and blender with high AUC, with hardly any decline in detection rate from increasing distances to source. At 60 dB, the geophone's AUC for coffee is decreased compared to 50 dB, but remains mostly stable. The microphone, however, exhibits signifi-




(a) Coffee – Geo

(b) Coffee – Mic

(c) Blender – Geo

(d) Blender – Mic

Fig. 13: For events *coffee* and *blender* alike, increasing noise levels result in poorer detection accuracy even for devices inside, as expected. Since the coffee machine has a significantly weaker signal than the blender, the degradation in detection accuracy is steeper for *coffee event* as the distance from source and noise level increases.

cantly degraded performance as the distance from the coffee maker increases at 60dB. As seen from previous analysis, the inherently higher sound and vibration generated by the blender results in the both sensors continuing to perceive it with high accuracy. At 70 dB, the signal to noise ratio for the coffee event degrades enough at higher distances to make its detection effectively a random guess for both nodes. Even for the blender, we see the geophone's AUC start to suffer at higher distances. Most home environments where *Perceptio* is suitable might have instances of high background noise (e.g. music playing loudly for a few minutes), during which sensors inside might not be able to fingerprint successfully. But as long as the environment exhibits ambient noise levels below 70dB for the most part, the sensors are able to detect events successfully for fingerprint extraction.

### C. Key Establishment

*1) Fingerprint Similarity between Legitimate Devices:* While we demonstrated generally high event detection accuracy of legitimate devices, $\mathcal{LD}$s, under prevailing conditions inside the squash court in Section VII-B, this may not directly translate to satisfactory key establishment. This could be due to occasional detection errors, clustering errors, and relative temporal offsets in event detection between different sensor modalities. Hence, we evaluate our protocol in an end-to-end manner to demonstrate *Perceptio*'s ability to establish shared keys between $\mathcal{LD}$s (with heterogeneous modality) located within the physical boundary. To do so, we use real-world data to execute the *Perceptio* protocol and evaluate the fingerprint similarities $F_{sim}$ between device pairs. Specifically, we first generate a data stream of three thousand events – consisting of *knocking*, *walking*, *coffee*, and *ambient noise* – by randomly drawing samples from the data set described in Section VII-B1. Upon executing the protocol, we compute $F_{sim}$ for all seven feasible sensor pairs across $\mathcal{LD}$s, as depicted in Figure 14 (note that there are ten sensing modality-pairs possible, but $\{acc, mot\}$, $\{acc, pow\}$ and $\{mot, pow\}$ are omitted as none of the tested events can be sensed in common by these pairs). We illustrate two interpretations of the fingerprint similarity for each sensor pair. First, we depict the overall fingerprint similarity across all fingerprint comparisons. The large standard deviation in this first set of bars reflects the variation across fingerprints that will be used and those that will be discarded due to low similarity. Second, we depict the average fingerprint similarity $F_{sim}$ for only those fingerprints that are not discarded (i.e., those with similarity above the threshold). These are the fingerprints that actually contribute to secure key establishment and confidence. As depicted in Figure 14, all the sensor pairs that perceive at least one common event have high $F_{sim}$ after the thresholding.

*2) Confidence Score:* Another important aspect of *Perceptio* is its *Key Strengthening Process* discussed in Section V, which takes advantage of incremental growth in the confidence score (*ConfScore*) upon a successful iteration of key establishment protocol. Figure 15 depicts *ConfScore* of sensor pairs over time. As in the previous discussion, we depict the sensor pairs that perceive at least one event in common. The notion of time is depicted as the number of events arrivals in this figure, as more events arrive with more time (detailed modeling of event inter-arrival times and resulting time for entropy extraction is presented in Appendix D). From this figure, we have two important takeaways. First, sensor pairs that detect more events reliably and/or frequently in common exhibit a steeper increase in confidence. For example, $\{geo, mic\}$ pair perceives three events in common – *knock*, *walk*, and *coffee* – while $\{acc, mic\}$ perceives only the *knock* event in common. Hence we see that as more events arrive,*ConfScore* of $\{geo, mic\}$ pair increases faster than that of $\{acc, mic\}$. The pairs that do not reliably or frequently perceive a common event, such as $\{geo, mot\}$ have much slower increase in *ConfScore*. Second, it is important to note that *ConfScore* never decreases over time. Upon fingerprint mismatches (which contributes to lowered average $F_{sim}$ in the first bar graphs of each sensor pairs depicted in Figure 14), the *ConfScore* levels off at the current state until the next successful fingerprint matching occurs. This means that any fingerprint mismatches – due to detection and/or clustering errors – *do not degrade* the key establishment process, but simply takes longer.

