iRyP: A Purely Edge-based Visual Privacy-Respecting System for Mobile Cameras

Yuanyi Sun
Penn State University
University Park, PA 16802, USA
yus160@psu.edu

Shiqing Chen
Penn State University
University Park, PA 16802, USA
u0vv0u@gmail.com

Sencun Zhu
Penn State University
University Park, PA 16802, USA
sxz160@psu.edu

Yu Chen
State University of New York
Binghampton, NY 13902, USA
ychen@binghamton.edu

ABSTRACT

With the growing popularity of mobile devices that have built-in cameras, capturing images has become a trivial job for ordinary people, who share the images with their friends or the public online. However, such digital images are often taken without the consent of some photographed persons, hence leading to privacy concerns. In this paper, we propose iRyP, a purely edge-based privacy-respecting system for mobile cameras. In order to meet the requirements of efficiency and usability, we propose to piggyback privacy policies in the advertising messages of Bluetooth Low Energy (BLE), which has been widely deployed in most mobile devices. As such, privacy policies of people in a photo view can be delivered timely and automatically. Moreover, we propose to use a perceptual hashing algorithm for fast face matching. To improve detection accuracy, we also design several new techniques for face-related image processing. We implement and evaluate a prototype system purely based on the Android platform. Our experiments show that iRyP can meet our design requirements and is practical and ready to use.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

KEYWORDS

Mobile cameras, Privacy protection, Face matching, Visual Privacy

ACM Reference Format:
Yuanyi Sun, Shiqing Chen, Sencun Zhu, and Yu Chen. 2020. *iRyP*: A Purely Edge-based Visual Privacy-Respecting System for Mobile Cameras . In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3395351.3399341

1 INTRODUCTION

The ubiquity of smartphones has made taking photos a trivial job even for people without expertise. Unlike traditional cameras, users do not need to understand the concept of focus or exposure; the only rule necessary to remember is to press a button on the phone's screen when ready. Because of the easy-to-use features, more and more people have become used to recording life with mobile (or wearable) cameras, from public events to individual activities, and then share them online. This habit is a mainstream trend in our modern society. However, the "unprocessed" pictures posted online may violate and impinge upon the privacy expectations of others, because people may be unwilling to be included in strangers' photos, which may be shared online without their consent.

As several user studies [6, 10, 13] indicated, many people do care about this privacy issue and want to customize their own context-dependable policies, and an overwhelming majority of surveyed users stated that they would choose to comply with the privacy preferences of friends and strangers, as long as doing so would not interfere with the spontaneity of photo capturing [6].

However, in practice there are lots of challenges to build a mobile camera system that is privacy-respecting. For example, *how do people define their privacy policies? how do they share the policies? how do the mobile cameras get the privacy policies from the surrounding people?* and *how to match a human object in the photo with someone in the surrounding area? how to protect people's privacy?* Motivated by these problems, in the recent years, especially because of the advancement on computer vision technology, researchers have proposed various novel systems [6, 8, 9, 14–17, 21, 26, 28]. However, because of limited application scope, special hardware requirement or dependence on Internet cloud, the solutions are still not ready for real-world deployment on mobile cameras.

In this work, we propose a new system named iRyP (acronym for "I Respect Your Privacy"): a purely edge-based privacy-respecting system for mobile cameras. In our system, privacy preferences of the photographed persons would be piggybacked into the advertisement messages of Bluetooth Low Energy (BLE). As such, privacy policies of people in a photo view can be delivered timely and automatically. In addition, we adopt the state-of-the-art face detection engines to complete the task of face detection and feature point extraction, and we employ a perceptual hashing algorithm for fast face matching. To improve accuracy, we also design several new techniques for face-related image processing. We implemented and evaluated a prototype system based on the Android platform. Our experiments in both offline and online settings show that our system can meet our design requirements and is practical to use.

The contributions of our work are listed below.

WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria

- First, iRyP completely runs on commodity mobile devices, which have relatively weak computation power. Different from many of the past works, iRyP neither relies on the Internet cloud for image processing and storage, nor require all users to register to host their personal information. As a purely edge-based end-to-end solution, it is readily deployable without the complexity and cost of setting up a cloud service, which also lacks the incentive to any business.
- Second, to improve the accuracy on face matching, we design a number of efficient techniques, including using multiple face hashes, face normalization, background reduction, and face orientation determination.
- Third, by piggybacking privacy profile in BLE advertising messages, our system provides a user-friendly way for people to share their privacy policies.
- Fourth, we design experiments for different scenarios to test the performance of our system, which shows that our system has high accuracy and good performance, and it is practical. We also build a mobile app for real-world deployment of our system for privacy-preserving photo sharing.

