A Data Hiding Approach for Sensitive Smartphone Data

Chu Luo[1], Angelos Fylakis[2], Juha Partala[2], Simon Klakegg[1], Jorge Goncalves[1], Kaitai Liang[3], Tapio Seppänen[2], Vassilis Kostakos[1]

[1]Center for Ubiquitous Computing, University of Oulu, Finland

[2]Center for Machine Vision and Signal Analysis, University of Oulu, Finland

[3]Department of Computer Science, Aalto University, Finland

[1,2]firstname.lastname@ee.oulu.fi, [3]kaitai.liang@aalto.fi

ABSTRACT

We develop and evaluate a data hiding method that enables smartphones to encrypt and embed sensitive information into carrier streams of sensor data. Our evaluation considers multiple handsets and a variety of data types, and we demonstrate that our method has a computational cost that allows real-time data hiding on smartphones with negligible distortion of the carrier stream. These characteristics make it suitable for smartphone applications involving privacy-sensitive data such as medical monitoring systems and digital forensics tools.

Author Keywords

Smartphones; ubiquitous computing; privacy protections; digital signal processing; mobile and wireless security.

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

We present and evaluate a data hiding method for smartphone sensing, which enables sensing applications to encrypt and embed sensitive data or identification codes within other data streams of a smartphone. Our method is motivated by the increasing diversity of sensor data that mobile devices generate, and the growing ecosystems of services that store, process, and share this data.

The constellation of personal devices that we regularly use now includes smartphones, smartwatches, tablets, fitness sensors, and a variety of domestic appliances. These devices contain a growing set of increasingly sophisticated sensors which improve interaction and provide new services. The richness and volume of this data has given rise to research opportunities for the UbiComp community and beyond. For instance, research often demonstrates how smartphone data that we previously discarded as noise can actually contain valuable information [1]. Furthermore, research on quantified self, self-monitoring, and e-health aims to harness the data that our devices generate.

This trend has motivated scientists across academia and industry to build platforms that collect, analyse, visualise, and share increasing amounts of end-user data generated from personal devices. It has become a norm for studies to instrument personal devices of volunteer participants (recruited both in-person or via app-stores), and much of this data may eventually become available for other scientists or the public in general. Initiatives such as Crawdad, Crowdsignals, and Nokia's Lausanne Data Collection Campaign are examples of how smartphone “traces” may be shared after the completion of an experiment. Similarly, the quantified-self movement has given rise to a large number of platforms where users may upload their health-related sensor data. While certain platforms may be free or come with associated costs, users typically share their data in exchange for a service.

Often, multi-stream data from a user’s device is treated as a coherent data unit. For instance, a typical experiment may simultaneously collect accelerometer, heart rate, and GPS data. This set of sensor streams may then be uploaded to a server for analysis, visualisation and sharing. This approach to “bundling” sensor streams has two important downsides. Firstly, it treats all sensor streams unilaterally, overlooking the unique privacy aspects of each individual sensor stream. For instance, accelerometer data may not be as sensitive as GPS data. Secondly, users practically relent control of their data once it leaves their devices.

Our work proposes a data hiding approach to address the privacy needs that arise when users share smartphone sensor data with scientists and platforms. Crucially, our technique is transparent, meaning that it is compatible with existing platforms and tools. As a result, users can maintain control of their sensitive data even after sharing it through online platforms, without having to give up those services. Additionally, the technique allows us to verify the integrity of embedded sensitive data, for example to confirm that no tampering has taken place in the form of record removal, decimal rounding, or filtering. Finally, our technique allows users to prove ownership of sensor or healthcare data that they have shared, and as such provides spoof resistance against tampered medical sensor data [31].

BACKGROUND

Data hiding is an application domain of digital watermarking techniques [7]. Traditionally, watermarks are found in official documents, and carry information about the object in which they are found. Watermarks are designed in way so that they are difficult to reproduce or counterfeit [12]. For example, banknotes have watermarks in the form of figures that become visible only under certain conditions.

The practice of watermarking can be defined as imperceptibly altering an object to embed a message about it [12]. Embedding a watermark *w* into a host object *C* produces a new object C_w, such that *w* can be reliably located and extracted even after C_w has been subjected to transformations [8].

In digital watermarking the host object *C* is a carrier signal of information, and the watermark *w* is a digital marker. The watermarking process is achieved through the introduction of errors not detectable by human perception [11]. Similar to traditional watermarking, digital watermarks can only be perceptible under specific conditions such as after using special extracting algorithms [33]. Unlike traditional watermarking, in digital watermarking when the carrier signal is copied or transferred, the watermark is also carried with the copy.

Watermarking methods have been used in various applications including digital audio, images or videos. Typically, these are used for owner identification, content authentication and copy control [12]. Similarly, data hiding is particularly popular with biomedical data because of the need to imperceptibly carry metadata or additional sensitive data such as name, ID, or sensitive medical data [37]. In biomedical research, data hiding techniques embed data reversibly, since it is important to use it in its original state in the analyses. In our case though, we can be flexible as the host data is not sensitive. In this context, data hiding can improve management efficiency, provides an additional layer of security, and can ensure confidentiality, availability and reliability [7]. As such, data hiding allows us to embed a set of metadata or sensitive data, imperceptibly, within another data set or digital file.

Finally, we observe that cryptographic techniques offer orthogonal benefits regarding these concerns. For this reason, data hiding techniques often encrypt the data before it is hidden. However, the key advantage of data hiding is the “physical” binding between carrier signal and digital marker in a manner that is transparent to computational infrastructure, can survive data migration, and does not give rise to software compatibility issues.

Properties of Data Hiding Techniques

Multiple data hiding techniques exist, and they can be classified in terms of three key properties [18].

- Robustness: Robust techniques are those where data can be extracted successfully even after the carrier signal has undergone malicious attacks, modifications or transformations. This feature is particularly desirable in cases where the host is prone to modifications, either intentional, or unintentional. An example would be lossy compression. Fragile techniques are those where minor distortions affect the hidden data. This can serve useful tamper-proofing purposes (e.g., loss of hidden data can reveal and localise modification of data).
- Imperceptibility: Data hiding techniques are considered as imperceptible when data is imperceptible to human under typical use. Hidden data can only be extracted algorithmically by an authorized user. Good imperceptibility also suggests high fidelity between the original work and the one containing data.
- Capacity: Refers to the size of the payload that can be encoded within a unit of a host object.

Traditionally in data hiding literature, there is a tradeoff between these three properties. Depending on the application domain, the priority of these properties varies. An additional constraint in the case of smartphones and mobile sensors is energy consumption and computational complexity. Previous work has highlighted that data hiding is more efficient than cryptography in terms of complexity and energy usage, and therefore more appropriate for resource-constrained hardware [18].

Data Hiding and Smartphone Sensor Data

Smartphone sensing techniques introduce a variety of applications and opportunities, such as activity recognition, health monitoring and intelligent transportation. However, smartphone sensor data may contain sensitive information, including GPS location, medical states (e.g., heart rate and travelled steps) and user profiles (e.g., identity, age, gender and calendar reminders). This challenges researchers in the design of smartphone sensing systems [20].