*3) Fingerprint Similarity between Attacker and Legitimate Devices:* It is evident from the attacker's event detection ROC studied in Figures 11(d) 11(e) 11(f) that the $\mathcal{M}$s can hardly perform better than random guess. Further, given that requisite clustering also incurs some errors, it is expected that the likelihood of an $\mathcal{M}$ achieving a high $F_{sim}$ with an $\mathcal{LD}$ can be no better than a random guess. We nevertheless evaluate this by further granting two unfair advantages in favor of the attacker. First, we assume that the $\mathcal{M}$s are capable of yielding less errors in event detection. There are two types of errors in event detection – *insertion* and *deletion errors*, each represented by $FP_{rate}$ and $TP_{rate}$

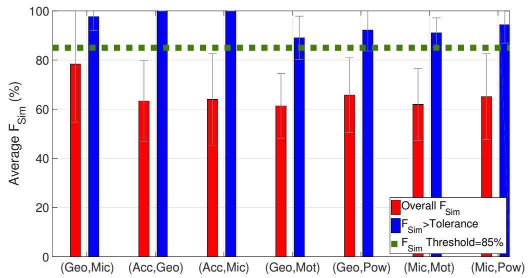



Fig. 14: We verify that $\mathcal{LD}$s that sense common events are indeed able to pair with high fingerprint similarity. Occasional inaccuracies in event clustering and temporal offsets in event detection cause the average fingerprint similarity between modality-pairs to be around 65% with a high variance. However, even at 85% similarity threshold for successful pairing, all sensor modalities manage to establish keys within a few successful tries, with low variance.

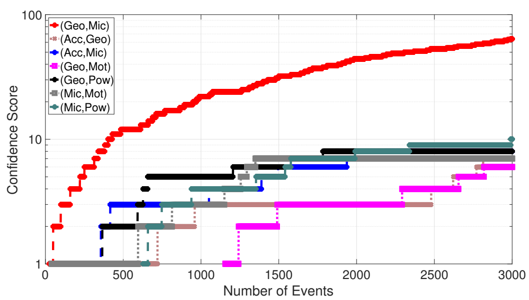



Fig. 15: We study the *key strengthening* process by observing the increase in confidence score for each established legitimate sensor pairing as the number of encountered events in the environment increases. Modalities such as geophone and microphone that are able to simultaneously sense most of the occurring events exhibit a much steeper increase in confidence scores as compared to pairings such as {*geo, mot*} that sense relatively fewer events in common.

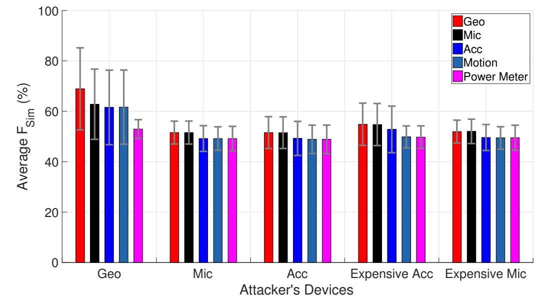



Fig. 16: We present a simulated study of $F_{sim}$ for $\mathcal{M}$s attempting to pair with $\mathcal{LD}$s. Even with overestimated capabilities of the attacker, average of all $F_{sim}$ is only at 55%, bar the expensive geophone (around 70%), but nevertheless sufficiently below the *tolerance* line of 85% set in Figure 14.

respectively. We only considering errors due to deletion, and assume that the $\mathcal{M}$s do not yield any insertion errors – i.e., yielding high $TP_{rate}$ with no $FP_{rate}$. From the ROC curves aforementioned in Section VII-B, we choose the best possible $TP_{rate}$ for each attacker sensor that corresponds to $FP_{rate} = 50\%$, but replace the $FP_{rate}$ to 0%. Second, we assume that the attacker has 100% clustering accuracy.

While these are unrealistic advantages, we evaluate $F_{sim}$ with such assumptions to account for the chance possibility that the attacker may detect events at a higher accuracy or have access to better clustering methods. Hence, the two advantages provide an optimistic scenario for the attacker.We evaluate fingerprint similarities between $\mathcal{M}$s and $\mathcal{LD}$s with a simulated stream of events by exhaustively searching for best matching fingerprints. Figure 16 depicts the reported values, with a maximum value of 70% between the attacker and legitimate geophones. Recall from Figure 14 that we draw the requisite similarity threshold at 85%. Hence the attacker's best case $F_{sim}$, even with the unfair advantages, are sufficiently below the tolerance level, demonstrating that *Perceptio* succeeds in thwarting the attack.

## VIII. Deployment Considerations

We now discuss practical considerations when deploying *Perceptio* in smart homes.

**Simultaneous Events.** While we present experiment evaluations with a single event per time period and background noise, this may not always be true in real life, as multiple events may occur simultaneously (e.g., coffee making while walking). In such cases, we have seen that the concurrent events will produce an overlapping signal and either be clustered as a separate event type or mismatch errors will occur leading to a longer time to reach the confidence threshold. To test our hypothesis, we conducted a preliminary experiment with two events – *coffee* making and generating *footsteps* (walking in place) – occurring simultaneously, while the sensors were located 1m away from the event sources. We then kept the locations of sensors and the coffee machine static, while varying the stepping positions from 1-6m. Figure 17 depicts an example plot of signals captured at 1-4m distances between the simultaneous events. At 1m distance, the signals differ significantly from those generated by the coffee machine and footsteps in isolation, while at 4m distance, the signal characteristics are closer to those of a coffee machine in isolation. We see that many overlapping signals will lead to new event clusters of their own, rather than with existing event types.