2 PRELIMINARIES

2.1 Bluetooth Low Energy (BLE)

Bluetooth low energy (BLE) [1] is a new generation wireless personal short-range network technology. Devices with BLE implemented can pair up to communicate with each other. Compared to classic Bluetooth, the main advantage of BLE is lower power consumption. BLE is widely used in our daily life. Mobile devices (iOS, Android, Windows Phone, etc) as well as some laptops already natively support BLE.

When we turn on BLE, it broadcasts advertising messages (also called beacon messages) to its surrounding. A nearby BLE-enabled device can discover it by reading advertising messages. An advertising message is 31-byte long, consisting of a flag part and one or multiple small AD structures [11]. The flag part takes three bytes. The first byte represents the length; the second byte is "0x01"; the third byte represents data. The flag part is a reserved part in any advertising message. Inside an AD structure, there are three components which are length, AD type, and AD data. The length field takes one byte; the AD type field takes three bytes (e.g., when the first byte has the value "0xFF", the other two bytes will represent the company identifier). For the most efficient use of advertising messages, one may include only one AD structure in the message ("0xFF" plus a non-used company identifier for iRyP to avoid interfering the existing BlueTooth protocols) In this way, one can get totally 24 bytes [1] in advertising messages to piggyback one's own data without affecting the normal operation of BLE.

2.2 Perceptual Hash Algorithm (PHash)

A perceptual hash algorithm [3] is an algorithm which generates a fingerprint for an image (or other type of multimedia file) based on various features in its content. Unlike cryptographic hash functions, which have the avalanche effect of small changes in the input leading to drastic changes in the output, perceptual hashing produces close outputs for two similar images. This feature makes it attractive for applications such as image searching, copyright infringement, digital forensics, etc. Next, we describe the details of pHash [4], one of the most famous perceptual hash algorithms.

- Step 1. Resize a given image to 32x32.
- Step 2. Convert the resized image into gray-scale.
- Step 3. Perform the DCT (Discrete cosine transform) to get a 32x32 DCT matrix.
- Step 4. Reduce the size of DCT and only keep the top-left 8x8 matrix.
- Step 5. Calculate the average DCT value for the 8x8 matrix.
- Step 6. Calculate the 64-bit hash value. For each value in the matrix, if equal to or greater than the average, it is set to '1'; otherwise, '0'.

From the above steps, one can clearly see it is like a highly lossy compression algorithm, which gives the robustness to small changes. Especially, given that the output size is very small (8 bytes), there could be a much higher chance of collisions than cryptographic hash functions. Moreover, the last step makes the process not reversible, similar to the one-way property in cryptographic hash functions although weaker. We will see in later sections (Section 5) that such properties make our proposed solution safer to use than user identification methods where a user needs to send his unique biometric identifier (such as facial biometric) to a stranger for privacy policy matching. In Section 7, we will present our detailed measurements on how illumination and facial expression may affect the accuracy of pHash-based facial matching.

2.3 Application Scenarios

In this work, we propose a privacy-respecting camera system named *I Respect Your Privacy* (iRyP). There are two types of human actors in iRyP. One is a camera user, who takes photos. For simplicity, we refer to this actor as the *user* with the name *Alice*. The second type of actor is a person (or people) in the camera view of *Alice*, whose name is *Bob*. The previous user studies [6, 10, 13] have shown that most users are willing to follow social norms by making people aware of recording and process photos accordingly (e.g., blurring their faces) once being notified of their privacy preferences. This motivates us the following design principle: *giving control back to people as much as possible*. This means that whenever feasible, recording and sharing decisions should be controlled by the people in the camera view, while not impeding camera users from using privacy-respecting camera systems.

In real-world scenarios, however, it is impractical for user (Alice) to ask each person (Bob) directly about his privacy preference before photo-taking, especially when Bob is a passerby in the scene and a stranger to Alice. A fundamental research challenge is: *how can Alice automatically learn the privacy policy of Bob and then enforce it?* Given the time, location and relationship between Alice and Bob, there might be multiple ways to represent and deliver privacy policies. In this research, leveraging the increasing popularity of mobile devices, we propose to install the privacy policy in mobile devices for automatic sharing. For example, Bob can define his privacy policy (e.g., others must blur his face in a photo) and load it into his smart watch (or smartphone). The smart watch broadcasts

[1] 31 bytes (Total advertising message length) - 3 bytes (Flag space) - 1 byte (Length) - 1 byte (0xFF) - 2 bytes (Company Identifier) = 24 bytes.

his policy to the surrounding through BLE. When Alice is taking a photo, her camera receives Bob's policy through BLE and tries to find whether Bob is in the view. If he is, face protection is applied to his face, according to his privacy policy.