Although researchers can alleviate these problems using cryptography and privacy-preserving data mining techniques, existing approaches are still insufficient to keep information imperceptible or to prove the authenticity of sensor data [19, 22]. For this reason, data hiding techniques can benefit users, for example by hiding sensitive data within non-sensitive data. Furthermore, data hiding techniques can be used to prove ownership of smartphone sensor data without compromising anonymity. By embedding identification information into sensor data, sensing systems that collect data from multiple users (e.g. in crowdsensing) can easily verify the source of data and filter untrusted sources without establishing secure access APIs or explicit authentication mechanisms.

Many projects have considered data hiding techniques, especially watermark-based methods, in smartphone-driven scenarios. Miao et al. [27] developed an Android application that uses digital watermarks to protect the ownership and integrity of digital photographs. They showed that the proposed approach can resist some common attacks, such as contrast change and compression. Zhou et al. [40] developed a system named AppInk that generates watermarked apps from the source code of original apps, to detect unauthorised apps which are repackaged by attackers. Suzuki et al. [35] developed a video annotation system which embeds real-time high-frequency audio watermarks into video data of a smartphone camera. Because high-frequency audio is inaudible to humans, the audio quality of watermarked video data is not compromised. Hence, users can add annotations into video in the form of audio watermarks.

Furthermore, data hiding has been used as a barcode-like mechanism. For example, previous work embeds hyperlinks within posters or videos [9], such that a mobile device can decode this information but it is not perceptible to humans. Similarly, researchers have shown how to embed information within an audio channel transmitted over loudspeakers [26] or the phone [30], a technique that can be used for ad-hoc secure pairing, verification, and synchronisation.

In the context of sensor networks that may have to operate in untrusted environments, data hiding can meet the requirements of data integrity and authentication in communication. For example, Wang et al. [38] proposed an adaptive watermarking approach to achieve secure image transmission with low distortion and energy cost. Similarly, Zhang et al. [39] presented an end-to-end authentication scheme that employs watermarking for secure data aggregation. In these cases, watermarks are embedded by each sensor node, and the server can verify the sources of the incoming data despite an untrusted communication network.

Our work builds on previous research in many ways. The recent proliferation of scientific and commercial platforms for sensor data has given rise to the need to consider "sensing" as the application itself. Therefore, we aim to provide users a transparent way to embed one set of smartphone sensor data within another. This will allow users to adopt services on a variety of platforms without necessarily trusting them with their sensitive data. Much like sensor nodes operating in an untrusted network [38, 39], we can enable users' personal devices to share sensor data with each other via untrusted platforms. Additionally, our approach establishes a physical binding between a sensor stream and annotation data, either to prove ownership of the sensor data (as demonstrated with photos [27]), to provide additional context (as has been shown for videos [35]), or for ad-hoc communication purposes (as has been used in ad-hoc pairing [26, 30]).

STUDY

Our objective is to investigate the feasibility of hiding one sensor stream within another on a smartphone. Due to the plethora of sensors, it is important to identify their main types for our purposes. Modern smartphones can provide sensor data across the following broad categories [21]:

- hardware sensors that include motion sensors (e.g., accelerometer and gyroscope), position sensors (e.g., GPS and magnetometer), environmental sensors (e.g., light sensor and barometer), and multimedia hardware (e.g., microphone and dual-cameras).
- software sensors that include operating system data (e.g. CPU load, network connections, app usage) and application data (e.g. calendar data, browsing history, music listening data).
- human input, which captures phenomena that are imperceptible for hardware or software sensor, mainly using smartphone-based surveys and the Experience Sampling Method [23].

Given the diversity of data sources, there are two important characteristics that we consider for data hiding purposes:

- *frequency*: some sensor data may be collected at high frequency, such as accelerometer and magnetometer data. Other sensors may provide data with much lower frequency, such as heart rate sensors, GPS, or human input text. Furthermore, some data may be constant, such as user identifier, names and date of birth.
- *privacy sensitivity*: some data may be highly sensitive if exposed, such as date of birth, user identifier, or heart rate data. Other data may be less sensitive, for example accelerometer and gyroscope values.

In the diagram below we map out many of the possible smartphone sensors in terms of their frequency and sensitivity. Data hiding is ideal for hiding low-frequency sensitive data into high-frequency non-sensitive data. It is challenging to objectively map the privacy concerns that may be associated with any particular sensor, since they can vary across users and strongly depend on what other data is available. For instance, gyroscope is typically used together with accelerometer to detect activities such as walking, standing, sitting and lying can be recognised with high accuracy 96% [3], while on the other hand complex activities (e.g., cooking, cleaning and sweeping) are still considered challenging to recognise [13]. Therefore, we rely on subjective assessment, heuristics, and our review of literature [19, 22, 28] to rate the privacy concerns for each sensor. We summarise the different types of sensor data in terms of frequency and sensitivity in Figure 1.

DATA HIDING METHODOLOGY

In data hiding techniques the payload can be hidden either in the time or frequency domain of the carrier. We adopt a time domain technique, and specifically a substitutive insertion method: parts of the carrier signal are replaced by the payload signal. Specifically, we apply the Least Significant Bit (LSB) substitution scheme, replacing the carrier signal's least significant bits with bits of the payload. This approach enables embedding data of high rate and size, while causing relatively insignificant modifications to the original values [15, 29].

More importantly, our method is appropriate for real-time data hiding on limited-resource platforms such as smartphones, due to its low time complexity, and can therefore guarantee that the embedded data is synchronised with the carrier data. For example, let us assume that we need to embed a heart rate value into a stream of accelerometer values. We first obtain the binary representation of the heart rate value, which for instance can be a 32-bit integer. The next step is to embed each bit in the oncoming accelerometer stream, by replacing the LSBs of consecutive accelerometer values that have also been converted to binary form (see Figure 2). One bit is used for a flag to denote the existence of embedded data, and additional bits are used for the payload. Depending on how much we can afford to distort the carrier values, we can opt to replace two or more LSBs.

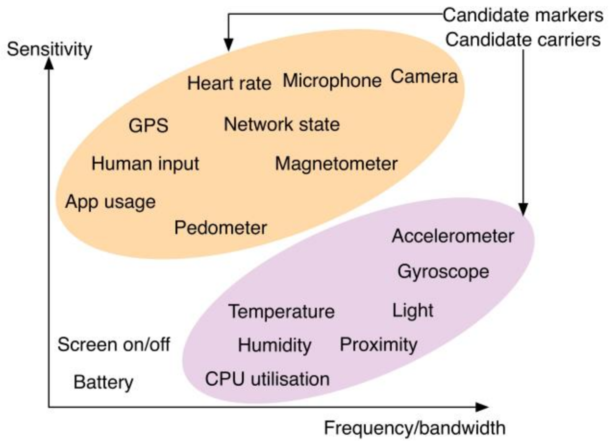

Figure 1. Sensitivity and frequency of different sensor data.

Authenticated encryption prior to embedding enables us to detect errors or tampering of the hidden data. Therefore, we apply the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) [14, 32] to encrypt and authenticate the embedded data. GCM is a mode of operation that supports simultaneous encryption and authentication of a data stream in an efficient, parallelisable manner. To ensure the integrity of the data, an authentication tag is generated at the end of the input stream. The GCM mode is nonce-based which means that a unique public identifier called nonce or initialisation vector needs to be used for each set of data. Both the nonce and the secret encryption key are needed for decryption and to check the integrity of the data. Any tampering of the encrypted hidden data will be detected during the decrypting phase.