**Ad-hoc networking.** *Perceptio* provides a novel solution to secure ad-hoc connectivity among IoT devices, *without* the need for a trusted home gateway. Many applications may benefit from such ad-hoc networks due to reduced communication and computational overhead, as it no longer requires going through a central gateway or cloud. In fact, there is a push in the industry to shift from star to mesh topologies, as seen by industry activities such as Thread [30].

**Resourceful attackers.** Through our evaluation, we demonstrated the difficulty of the attacker succeeding in Shamming–Eavesdropping attack due to the need to consistently detect events inside the home. However, as defined in our threat model in Section III, if an attacker launches

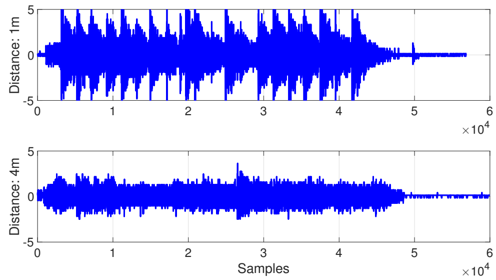



Fig. 17: When events *coffee* and *footsteps* occur simultaneously, the combined signals are distorted significantly enough to possibly cluster into a new event type of its own. However, the magnitude of this distortion also depends on distance between event sources.

Shamming–Signal Injection attack by creating and injecting events from outside that are consistent and loud enough to be sensed from legitimate devices inside the house, the attacker may succeed in fooling the legitimate device to pair. However, due to the same attenuation factor that protects inside events from the external attacker, it would be difficult for inside devices to consistently detect outside events unless they are *extremely loud*, otherwise the fingerprints would not match. To make this attack harder, *Perceptio*'s *Key Strengthening Process* requires multiple iterations that take enough time that such injections would be easily noticed by human users, making the attack extremely risky and likely impractical. Furthermore, our threat model also defines Shamming–Sensor Spoofing attack by injecting spoofing signals to sensors of legitimate devices similar to prior work [70], [72], [67], [68]. While such attacks may still be possible, *Perceptio* would cluster injected signals into another event type. Hence, the attacker would only slow down the *Key Strengthening Process*. Furthermore, such injection attacks require a high amplitude signal to be exerted to the sensor, which is rather difficult in our setting as signals attenuate significantly through the wall as our experiments have shown.

## IX. Limitations

We present some of the limitations of our work and discuss how they can be improved in future.

**Devices located in different rooms.** *Perceptio* is potentially unable to establish trust between valid devices located in different rooms of a smart home. A possible remedy is to introduce a *bridging device* in each room to facilitate cross-room connections. A bridging device would be like any other IoT device, but with the additional functionality for human-in-the-loop pairing. For example, two infrared- and NFC-enabled motion detectors in different rooms may be first manually paired by the user (e.g., via NFC tagging with a smartphone) and then deployed to each room. Devices in each room can leverage the Transitivity-of-Trust (ToT) protocol (Section V) via the bridging devices to pair with devices in other rooms. Manually bootstrapping bridging devices is reasonable because there are only as many bridging devices as rooms in the home. This is analogous to distributed WiFi systems that use multiple APs to provide or enhance connectivity through a large home [28], [62], [21].

**Calibration.** *Perceptio* depends on sensor calibration and determination with respect to appropriate threshold values presented in Section VI-A. Thresholding is important to helps distinguish signals from noise and is thus critical with respect to factors such as sensor placement, sampling rate, and events in the environment. Hence, in practice, *Perceptio* would require a calibration phase by allowing the IoT devices to perform local sensor calibrations for a given amount of time prior to starting the *Perceptio* protocol. Device manufacturers could also provide course-grained pre-calibrated settings.

**Public and Shared Spaces.** *Perceptio* is based on the assumption that physical boundaries draw natural barriers between the legitimate devices and the attacker's device outside, which may not hold for public spaces such as public libraries or shopping malls. However, with further work on fine tuning thresholding parameters, *Perceptio* can be extended from single family housing to other multi-tenant private office buildings with existing access control policies.

**Frequency of activity vs. Pairing time.** The pairing time between devices is directly proportional to the frequency of activities in the house. However, there may be households with less family members and thereby decreased sensor activity, leading to undesirably long pairing times. In such cases, users may introduce a *signal injecting device* (as presented in Section IV-A) for faster convergence. This solution, however, trades procurement cost and usability for speed.

## X. Conclusion

We propose *Perceptio* for autonomous, secure pairing of IoT devices using context information from embedded sensors. The novelty of *Perceptio* stems from its ability to address the difficult challenge of context-based pairing across devices equipped with different types of sensors. *Perceptio* achieves this goal by abstracting sensor measurements and using timing information as an invariant property to generate context fingerprints as a source of shared entropy for cryptographic key agreement. We demonstrate through proof-of-concept experiments that *Perceptio* is able to securely pair heterogeneous sensing devices co-located within the same physical boundary, while rejecting potential attacker devices placed outside.