Note that iRyP is incrementally deployable. That is, the iRyP-enabled camera is able to protect others as long as people's devices are able to notify their privacy policies. For other people in the view, by default iRyP does not protect any face at the photographing time, but protect strangers' faces at photo sharing time. We recommend this default policy considering a tradeoff between *visual friendliness* and *user privacy*. Users can define their own default policies as well.

2.4 Design Goals and Research Challenges

Under the above application scenario, next we discuss several specific design goals as well as the research challenges.

High Efficiency Compared to PCs, most mobile phones do not have powerful computing capability. Currently, many high performance face detection algorithms are based on deep learning algorithms. However, they require fast CPUs and high-performance graphics processing units (GPUs), and will not work smoothly in mobile cameras.

Ease of Use The system on both camera side and mobile device side should be easy to use. First, the whole process should be *automated* and *user intervention free*. As such, we cannot require BLE pairing between camera and mobile device for them to communicate, because pairing between unknown devices is neither automatic nor safe. Second, for pervasive use and immediate deployment, there should be no dependence on Internet access or relying on cloud computing. Existing systems like I-Pic [6] use the Internet cloud to help process photos and enforce policies. While this has the advantage of outsourcing expensive operations to the cloud and negotiating rich privacy policies, it is unclear who will have the business incentive to provide such a public service, and how to build and manage the trust among users.

High Accuracy In our system, the camera will need to match a human object in a photo with human identifiers it received from others through BLE, and then apply the right privacy policy. If a face in the photo is matched to a wrong person, it will cause a false match; on the other hand, if the face is not matched to the right person, it will cause a false non-match. Clearly, our system should have a high matching accuracy.

Security and Privacy In our system, a user needs to broadcast facial identification information and privacy policy to the surrounding. Such facial identification information must not contain sensitive information, and must not be misused by others to launch impersonation attacks against facial biometric based user authentication or online tracking attacks.

The above design goals motivate us to design a purely edge-based in-situ privacy-preserving camera system. Note that We do not consider a malicious photographer who is determined to take photos of others. Anyway, he or she can easily use spy cameras these days. At this moment, there is no known feasible solution to defeat that. Our design is motivated by multiple user studies [6, 10, 13] that indicate most people are willing to respect others' privacy preferences, but there is still a lack of practical solutions. Our work contributes by providing such a possible solution.

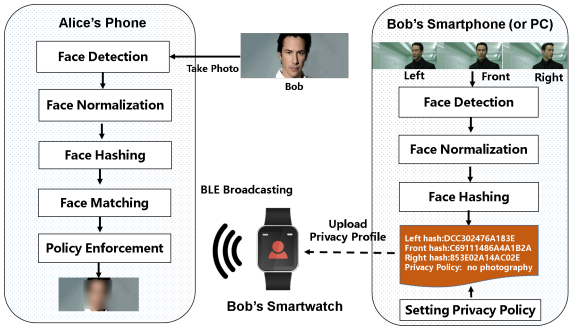


Figure 1: System Architecture of iRyP

3 SYSTEM ARCHITECTURE AND DESIGN

3.1 Architectural Overview

Fig. 1 shows the system architecture of iRyP. It shows how Bob generates his privacy profile offline (on the right side), and then uploads it to his mobile device (here a smartwatch). The mobile device broadcasts Bob's privacy profile periodically through BLE advertisements. When Alice takes a picture with Bob in the view, her device will process the photo and then apply Bob's privacy policy (on the left).