For our evaluation we implemented our method on Android smartphones. We developed a pair of applications that run simultaneously and communicate via Intent messages within Android. The sensing application collects the sensitive data to be hidden and passes it to the data hiding application. The latter encrypts the received data, hides it into the carrier signal, and potentially stores it or transmits it to a third party. The two-application architecture was chosen to allow flexibility in practical scenarios, and to conduct a realistic assessment of performance. To extract the hidden data, the receiving party needs access to the carrier signal (sorted by timestamp), knowledge of the data types, how many LSBs are used, the pre-shared nonce, the authentication tag and encryption key.

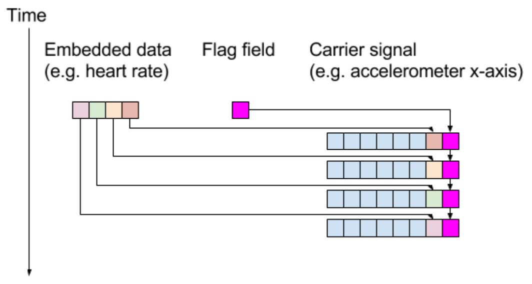

Figure 2. The Least Significant Bit (LSB) method replaces the least significant bits of the carrier signal with data to be embedded.

Distortion of the Carrier Signal

The distortion produced by hiding data into the carrier signal depends on the data type of the carrier signal and the number of LSBs. For example, given 32-bit integer types as the carrier signal, using n ($0<n<32$) LSBs for data hiding will produce an error from $-(2^n-1)$ to 2^n-1. The cases where floating-point numbers serve as carrier signals are more complex. Suppose the carrier signal is a 32-bit single precision floating number f [17] defined as

$$f = (-1)^s \times c \times 2^q, \qquad (1)$$

where s stands for the sign bit; c for the significand; and q for the exponent.

If we use n ($0<n<23$) LSBs (which determine the significand c) for data hiding, we must know the exponent value q (which depends on the magnitude of the floating number) to quantify the maximal amount of error $|E_{max}|$. It can be calculated as

$$|E_{max}| = \sum_{i=1}^{n} 2^{i-24+q} \qquad (2)$$

This formula reveals that a small number of LSBs will produce negligible errors with floating-point numbers. However, the absolute amount of error can be very high for large q. Although q can be up to 127 to represent a valid real number [17], smartphone sensors generally output much smaller readings in practice, such as accelerometer [25]. Thus, floating-point numbers are ideal carrier signals, since data hiding can have a negligible distortion on them.

EXPERIMENTAL DESIGN

We evaluated the performance of our approach by running experiments with off-the-shelf smartphones. Considering the availability of AES/GCM encryption, we selected 6 smartphones with Android OS version 5.0 or above:

Samsung Galaxy S6 edge (5.1.1); two samples of Motorola Moto G X1032 (5.1); LG Nexus 5 (5.1.1); Motorola Moto G2 (5.0.2); Yota YotaPhone 2 (5.0).

Experiment	Device	Marker	Carrier	LSBs
E1	All	Magnetometer (3 x 32-bit float) (normal, UI, game, fastest)	Accelerometer (3 x 32-bit float) (normal, UI, game, fastest)	2
E2	S6	Magnetometer (3 x 32-bit float) (normal, UI, game, fastest)	Accelerometer (3 x 32-bit float) (normal, UI, game, fastest)	3
E3	S6	Heart rate sensor (32-bit int) (normal, UI, game, fastest)	Accelerometer (1 x 32-bit float) (normal, UI, game, fastest)	2
E4	S6	GPS (3 x 64-bit double) (0.1, 0.2, 1, 10 Hz)	Accelerometer (3 x 32-bit float) (normal, UI, game, fastest)	2
E5	S6	Human input (8-bit char) (1, 10, 100, 200 Hz)	Accelerometer (1 x 32-bit float) (normal, UI, game, fastest)	2
E6	S6	Device ID (16 x 8-bit char)	Accelerometer (1 x 32-bit float) (normal, UI, game, fastest)	2

Table 1. Experimental parameters. The frequencies "normal, UI, game, fastest" are Android standards, and may perform differently across different handsets. The Least Significant Bit includes a 1-bit flag field.

We installed our pair of Android applications on each handset. We first launched the data hiding application to initialise the encryption and carrier signal. Then, we launched the sensing application to collect the data to be hidden, and pass it along for hiding. Because it is impractical to exhaust all the combinations of multiple variables, we designed 6 experiments to examine a range of conditions as summarised in Table 1.

We use the accelerometer as the carrier signal in all the experiments, and we consider 4 different sampling rates for it, as defined by Android. In E1 and E2 we hid magnetometer data at 4 different sampling rates. In E3 we hid heart rate data at 4 different sensing rates. In E4 we hid simulated streaming GPS data (64-bit double type in 3 dimensions: latitude, longitude and altitude) generated at 4 different frequencies. In E5, we hid simulated human input text at varying frequencies. In E6 we hid a device ID (an Android device ID has 16 characters).

For experiments E1, E2 and E4, we used 3 axes of the accelerometer as the carrier, since in those experiments we effectively had 3 streams of data to hide. In the other experiments only the *x* axis was used as a carrier. In experiments 5 and 6, the data hiding application used the 8-bit ASCII format. The experiments had a combination of sampling frequencies of the data to hide, and 4 frequencies of the carrier signal. Orthogonally, E1 had 6 different phones. Each condition ran for a period of 5 minutes, during which the performance of the system was monitored.

EXPERIMENTAL RESULTS

Figure 3 summarises the results in E1, where magnetometer data was encrypted and embedded into the accelerometer data using 4 different sampling rates on 6 handsets. The dark red shades represent the magnetometer records that were hidden, and the light blue shades above red shades represent the number of magnetometer records that could not be processed due to the too high bit rate of the payload, and therefore had to be dropped.

E1 primarily acted as a "stress test" to highlight performance differences across handsets. As such, we induced record dropping due to the relatively high volume of magnetometer data that we attempted to hide, as well as variances in the capabilities of the handsets. The results show that the sampling rate at "normal" and "UI" was consistent across handsets. However, the handsets performed substantially differently at the "Game" and "Fastest" sampling rates, for instance with the S6 outperforming G1 handsets by a factor of 2.

We further investigate the variation in the carrier frequency across handsets in E1. Figure 4 shows the average accelerometer delay, which denotes the time gap between two adjacent samples. Sensing delay is an indirect measure of the ability to execute data hiding.