## XI. Acknowledgements

This research was supported in part by the National Science Foundation (under grants CNS-1645759 and CMMI-1653550), University Transportation Center, Intel and Google. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of CMU, NSF, or the U.S. Government or any of its agencies.

## References


[1] *MetaSensor: Meet Sensor-1*. https://www.metasensor.com/.

[2] *Samsung SmartThings Motion Sensor*. https://www.smartthings.com/products/samsung-smartthings-motion-sensor.

[3] Knowles Acoustics. *MD9745APA-1 Product Specification*. https://www.digchip.com/datasheets/parts/datasheet/922/MD9745APA-1-pdf.php.

[4] Arduino AG. *Wireless SD Shield*. https://www.arduino.cc/en/Main/ArduinoWirelessShield.

[5] Arduino AG. *Arduino/Genuino UNO*, 2015. https://www.arduino.cc/en/Main/arduinoBoardUno.

[6] W.F. Alliance. Wi-fi protected access: Strong, standards-based, interoperable security for today's wi-fi networks. *Retrieved March*, 1:2004, 2003.

[7] Amazon. *Amazon Echo*. https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E.

[8] Lorraine E Bahrick, Robert Lickliter, and Ross Flom. Intersensory redundancy guides the development of selective attention, perception, and cognition in infancy. *Current Directions in Psychological Science*, 2004.

[9] Dirk Balfanz, Dirk Balfanz, Glenn Durfee, Glenn Durfee, Rebecca E. Grinter, Rebecca E. Grinter, D. K. Smetters, D. K. Smetters, Paul Stewart, and Paul Stewart. Network-in-a-box: How to set up a secure wireless network in under a minute. In *Usenix Security*, 2004.

[10] Elaine Barker. Recommendation for Key Management. NIST Special Publication 800-57, 2016.

[11] Bosch. *ISC-BPR2 - Blue Line Gen2 PIR Motion Detectors*. http://resource.boschsecurity.com/documents/BlueLine_Gen_2_Data_sheet_enUS_2603228171.pdf.

[12] U.S. Census Bureau. *Average number of people per household in the United States from 1960 to 2016*, 2017. https://www.statista.com/statistics/183648/average-size-of-households-in-the-us/.

[13] IO Sensor Nederland b.v. *SM-24 Geophone Element*. https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/SM-24\%20Brochure.pdf.

[14] Cuisinart. *SPB-650 Blender*. http://www.cuisinart.com/products/blenders/spb-650.html.

[15] Deloitte. *The Digital Predictions 2015*, 2015. The Deloitte Consumer Review.

[16] Analog Devices. *ADXL335 Datasheet*. http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf.

[17] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 1976.

[18] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*

[19] Morris J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, 2015. National Institute of Standards and Technology (NIST) Federal Information Processing Standards Publication 202.

[20] Ecolink. *Z-Wave Motion Detector with Pet Immunity*. http://www.discoverecolink.com/product/pirzwave2-eco/.

[21] Eero. *An Entirely New Approach to Home WiFi*. https://eero.com/.

[22] Samsung Electronics. *SmartTV Voice Control*. http://www.samsung.com/ph/smarttv/voice_control.html.

[23] Fibaro. *Motion Sensor*. http://www.fibaro.com/us/the-fibaro-system/motion-sensor.

[24] Center for Advanced Studies in Adaptive Systems. *WSU CASAS Datasets*. http://ailab.wsu.edu/casas/datasets.html.

[25] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. In *RSA Cryptobytes*, 2004.

[26] D.V. Giri and F.M. Tesche. *Modeling of Propagation Losses in Common Residential and Commercial Building Walls*, 2013. Interaction Note 624.

[27] Google. *Google Home*. https://home.google.com/.

[28] Google. *Google WiFi: Home Wi-Fi, simply solved*. https://madeby.google.com/wifi/.

[29] Bluetooth Core Specification Working Group. *Bluetooth simple pairing Whitepaper*, 2006. Bluetooth SIG Whitepaper V10r00.

[30] Thread Group. *What is Thread?* http://threadgroup.org/What-is-Thread/Overview.

[31] Jun Han, Albert Jin Chung, and Patrick Tague. Pitchin: Eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2017.

[32] Jun Han, Madhumitha Harishankar, Xiao Wang, Albert Jin Chung, and Patrick Tague. Convoy: Physical context verification for vehicle platoon admission. In *18th International Workshop on Mobile Computing Systems and Applications (HotMobile)*, Feb 2017.

[33] Jun Han, Abhishek Jain, Mark Luk, and Adrian Perrig. Don't sweat your privacy: Using humidity to detect human presence. In *Proceedings of 5th International Workshop on Privacy in UbiComp (UbiPriv'07)*, September 2007.

[34] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. Adaptive event detection with time-varying poisson processes. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[35] Honeywell Inc. *5870API Wireless Indoor Asset Protection*, 2017. http://www.goamt.com/wp-content/uploads/2015/08/5870API_WIRELESS-INDOOR-ASSET-PROTECTOR_AMT.pdf.