Overall, there are six technical modules in iRyP, which fall into three groups. The first group is to provide a user-friendly mechanism for sharing privacy profiles; the second group provide face-based human identification, and the third group is about setting up privacy policy and enforcing policy. Specifically, the first group contains only one module for *profile-sharing via BLE broadcast*. In the second group, *Face Detection* detects and extracts faces from a given photo; *Face Normalization* removes much noise from the background of a face and restores a face into the vertical position through rotation; *Face Hashing* converts a face into a 8-byte hash value for a front face, and 7 bytes for a left or right face, and a face hash serves as a face identifier; *Face Matching* matches faces based on face hashes. The third group is related to privacy policy. Specifically, *Setting Privacy Policy* allows Bob to set up his privacy policy (e.g., "no photographing") through an app GUI in his smartphone (or PC). Bob's privacy policy and face identifiers are combined into his *privacy profile*, which is uploaded to his mobile (e.g., a smartwatch) and periodically broadcast through BLE beacons. When Alice takes a photo, her camera device will receive Bob's privacy profile, and it will check if Bob is in her photo by matching with Bob's face hashes in the received privacy profile. If matched, her camera will apply Bob's privacy policy through the *Policy Enforcement* module. Next, we describe each group and its modules.

3.2 Profile-sharing via BLE Broadcast

While there are different ways for smart devices to talk to one another, BLE is the best choice for iRyP because it is getting ubiquitous these days and is low-energy. For BLE-enabled devices to communicate data, pairing is the first step, which typically involves a manual process. Clearly, pairing is not convenient for iRyP because of two reasons: (1) oftentimes the physical contact time is short (e.g., passing by), so pairing is a too slow process; (2) Bob,

especially as a passerby, is mostly likely unwilling to be interrupted. Hence, a challenge here is to design a user-friendly privacy profile sharing mechanism, which is completely automated and end user hassle-free.

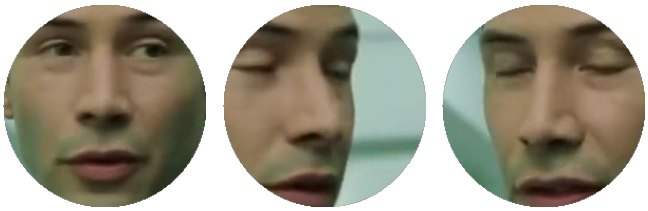
To address the challenge, first, we will not require pairing between devices. This is possible because in iRyP only one way communication is needed from Bob's mobile to Alice's camera. Indeed, as long as we are able to embed a privacy profile into BLE beacons, Alice's camera will receive Bob's privacy profile without needing a follow-up pairing process. From Section 2.1, we already know that BLE does offer a user customizable field, where users can insert their own content without affecting the normal BLE protocol and without introducing extra communication overhead. However, its maximum space is 24 bytes. That is, we are limited by 24 bytes for piggybacking profile information, which includes Bob's identifier and privacy policy. The identifier will help Alice's camera accurately identify Bob from her photo. We will see shortly that we may encode a privacy policy with 2 bytes, so there are only 22 bytes for us to encode identifiers.

3.3 Face-based Human Identification

An intuitive idea for face identification is through face matching, as long as Bob's facial biometrics can be encoded into 22 bytes. Alice's camera may extract facial biometrics from each face in the picture for matching. However, there are a number of challenges. First, well-known face recognition algorithms typically use a large vector to represent a face. Second, as facial recognition is increasingly used in biometric verification, broadcasting Bob's facial biometrics would be a great security and privacy risk.

To address the above two challenges, iRyP employs pHash, a perceptual hash algorithm[3] to compute a 8-byte face hash as identifier. From Section 2.2, it is clear that pHash has a very high compression rate. Especially, because human's faces are similar and perceptual hashing is designed to be nonsensitive to small changes of input, it is relatively easy to find collisions in face hashes. Fortunately, in our application setting, the number of people whose faces can be detected in a photo is mostly very small, so the chance of collisions can be well controlled by choosing appropriate thresholds. Note that *pHash is not reversible*. That is, based on the face hash of Bob, one cannot reversely construct his face, so the security and privacy risk of broadcasting face hashes as face identifier is little. Moreover, due to a good chance of collisions of face hashes in a relatively large population (e.g., hundreds of people), an attacker will have high false positives if he uses face hashes as a unique identifier to track someone else (more security and privacy analysis in Section 5).

There is another challenge for face identification. Unlike in biometric verification applications, where users are required to face cameras or pose in a certain way, in our application setting, where Bob is a passerby, Bob's face might be captured by others in different orientations (front or side view). Therefore, the identification accuracy would be very low if the face orientation of Bob in the photo is much different from that used by Bob to build his profile. Figure 2 shows an example. Here we extract the pHash value from the front face of a person and uses it as a baseline. For every new face, we compute its pHash value and measure the hamming distance between its pHash value and the baseline. The figure shows that the pHash values of his left face and right face have a very large hamming distance ($\geq$ 20) from the baseline. This will cause false non-matches. To address this challenge, in iRyP, Bob will generate his face-based identifiers with three faces: front face, left face and right face. With three face identifiers, there will be a higher face matching probability for Bob. Since we only have 22 bytes in total for identifiers, we will allocate 8 bytes for front face, 7 bytes for left face and 7 bytes for right face.