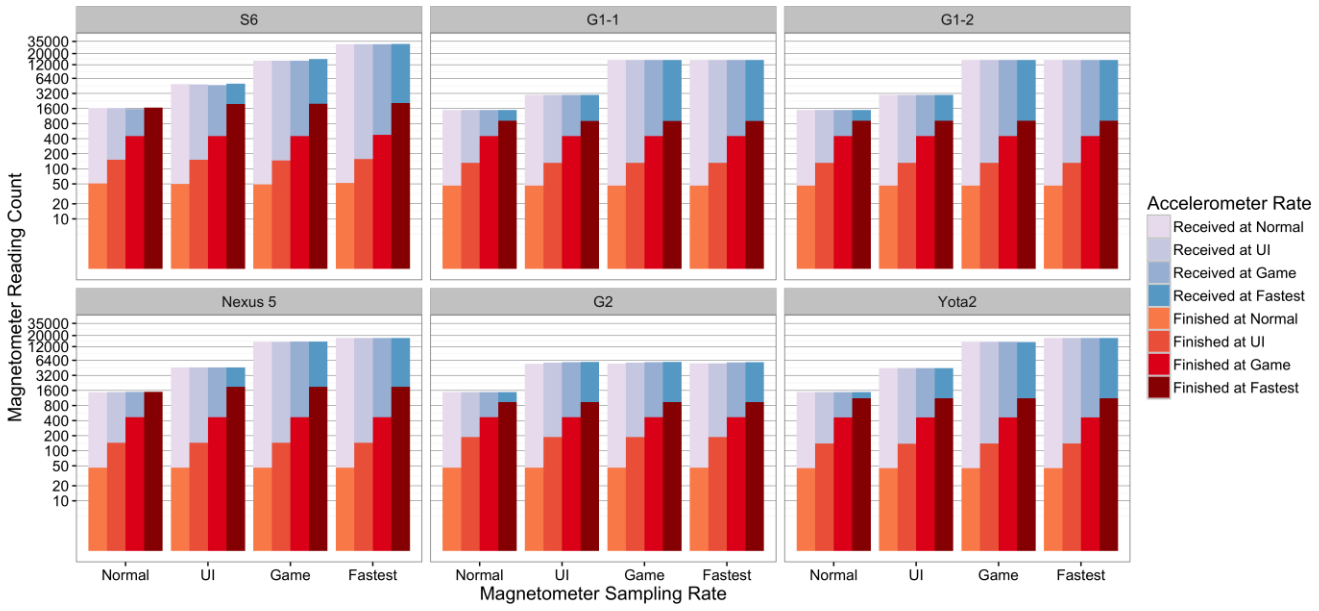

Figure 3. Results of E1. The number of magnetometer readings which are successfully hidden is shown in red, and those dropped is shown in blue. The *y* axis is on a base-2 logarithmic scale.

Based on the sensing delay, we can estimate the capacity of accelerometer as a carrier signal on each device. Figure 5 presents the capacity of one axis of the accelerometer for 2 LSBs (1 bit flag & 1 bit payload). We observe that at the fastest sampling rate, all handsets can provide a capacity of more than 10B/s, with the highest being 26.8B/s for the S6. If 3 axes are used, then the capacity increases by a factor of 3. In addition, the capacity increases proportionally for each additional payload bit we use. Therefore, we expect the S6 with 3 axes and 3 LSBs (1-bit flag & 2-bit payload) to provide 26.8 × 3 × 2 = 160B/s capacity.

Once our data hiding application received a new magnetometer data reading, it executed AES/GCM encryption and hid the ciphertext bits into the incoming accelerometer records. When the ciphertext bits are more than the payload of one accelerometer record, phones have to embed the rest cipher bits into more incoming accelerometer records.

In Figure 6 we show the computational overhead that encryption induced in E1. Results show that, on average, all the handsets were able to finish the task of encryption plus data hiding for one sample within 0.6 ~ 6.2ms for any condition (max: 302.13ms due to CPU scheduling). Of this time, less than 0.2 ~ 0.8ms on average (max: 428.95) was spent on just data hiding.

Figure 7 shows the performance of the S6 handset across all experiments, and therefore for multiple data types. As expected, using an additional LSB in E2 doubled its capacity. In E3 we noted that the heart rate sensor hardware did not alter its sampling rate, contrary to Android API specifications. In E4, as expected, the results show that the number of GPS records we could hide was approximately half of the magnetometer in E1. In E5 the hidden data was simulated human entry text, which was on average 3 times faster than E1. In E6 we hid a Device Identification code, and therefore the sampling rate did not vary.

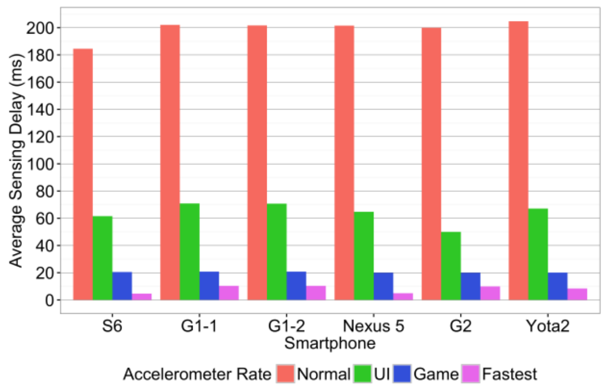

Figure 4. Average sensing delay of accelerometer for different handsets in E1.

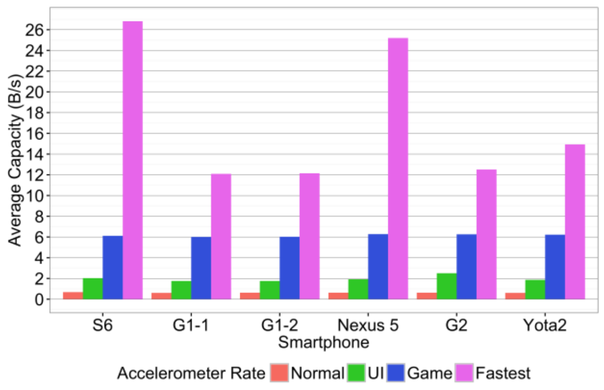

Figure 5. Average capacity using one-axis accelerometer carrier on 6 phones, using 2 LSBs (1 bit flag & 1 bit payload).

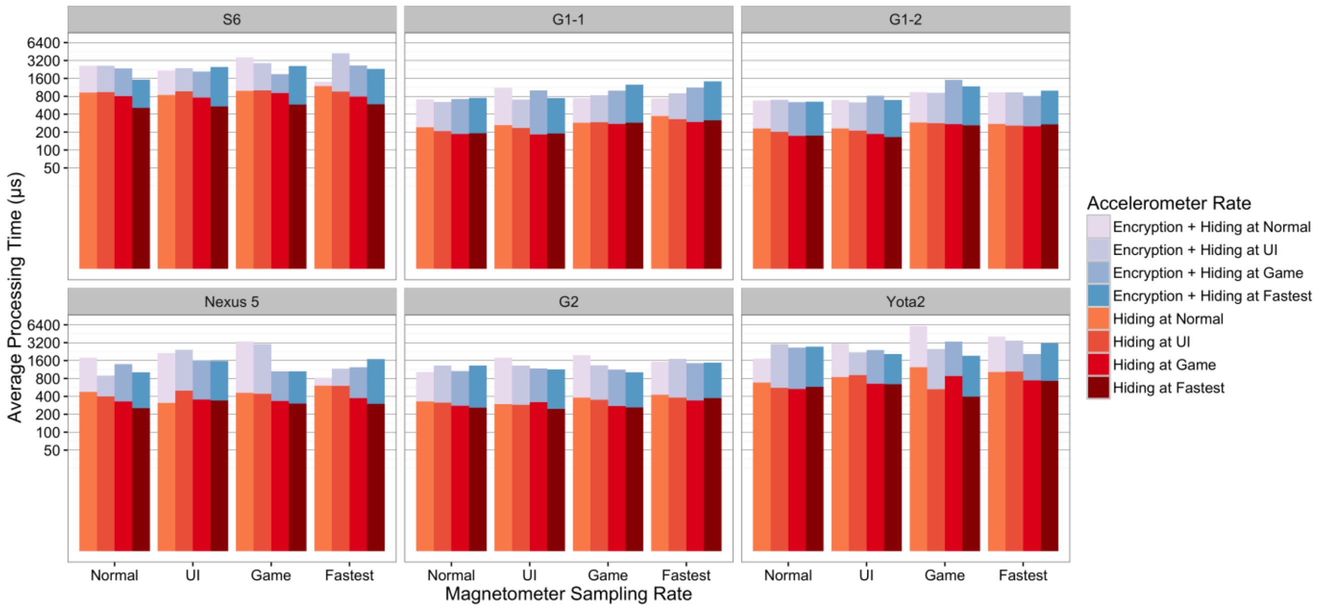

Figure 6. Average processing time for encryption & hiding (blue), or just hiding (red) in E1. This is the time needed for one magnetometer record. The *y* axis is in base-2 logarithmic scale.