[36] Insteon. *Low Voltage/Contact Closure Interface*. http://www.insteon.com/io-module.

[37] P3 International. *Kill-A-Watt P4400*. http://www.p3international.com/products/p4400.html.

[38] Slawomir Jasek. *Gattacking Bluetooth Smart Devices*, 2016. Black Hat Briefings.

[39] Larry P. Jedele. *Energy Attenuation Relationships from Construction Vibrations*, 1985. Vibration Problems in Geotechnical Engineering. ASCE Convention in Detroit, Michigan.

[40] Michael O. Jewell, Enrico Costanza, and Jacob Kittley-Davies. Connecting the things to the internet: An evaluation of four configuration strategies for wi-fi devices with minimal user interfaces. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2015.

[41] A. Juels and M. Sudan. A fuzzy vault scheme. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 408–, 2002.

[42] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 1996.

[43] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5(6):734–749, 2009.

[44] Cao Gadgets LLC. *Wireless Sensor Tag System: Monitor Everything from the Internet*, 2017. http://www.wirelesstag.net.

[45] T. Mahmud, M. Hasan, A. Chakraborty, and A. K. Roy-Chowdhury. A poisson process model for activity forecasting. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

[46] Christopher Mazur. *Physical Characteristics of Housing: 2009–2011*, 2013. American Community Survey Briefs – U.S. Census Bureau, Department of Commerce.

[47] J.M. McCune, A. Perrig, and M.K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *Security and Privacy, 2005 IEEE Symposium on*, pages 110–124, May 2005.

[48] Jonathan McCune, Adrian Perrig, and Michael Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[49] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security Privacy*, 2009.

[50] Blue Microphones. *Yeti*. http://www.bluedesigns.com/products/yeti.

[51] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. Context-based zero-interaction pairing and key evolution for advanced personal devices. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2014.

[52] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, 2010.

[53] Samantha Murphy. *Samsung: By 2020, all of our products will be connected to the web*, 2015. http://mashable.com/2015/01/05/samsung-internet-of-things/\#c3_K6GzxJ65N.

[54] Nestle. *Nespresso Pixie Carmine*. https://www.nespresso.com/us/en/order/machines/original/Nespresso-Pixie-Dark-Red.

[55] Nexia. *Schlage Motion Sensor*. http://www.nexiahome.com/compatible-products/schlage-home-motion-sensor/.

[56] Notion. *Home awareness, simplified. Monitor your home with a single sensor, wherever you are.* http://getnotion.com/.

[57] Shijia Pan, Amelie Bonde, Jie Jing, Lin Zhang, Pei Zhang, and Hae Young Noh. Boes: Building occupancy estimation system using sparse ambient vibration monitoring. In *Proc. SPIE 9061*, 2014.

[58] Shijia Pan, Ningning Wang, Yuqiu Qian, Irem Velibeyoglu, Hae Young Noh, and Pei Zhang. Indoor person identification through footstep induced structural vibration. In *International Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, 2015.

[59] Panasonic. *MP Motion Sensor NaPiOn*. https://www3.panasonic.biz/ac/e/control/sensor/human/napion/.

[60] Xerox PARC. *U.S. Department of Energy Chooses PARC to Develop Wireless Peel-And-Stick Sensors; New Technology Helps Fuel IoT Growth*. http://xerox.bz/2nTDF8G.

[61] Colin Percival. Stronger key derivation via sequential memory-hard functions. 2009. http://www.tarsnap.com/scrypt/scrypt.pdf/.

[62] Plume. *Plume WiFi*. https://www.plumewifi.com/.

[63] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. Heart-to-heart (h2h): authentication for implanted medical devices. In *ACM SIGSAC conference on Computer and Communications Security (CCS)*. ACM, 2013.

[64] J. Salowey, A. Choudhury, and D. McGrew. AES Galois Counter Modes (GCM) Cipher Suites for TLS. RFC, 2008.

[65] Dominik Schurmann and Stephan Sigg. Secure communication based on ambient audio. *IEEE Transactions on Mobile Computing*, 12(2):358–370, February 2013.

[66] Freescale Semiconductor. *MMA1270KEG Datasheet*. http://www.nxp.com/docs/en/data-sheet/MMA1270KEG.pdf.

[67] Hocheol Shin, Yunmok Son, Youngseok Park, Yujin Kwon, and Yongdae Kim. Sampling race: Bypassing timing-based analog active sensor spoofing detection on analog-digital systems. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[68] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[69] Samsung SmartThings. *Multipurpose Sensor*. https://support.smartthings.com/hc/en-us/articles/213496343.

[70] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[71] Ahren Studer, Timothy Passaro, and Lujo Bauer. Don't bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In *ACSAC '11: Proceedings of the 27th Annual Computer Security Applications Conference*, December 2011.

[72] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, 2017.

[73] Hasan S. Ulusoy, Erol Kalkan, Jon Peter B. Fletcher, Paul Friberg, W. K. Leith, and Krishna Banga. Design and implementation of a structural health monitoring and alerting system for hospital buildings in the united states. In *World Conference on Earthquake Engineering*, 2012.