(a) Front face **(b) Right face** **(c) Left face**

Figure 2: pHash values of left and right faces have large hamming distances (20 and 27, respectively) from that of front face.

Given the above high-level idea for face identification, next we present some details, including face detection, face normalization, and face matching.

Face Detection: For face identification, the first step is face detection, which, given a photo, marks out each face in a rectangular face area. Over the years, many face detection algorithms have been designed and some of them are very popularly used, such as OpenCV with HOG cascade [30] or haar cascades [2], and they also have mobile versions. However, they have difficulty in detecting side faces from a large distance. Most recent techniques are based on deep learning algorithms, such as Faster R-CNN [23], and YOLO [22]. They are superior in detecting faces of different angles from far away, but require a powerful graphics card. Most mobile devices, however, do not have such graphics cards.

Not every algorithm can be employed in our setting, because of the following two considerations. First, as a passerby, Bob's face might be captured by others in different angles (front or side view). Clearly, the face detection algorithm must be good at detecting side faces. To test side face detection capability, we use a short video clip, which contains a face turning from left to right. We extract continuous frames from the video to test two face detection algorithms. Table 1 shows that when a face is turned over 20 degrees from the front view, OpenCV with haar cascades cannot detect it anymore. Second, the face detection algorithm should be able to detect face landmarks (semantic positions), which are needed for our face normalization module. Haar classifier does not detect face landmarks either.

Considering the above requirements, we instead adopt Seetaface [19]. Table 1 shows that Seetaface is good at detecting side faces. In our experiment, SeetaFace successfully detected all side faces, as shown in Table. 1. Moreover, it also locates landmarks, such as left and right eye center, nose tip and mouth center, in the detection process.

Table 1: Face detection result under different turning angles

Turning Angle with Front	OpenCV + Haar Classifier	SeetaFace
Small Angle (Within 20 degrees)	Detected	Detected
Big Angle (20 to 70 degrees)	No Detection	Detected
Side Face (70 to 90 degrees)	No Detection	Detected
Face landmarks	No Detection	Detected

We further tested the detection capability of Seetaface at different distances and camera resolutions, which also showed very good performance. For example, a face which is over 8 meters (or 12 meters) away can be detected in a 1920x1080 (or 2560x1440) photo. The time complexity of SeetaFace is also acceptable, as we will show in the evaluation section.

Finally, we crop every face in the photo and save it as an individual image in order to calculate its face hash. These cropped face images are the input to the next module.

Face Normalization: For face matching, it is important that two faces are compared in a similar position. Moreover, background and hair style should not introduce much noise.

Calibrating Head Rotation: When Alice is taking a photo with Bob in her camera view, the position of Bob's head may be rotated in some degree from the vertical 'T' pose (Fig. 3a), either clockwisely or counter-clockwisely. The rotation would introduce face identification errors if not corrected, because pHash is a rotation sensitive algorithm. Figure 3 shows that the hamming distance from that of the baseline 'T' position increases when keeping rotating the head position. Especially, when the angle of rotation reaches 10° or above, the distance would be big enough to cause false non-matches.

(a) The baseline T Pose Photo. **(b) Rotate by 10°. Hamming distance is 7.** **(c) Rotate by 25°. Hamming distance is 22.**

Figure 3: Hamming distance changes when rotating the 'T' pose photo

To eliminate the errors caused by head rotation, we use human eyes and nose as feature points (i.e., landmarks) to calculate the degree of angle the face is rotated clockwisely or counter-clockwisely. We then rotate the face photo in the reverse direction until two eyes and nose approximately reach the 'T' pose.

Background Noise Reduction: After applying a face detection algorithm (here Seetaface), we obtain face areas. However, face areas often include some background and also hair, which are subject to change. When calculating the hamming distance between two faces captured in a different time and location, background and hair are the noise factors, which will introduce errors in face matching. To address this problem, we use face landmarks to find the center of a face and then extract a circular face area based on an appropriate radius. We only keep the meaningful features like human eyes, eyebrows, nose, and mouth in our circular face areas for face matching. More specifically, after extracting face areas, we get a 256x256 square face area. Then we choose 96 pixels as the radius to get a circular area, as shown in Figure 3.