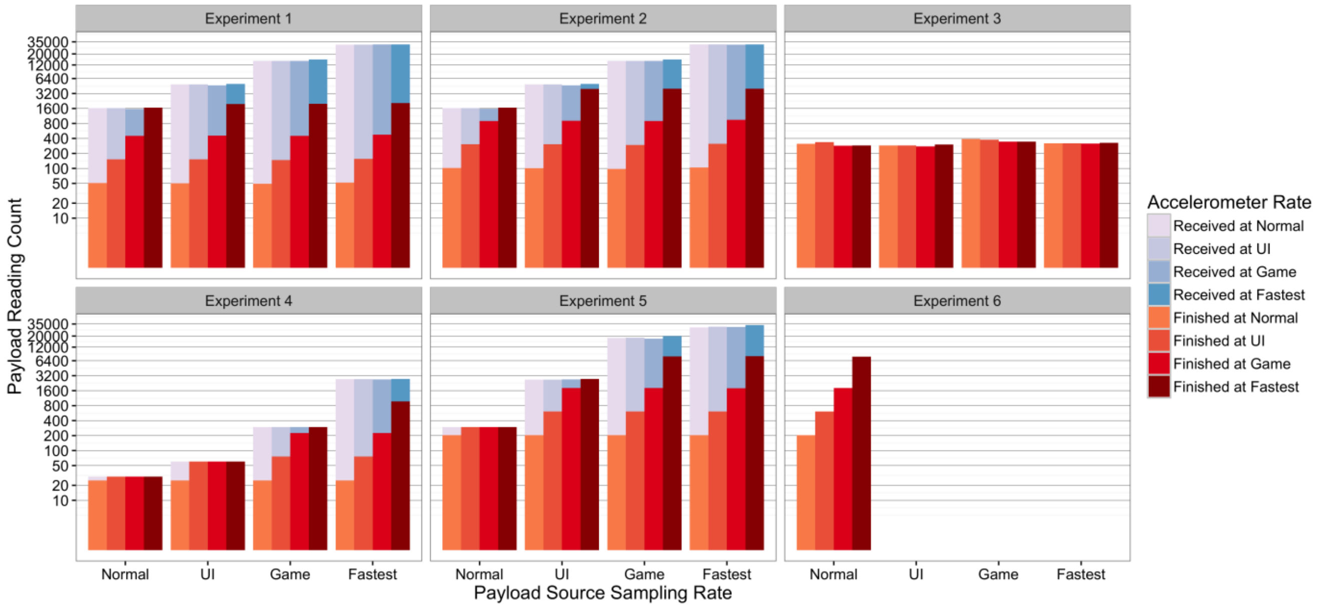

Figure 7. Performance of the S6 handset across all experiments. Number of readings which are successfully hidden is shown in red, and those dropped is shown in blue. The *y* axis is on a base-2 logarithmic scale.

Also, in E3 we observed that the accelerometer sampling rate was unexpectedly doubled compared to all other experiments (for UI speed: 30ms in E3 vs 60ms in other experiments). This is a phenomenon that we were able to reliably reproduce. Given the lack of official documentation we believe that on this particular handset, using the heart rate sensor triggers additional mechanisms that increase the sampling rate of the accelerometer. Figure 8 shows the average processing time of encryption and data hiding on S6 across all experiments. Considering encryption plus data hiding (blue), the average processing time follows the complexity of payload types and the number of LSBs: E1 (32-bit float on 3 axes, 2 LSBs): 2.49ms; E2 (32-bit float on 3 axes, 3 LSBs): 2.56ms; E3 (32-bit int, 2 LSBs): 1.14ms; E4 (64-bit double on 3 axes, 2 LSBs): 2.92ms; E5 (8-bit ASCII, 2 LSBs): 0.74ms; E6 (8-bit ASCII, 2 LSBs): 0.52ms. Similar to the worst case (among all handsets) in E1, the worst cases in E2-E6 ranged from 40.76ms to 380.67ms. When considering only data hiding (red), the S6 handset was able to finish within 0.9ms on average across all 6 experiments. The worst cases in E2-E6 ranged from 57.29ms to 593.36ms.

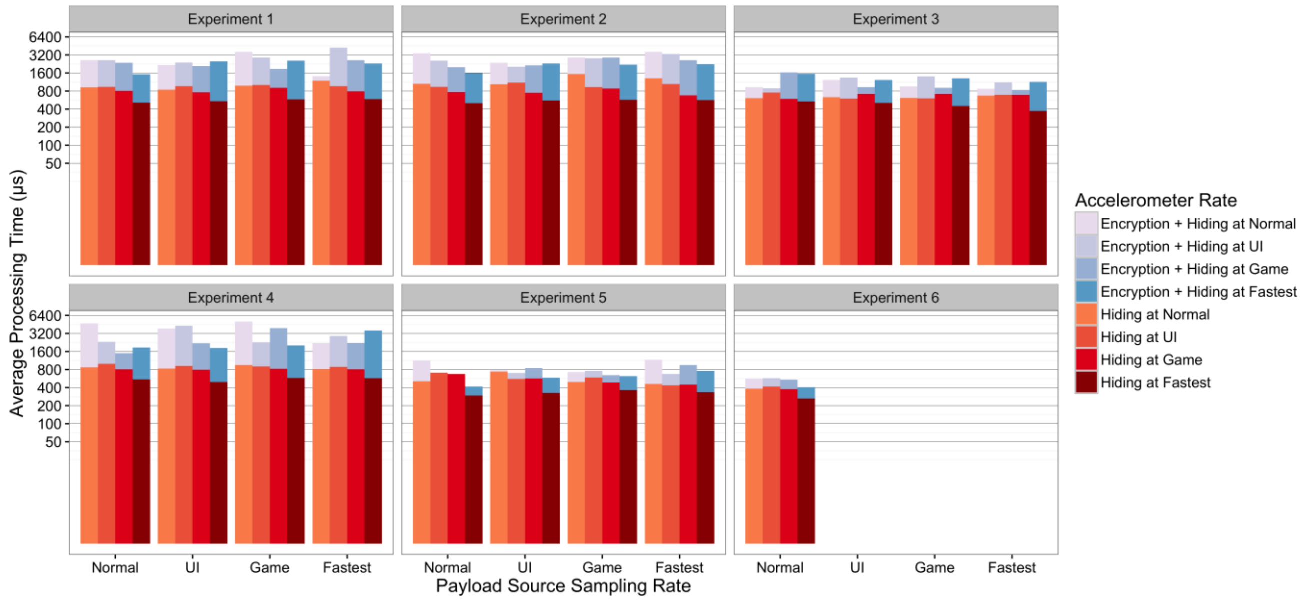

Figure 8. Average processing time (S6 handset across all experiments) for encryption & hiding (blue), or just hiding (red) in E1. The *y* axis is in base-2 logarithmic scale.