[74] Xiao Wang and Patrick Tague. Non-invasive user tracking via passive sensing: Privacy risks of time-series occupancy measurement. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, AISec '14, 2014.

## APPENDIX

This section presents the detailed descriptions of sensors in smart home IoT devices, the *Perceptio* cryptographic protocol and its security analysis, and simulation of entropy extraction.

### A. Sensors in Smart Home IoT Devices

Numerous IoT devices for smart homes are already commercially existing. We introduce some of the most prevalent sensing modalities, and their use in real-world commercial products as well as research prototypes. Table I summarizes them. Smart speakers (e.g., Amazon Echo [7] and Google Home [27]) and TVs [22] are equipped with with microphones used for voice-activated controls. Motion detectors used to monitor movements, are equipped with *Passive Infrared (PIR)* sensors [55], [20], [2]. We also find many *on-object sensing devices* in the market that monitor the status of the object it is attached to [56], [69], [1], [44], [20], [35]. For example, when attached on a door, it monitors events such as *door open/close* as well as *knocking*. These devices mainly utilize *accelerometers* by performing simple signal processing to output object status information. Power meters that measure the electrical usage of the appliance it is attached to is also gaining popularity along with the emergence of smart grid [37], [36]. Geophones are seismic sensors that are gaining both industry and academic attention for their applicability in building health monitoring [73] as well as occupant monitoring via gait detection [58], [57]. We exploit the heterogeneous sensing modalities of these prevalent IoT device to prove that they are co-located within a physical boundary.

### B. Perceptio Protocol Details

We present the details of the cryptographic protocol described in Section V. In the *Key Agreement Phase*, $A$ and $B$ generates fingerprints $\{F_{A_i}, i = 1,\ldots,p\}$ and $\{F_{B_j}, j = 1,\ldots,q\}$ for the $p$ and $q$ observed event clusters. Device $A$ then encodes a randomly generated secret key $k_i$ using each fingerprint $F_{A_i}$, $i = 1,\ldots,p$, to create a set of commitments as $C_{A_i} = F_{A_i} \ominus ENC(k_i)$, where $\ominus$ is subtraction in a finite field, $\mathbb{F}^n$, equivalent to an XOR operation, and $ENC(\cdot)$ is the encoding operation for an error correcting code (e.g., Reed-Solomon). $A$ then sends $\{(C_{A_i}, h(k_i)), i = 1,\ldots,p\}$ to $B$, where $h(\cdot)$ is a collision-resistant hash function, which discloses no information about the keys $k_i$ or the fingerprints $F_{A_i}$. Upon receiving the set of commitments from $A$, device $B$ attempts to open the commitment to acquire any one of the original secrets $k_i$ using its fingerprints $F_{B_j}$. $B$ computes $\hat{k}_{i,j} = DEC(F_{B_j} \ominus C_{A_i})$ for all $i, j$ pairs, where $DEC(\cdot)$ is the complementary decoding function, such that $DEC(ENC(m) \ominus \nu) = m$ for a bit string $m$ whenever the Hamming weight ($l_1$ norm) $|\nu|_1$ is within the code's decoding capability $t$. If $B$ finds an $i, j$ pair such that $h(k_i) = h(\hat{k}_{i,j})$, then it most likely found a fingerprint match,$F_{A_i} \approx F_{B_j}$. There are many protocol variations at this point, but we choose one in which $B$ needs to find only one such pair, so not all $pq$ values need to be computed if a match is found. At this point, $B$ can use a key derivation function $KDF(\cdot)$ [61] to create a shared symmetric key as $k_{AB} = KDF(\hat{k}_{i,j})$, though $A$ is unaware of this key at this point (Figure 18 Steps 1-4).

To allow $A$ to generate the matching symmetric key $k_{AB}$ and verify it actually matches the key generated by $B$, both $A$ and $B$ further participate in the *Key Confirmation Phase*. $B$ generates a random nonce $n_B$ and transmits $\beta$, where $H(\hat{k}_{i,j})$ equals to $H(k_i)$ and $M_k(m)$ represents a keyed message authentication code (MAC) of message $m$ using key $k$. $A$, upon receiving this message, first identifies the key, $k_i$, from $H(k_i)$. If found, $A$ derives the shared key as $k_{AB} = KDF(k_i)$ for the matching $i$. $A$ then performs

| Sensor | Microphone | PIR | Accelerometer | Power Meter | Geophone |
|---|---|---|---|---|---|
| **IoT Device Category** | Voice Recognition (Smart Speakers [7], [27], Smart TV [22]) | Motion Detector [55], [20], [2] | Door Knock / Open / Close Monitor [56], [69], [1], [44], [20], [35] | Electricity Usage Monitor [37], [36] | Structural Health / Footstep Monitor [73], [58] |

TABLE I: We provide a table of sensors embedded in commercial IoT products and research prototypes.

a MAC verification with $k_{AB}$ and if successful, it also generates a nonce, $n_B$, and transmits to $B$, $\alpha$. $B$, upon receiving $\alpha$, performs MAC verification to verify that $A$ also generated the same key $k_{AB}$. If successful, device $A$ and $B$ successfully computed a shared symmetric key for one round (Figure 18, Steps 5- 8).