Face Orientation Determination: Figure 2 showed that the pHash values of Bob's left and right faces had a very large hamming distance from that of the baseline. To improve face matching accuracy, we have proposed to piggyback the pHash values of left, right and front faces into Bob's privacy profile. Now the question is: how do we know the orientation of a face in the photo? Once we correctly determine whether an extracted face is a left, right or front face, we can match it with the corresponding one in the privacy profile. Although some algorithms can calculate face turning angles with good accuracy, they are relatively expensive in computation. Next, we propose an efficient algorithm to determine face orientation.

In our face detection process, we have already extracted face landmarks. In Figure 4, we use green dots to represent the corresponding feature points. We calculate the angle between a vertical line (red) and the line (blue) going through four green dots on the bridge of the nose. Note that this angle is different from the actual face orientation (turning) angle. It serves our purpose because we are not to determine the exact face orientation angle, but to determine face orientation (left, right or front). If this angle falls in the range of $-\theta$ to θ, we treat it as a front face. We treat it as a right face if below $-\theta$ and left face if above θ. Here, θ is our threshold, which may be set differently. Accordingly to our empirical study, $\theta = 5^\circ$ is a good choice. The computation overhead is negligible.

Figure 4: Examples of face orientation determination

Face Matching: Due to space limit in BLE beacons, we have totally 22 bytes to represent faces. In our system, we use a 8-by-8 matrix (8 bytes) to represent a front face and a 8-by-7 matrix (7 bytes) to represent a left or right profile face. Each cell in a matrix takes 1 bit, so totally 22 bytes (176 bits) is used to represent a person.

After face normalization and orientation determination, we match faces in pair, one from the photo and the other from the profile. Our matching is based on the hamming distance between their face hashes. When the distance exceeds a threshold, they would be

considered to be from different people. Finally, we return the result of face matching (yes or no) to the policy enforcement module.

3.4 Generating and Enforcing Privacy Policy

Generating Privacy Policy Currently iRyP supports the following common policies.

- *Who_List*: people allowed to see *Bob*'s face. Examples are "none", "photographer only", "common friends", and "all".
- *Where_List*: allowed and/or disallowed photo/video sharing websites. The allowed domains start with '+', disallowed domains start with '-', empty for nowhere, * for anywhere. For example, "-youtube.com" indicates no sharing in YouTube.

We encode the above options into 16 bits. The first two bits encode the *Who_List*. For example, "00" stands for "none", which requires blurring the face before saving the photo into flash memory. "01" stands for "photographer only", which require no sharing with others online. The rest of 14 bits represent 14 pre-selected popular photo sharing websites for fine-grained access control, '1' means allow and '0' disallow. While the space limit does not allow us to select an arbitrary number of websites, in practice when the desirable websites are not included, one may combine with the *Who_List* to achieve coarse-grained access control. To enable fine-grained access control, one option is to use a large space to encode privacy policy information and then split the information into two or multiple beacon packets. This will require a slight change of our current protocol.

Policy Enforcement: If a face in Alice's photo matches with a face received from Bob via BLE, iRyP will apply Bob's privacy policy, for example, blurring his face in the photo if the *Who_List* bits are '00'. Otherwise, it applies the default policy Alice set up based on her own preference.

Policy regarding photo sharing requires an additional step. For example, if the *Who_List* bits are "01", no blurring will be done with the photo immediately. Instead, the blurring should happen before sharing. To implement this idea, for each face in the photo, we write its position and its associated 2-byte privacy policy into the photo file. The position of each face is represented by a rectangular region (i.e., top-left and bottom-right coordinates) in the photo, which was extracted in the face detection phase. To guarantee the readability of an augmented photo file, we add a new section to the original photo file. Take JPEG picture as an example. In a JPEG file, the image part always ends with bytes "0xFF D9". When a photo viewer software reads the picture, it will stop reading when it encounters "0xFF D9". Accordingly, we add a new section after "0xFF D9", which starts with "0xFF DA" and ends with "0xFF DB". The space between "0xFF DA" and "0xFF DB" is used to record privacy policy data. Besides the above customized method, we may adopt an standard one by adding the above data as a type of EXIF metadata into the JPEG file (however, some users may disable EXIF for better privacy).

When Alice wants to share the augmented photo file, she first loads it into her iRyP client app, which has a GUI for Alice to select the destination of sharing. Then the app processes each face based on the sharing destination and the embedded policy, e.g., blurring Bob's face in the previous example. Finally, it outputs a policy-compliant photo, which can be shared by Alice to the destination site.