CPU Utilisation

We also considered the impact of our data hiding method on CPU utilisation. We logged CPU utilization data for the S6 handset in E1 using the Android Device Monitor. We consider encryption and data hiding as two independent processes, since they are separate functions in our source code and can be monitored independently in CPU utilisation analysis.

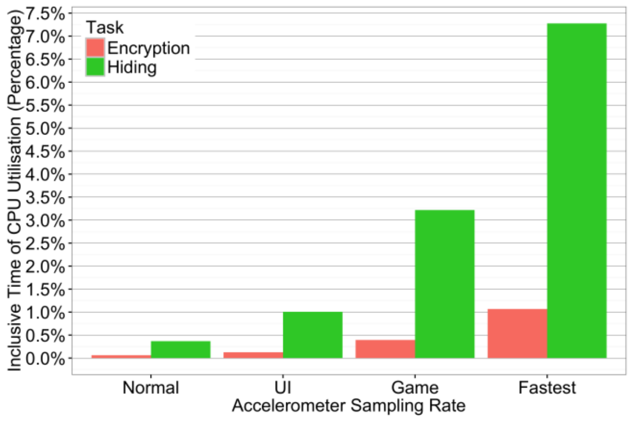

Figure 9. Inclusive time of CPU utilization (%) on S6 handset in E1. Separate utilisation is shown for encryption (red) and hiding (green).

Figure 9 presents the results of encryption vs. hiding at different accelerometer sampling rates. Note that 100% of inclusive CPU time would indicate that the whole period when the data hiding application is running its thread uses a CPU. These results show that our software does not occupy the CPU all the time, meaning that the CPU may set the application thread into the wait state to save energy. We also observe that the CPU was occupied more often with data hiding rather than encryption, even though one call to the data hiding function takes much less time than one call to the encryption function (Figure 6). This disparity is due to the fact that each record of data to be hidden is encrypted once, but requires many calls to the data hiding function, since only 1 or 2 bits can be hidden at a time. For instance, a 32-bit payload is encrypted once but requires 32 calls to the data hiding function when using 2 LSBs (1-bit payload & 1-bit flag).

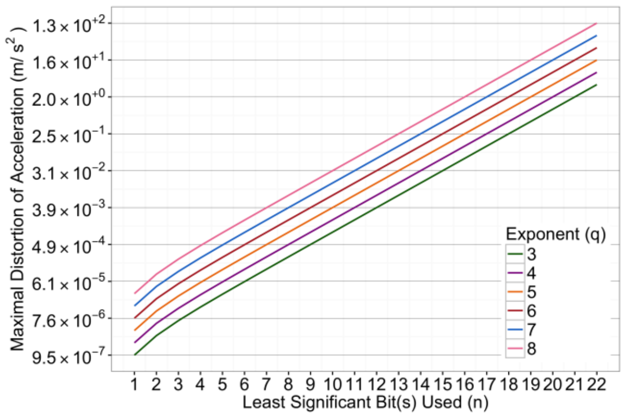

Figure 10. Maximal distortion of acceleration for different LSBs and floating-point exponents. The *y* axis is in base-2 logarithmic scale.

Distortion

According to standards [17], to represent a floating-point number v, the exponent value q in equation (1) must be maximised with the constraint that 2^q is not greater than $|v|$. This means that the amount of error increases as the maximal possible value of $|v|$ is greater. According to the measurement range of common smartphone accelerometer

[25], q is at most 8. Figure 10 depicts the maximal distortion that we theoretically induce for different LSBs and exponents. The number of LSBs (i.e., n) depends on the experiment settings.

We contrast the theoretical prediction with empirical data of the distortion in the carrier signal in E1 and E2 on the S6 handset. The handset was placed on the flat table so that the z axis of accelerometer showed the gravity which was about 10m/s^2, as meaning that a floating point number needs q=3 to represent this value.

In E1 (where 2 LSBs are used) we recorded 2.861×10^{-6}m/s^2 as the maximal absolute value of error in the carrier signal. This result exactly matches our theoretical estimation which is given by equation (2) when n=2 and q=3. Similarly, in E2 (when the number of LSBs was 3), we logged the maximal absolute error 6.676×10^{-6}m/s^2. This also exactly matches our theoretical estimation where equation (2) has n=3 and q=3.

DISCUSSION

Performance

Our results show that smartphone sensor streams can provide sufficiently high capacity for common data hiding scenarios, especially when used with high frequency carrier signals. Depending on the security concerns of smartphone sensing systems, a variety of smartphone data types, such as floating numbers (e.g., magnetometer and GPS), integers (e.g., heart rate) and characters (e.g., human input text and device ID code) can be a suitable payload hosted in the carrier signal.

Indicatively, we measured on the S6 handset a maximum capacity of 26.8B/s with a 1-axis accelerometer carrier signal. Give the expected distortion shown in Figure 10, the capacity for 7 LSBs on a 3-axis carrier signal is 526B/s, with expected distortion between 10^{-5}m/s^2 and 4×10^{-3}m/s^2. To extract this hidden data, a recipient requires knowledge of:

- the data type of the host signal;
- the data type of hidden data;
- the number of LSBs used in the host signal;
- the host signal sorted by timestamp;
- the information for decryption (in the case of AES/GCM, they are the nonce, the authentication tag and the decryption key).

Beyond the confidentiality and integrity provided by AES/GCM encryption, hiding data into another sensor stream obscures the existence of sensitive and private data secret by making it imperceptible. Thus, as Lane et al. [22] have called for, using our approach the type and value of sensitive data streams are not accessible or noticeable to a third party, taking one step closer towards the preservation of privacy. For example, an attacker may find it useful to know that a user is uploading location data, even if they cannot see the actual data. Our method alleviates this concern by obscuring the existence of such sensitive data. In practice, this means that sensitive data is not stored in a separate database field (thus making it perceivable to third parties). In addition, if the payload is an encrypted identity code such as a device ID, it can be used to verify the authenticity of the carrier signal source.

Implementation Issues

Our approach has a manageable computational cost (Figure 9), making it practical for smartphones [18] and allowing power-efficiency OS techniques to reduce its energy footprint, for example setting threads to sleep mode. In addition, the theoretical predictions regarding the distortion caused by our technique (Figure 10) have been empirically confirmed, thus guaranteeing the level of fidelity between the original carrier signal and the signal containing hidden information.

This is important for a range of applications. Certain applications that use accelerometer data require high precision, such as gesture recognition [34], while other applications like scrolling via tilting [5] require crude precision since smartphone accelerometers and gyroscopes produce measurement errors anyway [10]. Our method is flexible enough to account for varying needs regarding the fidelity of processed data, by trading off fidelity and capacity.