***» Key Agreement Phase «***

1. $A$ : $F_{A_i} = extractFs(ctx, t_F)$; $i = 1, \ldots, p$
   $B$ : $F_{B_j} = extractFs(ctx, t_F)$; $j = 1, \ldots, q$
2. $A$ : $k_i \xleftarrow{R} KGen(1^\gamma)$
   $C_{A_i} = F_{A_i} \ominus ENC(k_i)$
3. $A \rightarrow B$ : $C_A = C_{A_1} || H(k_1), .., C_{A_p} || H(k_p)$
4. $B$ : $\hat{k}_{i,j} = DEC(F_{B_j} \ominus C_{A_i})$
   Verify $H(k_i) \stackrel{?}{=} H(\hat{k}_{i,j})$; Aborts if fails
   Creates $\hat{k}_{AB} = KDF(\hat{k}_{i,j})$

***» Key Confirmation Phase «***

5. $B \rightarrow A$ : $\beta = H(\hat{k}_{i,j}) || n_B || M_{\hat{k}_{AB}}(n_B)$,
   where $n_B \xleftarrow{R} \{0,1\}^\eta$
6. $A$ : Creates $k_{AB} = KDF(k_i)$;
   $M_{\hat{k}_{AB}}(n_B) \stackrel{?}{=} M_{k_{AB}}(n_B)$; Aborts if fails
7. $A \rightarrow B$ : $\alpha = n_A || M_{k_{AB}}(n_B || n_A)$,
   where $n_A \xleftarrow{R} \{0,1\}^\eta$
8. $B$ : $M_{k_{AB}}(n_B || n_A) \stackrel{?}{=} M_{\hat{k}_{AB}}(n_B || n_A)$;
   Aborts if fails

Fig. 18: Details of *Perceptio* key agreement and confirmation protocol using contextual information

### *C. Security Analysis*

We now present the analysis of *Perceptio*'s cryptographic protocol, namely presenting how an attacker would try to launch attacks to compromise the shared secret. Specifically, the attacker's goal is to acquire $k_i$ generated by $A$ in Figure 18 Step 2. We analyze two types of attacks that an attacker may launch to achieve the aforementioned goal – *(1) bruteforcing* and *(2) eavesdropping* attacks.

*(1) Bruteforcing attack.* The attacker first tries to directly bruteforce the key, $k_i$ by attempting to perform dictionary attack on the hash, $H(k_i)$, which is transmitted together with $C_{A_i}$ in Figure 18 Step 3. As long as the length of the cryptographic hash function ($H(\cdot)$), $l_{H(\cdot)}$, is longer than $l_{NIST}$ bits, bruteforce attack is computationally infeasible (i.e., $l_{H(\cdot)} \geq l_{NIST}$ bits). We leverage the state-of-the-art secure cryptographic hash function such as SHA-3 [19], which is well above $l_{NIST}$ bits. We define $l_{NIST} = 112bits$, as recommended by NIST [10].

*(2) Eavesdropping attack.* A more sophisticated attacker pretends to be a legitimate device placed within the physical boundary by trying to open the commitment. The attacker launches an *eavesdropping attack* to try to capture some of the events by placing his/her devices just outside of the physical boundary. Hence, these devices may capture some of the signals, depending the transmission media as well as the amplitude of the original signal. Hence, rather than performing a bruteforce attack with no known information, the attacker has more information at guessing the fingerprint, which can be decoded with $DEC(\cdot)$, which in turn leads to less amount of computations to acquire $k_i$.

We denote $l_{eaves}$ as the number of bits of the fingerprint that the attacker knows as a result of the eavesdropping attack. Hence, we denote $l_{bf}$ as the number of bits the attacker needs to bruteforce in order to successfully know $l_{tol}$ bits in order to succeed in the attack, such that $l_{bf} = l_{tol} - l_{eaves}$. Hence, the attacker's success probability is $P(Adv) = 1$ with computational complexity, $Cpx$, as following:

$$
\begin{aligned}
Cpx &= p2^{l_{bf}}(Ops + \ominus + DEC(\cdot) + H(\cdot) + V_{H(\cdot)}) \\
&\approx O(2^{l_{bf}})
\end{aligned}
$$