In this work, we do not focus on face protection techniques. Although face blurring is a common solution, we understand it may cause a photo with blurred face(s) to look weird or ugly. There are also visual-friendly face denaturing solutions, for example, altering gender, race, age, and identity in a flexible way [25].

4 IMPLEMENTATION AND EVALUATION

We choose Samsung Galaxy S8 Plus (Android 7.0 Nougat, 4GB RAM) to implement and deploy our system, and to test the feasibility of our design. The entire system has around 8 thousand lines of code (Including C/C++ and Java). Specifically, we import into our project the Android version of SeetaFace [5], which is a library file. There is no SSE (Streaming SIMD Extensions) to optimize code for the mobile platform, which is one reason SeetaFace runs slower in mobile phones than in PCs.

For vision operations, we choose OpenCV 2.4.9. We import some libraries, like "libopencv_java.so", from the OpenCV android version, and install OpenCV manager on the mobile device. After that, we can use functions like "resize()" to process images. For BLE operations, we use the classes and APIs provided by the Android framework, such as "mBluetoothManager" and "mBluetoothLeAdvertiser.startAdvertising()".

To measure the accuracy of our system, we use the standard metrics such as *false positive*, *false negative*, *precision*, *recall* and *F-score*. Our evaluation is done in two stages. First, we evaluate the performance of facial hashing in different conditions. Especially, we will evaluate the sensitivity of facial hashing to the change of illumination and facial expression. The results indicate that our system has very good resistance to such changes, as long as they are not extreme. (Due to space limit, the detailed results are presented in Appendix A). Second, we perform an in-lab, large-scale, comprehensive evaluation with photos and videos downloaded from online. We also conduct in-wild evaluations in various settings.

4.1 Evaluation on Orientation Determination

To evaluate the accuracy of face orientation detection, we use a video clip in which a human's face turned from left to right in totally close to 180 degrees. The video clip was broken into 65 continuous frames. Our face orientation determination algorithm classified these 65 frames into three groups (front, left, and right). The false positive number was 3, which means 3 frames were classified into a wrong group. The false negative number was 0, meaning that faces in all frames were detected and their orientations were determined. The precision is 95.38% and recall is 100%. All the three false positives were similar and caused by extremely large turning angles, where right eyes are invisible.

4.2 Evaluation on Face Matching

To evaluate face matching accuracy, we first divide people into two categories, called *known people* and *unknown people*. Known people represent those whose privacy profiles are in the profile database of camera users (through BLE broadcasting or pre-loaded). On the contrast, unknown people represent those not using our system, so their faces do not match with known people. Note that for a comprehensive and efficient evaluation of face matching accuracy, we will not run our entire system in the wild for hundreds

Table 2: Unknown people face matching test result (Database group size 10. The letter "F", "L", "R" represent front face, left face and right face, respectively. The number following each letter is the Hamming distance threshold)

	True Positive	False Positive	Precision
F 5	379	0	1.0000
"F" 6	375	4	0.9894
"F" 8	367	12	0.9683
"F" 10	355	24	0.9367
"F" 12	300	79	0.7916
L 5	88	0	1.0000
"L" 6	88	0	1.0000
"L" 8	85	3	0.9659
"L" 10	76	12	0.8636
"L" 12	59	29	0.6705
R 5	83	0	1.0000
"R" 6	81	2	0.9759
"R" 8	78	5	0.9398
"R" 10	68	15	0.8193
"R" 12	50	33	0.6024

or thousands of times, which is very time consuming and labor intensive. Instead, we use an in-house experiment to achieve the same evaluation purpose.

Specifically, to simulate 10 known people, we download 10 movie clips, and each of them contains the character's face (e.g., Keanu Reeves in "The Matrix Reloaded", Hyo-joo Han in "The Beauty Inside"), turning from left to right. The videos are then broken into continuous frames, as we did before. The frames are then divided into three sets, based on face orientation and the middle one of each frame set is selected. In this way, we obtain three frames from each person and use them as input to our system to construct a privacy profile for this person. In total, we obtain a group of 30 faces for these 10 people, and we call it the *database* group.