Our approach can be adopted by existing sensing systems that already support smartphone sensor data. In particular, we envision that a user with multiple devices (e.g. phone, tablet, smartwatch) would be able to transparently share sensitive between those devices via existing platforms. As long as each device has access to the sensor data, it is possible to extract and decrypt hidden data on the client, without allowing the platform to gain access, or even know that the hidden data exists. This is possible without modifying the platform itself, and not requiring additional “encrypted” fields to be supported.

Medical Sensor Data

Due to its technical characteristics, our proposed data hiding technique can help to address the legislation that many countries have to protect sensitive data, especially medical sensor data [4, 36]. In general, the development of medical information systems has been a challenging and costly affair for many countries [16] due to the complex privacy requirements.

For instance, it is challenging to enable users to retain control of their own data after it has been entered in the system, and giving them access to this data is often a thorny issue [2]. Our method enables users to retain control of their sensitive data even after it has been uploaded on a healthcare information system. For example, during consultation a user could decrypt sensitive information using the secrets stored in their personal device, and show it to the doctor.

Crowdsensing

Additionally, our technique enables the verification of the authenticity or owner of smartphone sensor data. This is particularly relevant to mobile crowdsensing scenarios, either user-driven [6] or agent-driven [24], with diverse application including environmental monitoring and intelligent transportation. In such settings, malicious users or faulty systems can upload tampered or faked data to damage the systems or to defraud benefits if the systems offer rewards for uploading certain data. In this scenario, our technique can offer crucial digital evidence for forensics [28] to ensure the authenticity of smartphone sensor data. For example, this can be achieved by smartphone applications embedding an encrypted unique identification number into every uploaded sensor data stream. When the streams are received, their authenticity can be established by inspecting the identification number. If the received data stream does not contain the ID assigned to a particular client, the systems can consider the data invalid and ignore it.

Along the same lines, initiatives such as Crawdad and Crowdsignals are building up large archives of sensor data. Using our technique, it is possible for users to "physically" embed in this data a unique signature that serves as proof of ownership of the data, and can be used to confirm that no tampering has taken place. The "physical" binding means that even if this sensor data is shared between scientists via email, database services, physical media, and across a variety of file formats, the hidden data persists. This property also ensures that it is "future proof", in the sense that if in the future new ways of sharing data is established, the hidden data will remain available as proof of who owns or generated this sensor stream.

LIMITATIONS AND FUTURE WORK

We have only verified our approach on 6 phones with Android OS 5.0 or above, and we are aware that approximately 65% of Android smartphones run a lower version that 5.0 at time of writing. We expect our method's performance to vary across different handsets, but only in terms of capacity and CPU load. The other features of our method should remain invariant.

Clearly, our method has not been tested on other operating systems, such as Symbian, iOS and Windows, and this would be a crucial next step for our work. A key challenge may be that the implementation of AES/GCM may be unavailable on other handsets, meaning that an ad-hoc algorithm may be needed. Although developers can employ other encryption algorithms, this may downgrade the performance and security level. Another technical issue is that the computational efficiency in other handsets environments can be significantly lower than Android 5.0, meaning that they cannot use high-frequency sensor data as the carrier signal.

In addition, we have not tested our method with a broad range of external sensors or devices (such as smartwatches). Platforms with higher constraints (such as smartwatches) may find it challenging to attain high capacity data hiding.

During the selection of carrier signals, objectively quantifying the sensitivity of each sensor can provide greater robustness to the data hiding mechanism. This requires a substantial body of future work. The positions of sensors (i.e., their sensitivity) in Figure 1 may depend on various factors, such as social context, network environments and capabilities of attackers.

Our future work will also include a mechanism to balance the tradeoff between capacity and fidelity in carrier signals. A large number of LSBs leads to high capacity and low fidelity in the carrier signals. Therefore, this mechanism should adaptively identify a suitable upper bound of LSBs in different types of data hiding scenarios.

CONCLUSION

We propose a data hiding method to embed sensitive information into smartphone sensor data streams. Our method combines encryption with data hiding, and can be adopted by smartphone sensing systems to secure sensitive data or to prove the authenticity of data. Due to the imperceptibility of data hiding techniques, an unauthenticated party does not notice the type and value of hidden sensitive data stream by interception, thus alleviating some of the privacy problems of smartphone sensing systems mentioned in literature [22].

We evaluated with a variety of handsets, data types, and settings. Our experimental results show that it is feasible to encrypt and embed common types of smartphone data (e.g., magnetometer readings, heart rate, GPS location, human input text and device identification code) into high-frequency sensor streams, such as accelerometer, in real time. Moreover, we show that AES/GCM encryption and data hiding operations have manageable impact on the CPU utilisation of the sensing application thread, meaning that our approach will not be bottlenecked by resource-constrained environments of smartphones. We demonstrate that our approach is able to maintain high fidelity after data hiding, and can provide strong guarantees regarding fidelity by adjusting the number of LSBs used for hiding. Our findings make this data hiding method attractive for smartphones sensing systems that collect sensitive data or require high data authenticity, such as medical systems and digital forensics applications.

ACKNOWLEDGMENTS

This work is partially funded by Infotech Oulu, TEKES-funded VitalSens project, the Academy of Finland (Grants 276786-AWARE, 285062-iCYCLE, 286386-CPDSS, 285459-iSCIENCE), and the European Commission (Grants PCIG11-GA-2012-322138, 645706-GRAGE, and 6AIKA-A71143-AKAI).

REFERENCES

1. Gregory D. Abowd. 2012. What next, ubicomp? Celebrating an intellectual disappearing act. In

Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12, 31-40. http://dx.doi.org/10.1145/2370216.2370222.