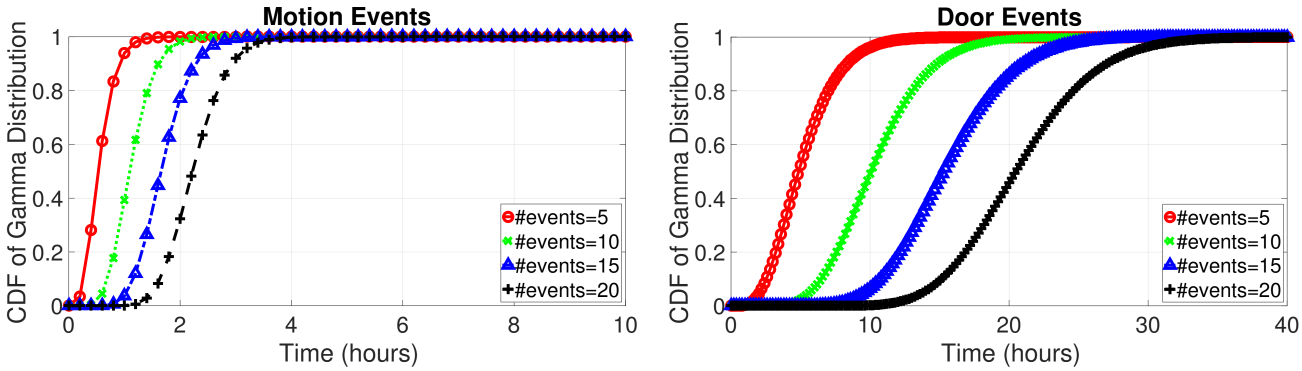
where $p$ is the number of $F$s and $V_{H(\cdot)}$ is hash verification. $Cpx$ is computationally infeasible if $l_{bf} \geq l_{NIST}$. Hence the gain from eavesdropping, $l_{eaves}$ should be bounded by $l_{eaves} = l_{tol} - l_{NIST}$.

### *D. Evaluating Entropy Extraction*

We now evaluate the required time to extract $l_F$ (i.e., length of fingerprint) to ensure sufficient entropy (e.g., 128 bits). As discussed in Section IV-B, $F$ is created by concatenating the time intervals of consecutive events of same cluster type (e.g., series of knocking events).

*1) Modeling the Arrival Time:* We follow the traditional approaches of modeling event arrivals as a Poisson process [34], [45]. We define $S_n$ as the waiting time until the $n^{th}$ event, assuming that $n$ events yields $l_F$ bits of fingerprint. We define $T_i$ as the sequence of inter-arrival times for $i = 1, 2, ...$, which can also be described as i.i.d. exponential random variables. Furthermore, the probability density function of $S_n$ has a gamma distribution with average arrival rate $\lambda$, number of events $n$, and time $t$ as depicted in Equation 2.

$$
S_n = \sum_{i=1}^{n} T_i, \quad n \geq 1, \qquad f_{S_n}(t) = \lambda e^{-\lambda t}\frac{(\lambda t)^{n-1}}{(n-1)!}. \quad (2)
$$




(a) Motion (b) Door

Fig. 19: Cumulative probability distribution of *motion* and *door opening* events modeled after real world smart home data collected for two months

The corresponding expected time of $n^{th}$ event, $E(S_n)$, is depicted in Equation 3. We also define bit rate which is the effective rate of the generating the $l_F$ fingerprint bits in a time duration of $E(S_n)$, capturing the effective rate of generating useful fingerprint bits. The bit rate is modulated by a correction factor, $\rho$, which is proportional to the detection rate of the events. We note that the units of the bit rate can be measured in bits per second, but in many practical scenarios it may be more meaningful to express this value in bits per hour.

$$E(S_n) = \frac{n}{\lambda}, \qquad BitRate = \frac{l_F\,\rho}{E(S_n)} = \frac{\lambda\, l_F\,\rho}{n} \tag{3}$$

*2) Evaluation Using a Real-world Smart Home Dataset:* To ensure the practicality of our analysis, we analyze a real-world smart home data set, publicly available from CASAS online repository [24]. We analyze two sets of sensor data collected for two months (i.e., over 1450 hours of data): a motion detector used to monitor movement in the home, and a door sensor to monitor door open/close activities. Specifically, we extract mean arrival rate of the two events, $\lambda_{motion}$ and $\lambda_{door}$, to be 8.85 events/hour and 0.96 events/hour, respectively. We note that the average was computed from the users' daily activities only. This reflects the practical use case of *Perceptio*, as the system will not extract much entropy at night due to stagnant event occurrences. Using these values, we plot a cumulative probability density function (CDF) and vary $n$ and $t$. Figure 19 (a) and (b) depict the CDF of the two types of events, respectively. The results are intuitive as the plots demonstrate that for more number of $n$ events, the longer $t$ is required to reach a high probability. Furthermore the two figures of motion and door events depict clear contrast, as the door events require much longer time to reach a high probability. We note that this analysis is an optimistic approach as we assume perfect detection accuracy (i.e., $\rho = 1$) for simplicity of the analysis.

For example, assume that it takes 20 events to yield $l_F = 128$ *bits* of the fingerprint, then using (Equation 3), $n = 20$ events arrive in about 2.3 hours for motion events, as opposed to 20.8 hours for door events. Hence, the corresponding bit rate for the two events are $BitRate_{motion}$ of 56.6 bits/hour and $BitRate_{door}$ of 6.1 bits/hour. We note that the bit rate would potentially increase if there were more occupants in the house as opposed to a single resident case from this data set. (For example, the average number of occupants in a home in the United States is 3.14 persons [12]).