2. Corey M. Angst and Ritu Agarwal. 2009. Adoption of Electronic Health Records in the Presence of Privacy Concerns: The Elaboration Likelihood Model and Individual Persuasion. *MIS Q.* 33, 2: 339-370.
3. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge J. L. Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*.
4. David Blumenthal and Marilyn Tavenner. 2010. The meaningful use regulation for electronic health records. *New England Journal of Medicine* 363, 6: 501-504.
5. Sebastian Boring, Marko Jurmu and Andreas Butz. 2009. Scroll, Tilt or Move It: Using Mobile Phones to Continuously Control Pointers on Large Public Displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, ACM, 161-168. http://dx.doi.org/10.1145/1738826.1738853.
6. Yohan Chon, Nicholas D. Lane, Yunjong Kim, Feng Zhao and Hojung Cha. 2013. Understanding the Coverage and Scalability of Place-centric Crowdsensing. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 3-12. http://dx.doi.org/10.1145/2493432.2493498.
7. G Coatrieux, L Lecornu, B Sankur and Ch Roux. 2006. A Review of Image Watermarking Applications in Healthcare. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, IEEE, 4691-4694. http://dx.doi.org/10.1109/IEMBS.2006.259305.
8. Christian Collberg and Jasvir Nagra. 2009. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection.* Addison-Wesley Professional.
9. John P. Collomosse and Tim Kindberg. 2008. Screen Codes: Visual Hyperlinks for Displays. In *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications*, ACM, 86-90. http://dx.doi.org/10.1145/1411759.1411782.
10. Ionut Constandache, Xuan Bao, Martin Azizyan and Romit R. R. Choudhury. 2010. Did You See Bob?: Human Localization Using Mobile Phones. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ACM, 149-160. http://dx.doi.org/10.1145/1859995.1860013.
11. Ingemar J. Cox, Joe Kilian, Tom Leighton and Talal Shamoon. 1996. A secure, robust watermark for multimedia. In *Information Hiding* (eds.). Springer Berlin Heidelberg, 185-206.
12. Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich and Ton Kalker. 2008. *Digital Watermarking and Steganography.* Morgan Kaufmann Publishers Inc..
13. Stefan Dernbach, Barnan Das, Narayanan C. Krishnan, Brian L. Thomas and Diane J. Cook. 2012. Simple and Complex Activity Recognition through Smart Phones. In *International Conference on Intelligent Environments*, IEEE, 214-221. http://dx.doi.org/10.1109/IE.2012.39.
14. Morris J. Dworkin. 2007. *SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.*
15. Frank Hartung and Martin Kutter. 1999. Multimedia watermarking techniques. *Proceedings of the IEEE* 87, 7: 1079-1107. http://dx.doi.org/10.1109/5.771066.
16. Richard Heeks. 2006. Health information systems: Failure, success and improvisation. *International Journal of Medical Informatics* 75, 2: 125-137. http://dx.doi.org/10.1016/j.ijmedinf.2005.07.024.
17. 2008. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*: 1-70. http://dx.doi.org/10.1109/IEEESTD.2008.4610935.
18. Hussam Juma, Ibrahim Kamel and Lami Kaya. 2008. Watermarking sensor data for protecting the integrity. In *International Conference on Innovations in Information Technology*, IEEE, 598-602. http://dx.doi.org/10.1109/INNOVATIONS.2008.4781662.
19. Apu Kapadia, David Kotz and Nikos Triandopoulos. 2009. Opportunistic sensing: Security challenges for the new paradigm. In *Communication Systems and Networks and Workshops*, IEEE, 1-10. http://dx.doi.org/10.1109/COMSNETS.2009.4808850.
20. Predrag Klasnja, Sunny Consolvo, Tanzeem Choudhury, Richard Beckwith and Jeffrey Hightower. 2009. Exploring Privacy Concerns About Personal Sensing. In *Proceedings of the 7th International Conference on Pervasive Computing*, Springer-Verlag, 176-183. http://dx.doi.org/10.1007/978-3-642-01516-8_13.
21. Vassilis Kostakos and Denzil Ferreira. 2015. The Rise Of Ubiquitous Instrumentation. *Frontiers in ICT* 2, 3: 1-2. http://dx.doi.org/10.3389/fict.2015.00003.
22. Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury and Andrew T. Campbell. 2010. A Survey of Mobile Phone Sensing. *Communications Magazine, IEEE* 48, 9: 140-150. http://dx.doi.org/10.1109/MCOM.2010.5560598.

23. Reed Larson and Mihaly Csikszentmihalyi. 2014. *The Experience Sampling Method.* Springer Netherlands.

24. Teemu Leppänen, José A. Lacasia, Yoshito Tobe, Kaoru Sezaki and Jukka Riekki. 2015. Mobile crowdsensing with mobile agents. *Autonomous Agents and Multi-Agent Systems*: 1-35. http://dx.doi.org/10.1007/s10458-015-9311-7.

25. LIS3DH MEMS digital output motion sensor ultra low-power high performance 3-axes "nano" accelerometer. http://www.st.com/web/en/resource/technical/document/datasheet/CD00274221.pdf.

26. Cristina V. Lopes and Pedro M. Q. Aguiar. 2003. Acoustic Modems for Ubiquitous Computing. *IEEE Pervasive Computing* 2, 3: 62-71. http://dx.doi.org/10.1109/MPRV.2003.1228528.

27. Nai Miao, Yutao He and Jane Dong. 2012. hymnMark: Towards Efficient Digital Watermarking on Android Smartphones. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, 1-8.

28. Alexios Mylonas, Vasilis Meletiadis, Lilian Mitrou and Dimitris Gritzalis. 2013. Smartphone sensor data as digital evidence. *Computers & Security* 38: 51-75. http://dx.doi.org/10.1016/j.cose.2013.03.007.

29. W. Pan, G. Coatrieux, J. Montagner, N. Cuppens, F. Cuppens and C. Roux. 2009. Comparison of some reversible watermarking methods in application to medical images. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, IEEE, 2172-2175. http://dx.doi.org/10.1109/IEMBS.2009.5332425.

30. Jennifer Pearson, Simon Robinson, Matt Jones, Amit Nanavati and Nitendra Rajput. 2013. ACQR: Acoustic Quick Response Codes for Content Sharing on Low End Phones with No Internet Connectivity. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, ACM, 308-317. http://dx.doi.org/10.1145/2493190.2493195.

31. Yanzhi Ren, Yingying Chen, Mooi C. Chuah and Jie Yang. 2015. User Verification Leveraging Gait Recognition for Smartphone Enabled Mobile Healthcare Systems. *Mobile Computing, IEEE Transactions on* 14, 9: 1961-1974. http://dx.doi.org/10.1109/TMC.2014.2365185.

32. Amit Sahai. 2004. Secure Protocols for Complex Tasks in Complex Environments. In *Progress in Cryptology - INDOCRYPT 2004*, Springer Berlin Heidelberg, 14-16. http://dx.doi.org/10.1007/978-3-540-30556-9_2.

33. Frank Y. Shih. 2007. *Digital Watermarking and Steganography Fundamentals and Techniques.* CRC Press.

34. Boris Smus and Vassilis Kostakos. 2010. Running gestures: hands-free interaction during physical activity. In *International Conference on Ubiquitous Computing Adjunct*, ACM, 433-434. http://dx.doi.org/10.1145/1864431.1864473.

35. Ryohei Suzuki, Daisuke Sakamoto and Takeo Igarashi. 2015. AnnoTone: Record-time Audio Watermarking for Context-aware Video Editing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 57-66. http://dx.doi.org/10.1145/2702123.2702358.

36. Astrid M. van Ginneken. 2002. The computerized patient record: balancing effort and benefit. *International Journal of Medical Informatics* 65, 2: 97-119. http://dx.doi.org/10.1016/s1386-5056(02)00007-2.

37. R. Velumani and V. Seenivasagam. 2010. A reversible blind medical image watermarking scheme for patient identification, improved telediagnosis and tamper detection with a facial image watermark. In *IEEE International Conference on Computational Intelligence and Computing Research*, IEEE, 1-8. http://dx.doi.org/10.1109/ICCIC.2010.5705832.

38. Honggang Wang, Dongming Peng, Wei Wang, Hamid Sharif and Hsiao-Hwa Chen. 2008. Energy-Aware Adaptive Watermarking for Real-Time Image Delivery in Wireless Sensor Networks. In *International Conference on Communications*, IEEE, 1479-1483. http://dx.doi.org/10.1109/ICC.2008.286.

39. Wei Zhang, Yonghe Liu, Sajal K. Das and Pradip De. 2008. Secure data aggregation in wireless sensor networks: A watermark based authentication supportive approach. *Pervasive and Mobile Computing* 4, 5: 658-680. http://dx.doi.org/10.1016/j.pmcj.2008.05.005.

40. Wu Zhou, Xinwen Zhang and Xuxian Jiang. 2013. AppInk: Watermarking Android Apps for Repackaging Deterrence. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ACM, 1-12. http://dx.doi.org/10.1145/2484313.2484315.