Unknown People Face Matching Test In our first test, to simulate unknown people, we download 550 face photos from the Internet. They together form a testing group, consisting of 379 front faces, 88 left faces and 83 right faces. This test is to show how likely a random person (not using iRyP) captured in a photo is able to match someone in the database group of size 10. When there is a match, it is a false positive. Table 2 shows that with threshold hamming distance 5, there is no false positive for all 550 unknown people. The false positive rates increase with hamming distance thresholds. When the threshold is 8 or 9, the precisions are around 95%. This is because with a larger threshold, more face hashes are considered the same, and hence more false matches. Note that, when the database group size is smaller (e.g., only a few people nearby the photographer are using iRyP), false matching will be very unlikely. On the other hand, it can be expected that when the database group size is larger, the false positive rate could increase unless we decrease the hamming distance threshold. We consider group size 10 is a relatively big number in a real world setting.

Table 3: Known people face match test result (In first column, the letter "F", "L", "R" represent front face, left face and right face, respectively. The number following each letter is the Hamming distance threshold. "TP", "FP", "FN" represent true positive, false positive, False Negative)

	TP	FP	FN	Precision	Recall	F-Score
"F" 6	60	0	12	1.0000	0.8333	0.9091
"F" 8	63	0	9	1.0000	0.8750	0.9333
"F" 10	65	0	7	1.0000	0.9028	0.9489
"F" 12	69	42	2	0.6216	0.9718	0.7582
"L" 6	45	0	24	1.0000	0.6522	0.7895
"L" 8	52	5	17	0.9123	0.7536	0.8254
"L" 10	59	20	9	0.7468	0.8676	0.8027
"L" 12	63	89	4	0.4145	0.9403	0.5753
"R" 6	41	2	26	0.9535	0.6119	0.7455
"R" 8	50	15	14	0.7692	0.7813	0.7752
"R" 10	53	40	9	0.5699	0.8548	0.6839
"R" 12	56	113	2	0.3314	0.9655	0.4934

Known People Face Matching Test Recall that when we construct the original database group, we divided the frames of 10 known people into three sets, based on face orientation, and the middle one of each frame set is selected as a face identifier. Now we use all other frames as the testing group, which contains 72 front face frames, 69 left face frames, and 67 right face frames, in total 208 frames.

Next, we test how accurate our system is to match known people. For this purpose, we may directly compare the faces from the testing group against their profile faces in our database group. In this study, we however choose to perform a "stress" test, in which we increase the population of database group by adding a distractor group. The distractor group includes the left, right, and front faces of 28 additional people. In the end, the database group contains 114 faces of 38 people. Then we match the testing group (208 frames) against the new database group of 38 people. When a face is matched to the right person in the new database group, it is a true positive; otherwise, if it is matched to a wrong person, it is a false positive. Note that for some threshold values, it is possible for one face to match multiple faces of different people. In this case, we count true positives and false positives separately. That is, there could be multiple false positives out of one test case. A false negative (false non-match) happens when there is no matching at all or when matched to a wrong person.

Table 3 shows that false negatives and precision decreases with hamming distance threshold. This result is consistent with that in Table 2. On the other hand, both false positives and recall increase with hamming distance threshold. Clearly, this is a tradeoff between precision and recall. Based on our testing dataset, it looks the threshold value of 10 or 11 is good for front faces, 8 is good for

Table 4: Average Time Cost

Process	Average Time (ms)
One Face Detection	792
Two Faces Detection	928
Three Faces Detection	1057
Four Faces Detection	1178
Five Faces Detection	1304
One Face Feature Extraction	427
Face Normalization	Negligible
Face Orientation Determination	Negligible
Calculating pHash	Negligible
Face Matching	Negligible

side faces. Overall, false matching is less an issue here, but false non-matching is harder to eliminate, especially for side faces. This is not surprising because the turning angles of some faces in our testing group are very large, which makes it even difficult for users to identify the faces.

By default, our system adopts a more strict privacy policy when there are multiple matches in one case. For example, suppose a face in a photo matches both Bob's and Charlie's. Bob's policy is "no photographing" and Charlie's policy is "all people can see my face", our system will enforce the more strict privacy policy of the two, i.e., blurring the face.

4.3 Efficiency

In our system, we adopt SeetaFace as our face detector and our modified version of perceptual hash algorithm for face matching. The question is: how efficient and scalable is our system, especially when there are multiple people in the view?

We first design an experiment to test the performance of SeetaFace in our phones. We prepare five photos of the same size (1600x400). The number of faces in them increases from 1 to 5, and each face has the same size. Then we apply SeetaFace to each photo 100 times and record the processing time. The test result, in the top 5 rows of Table 4, shows that the face detection algorithm has the linear time complexity w.r.t. the number of faces in a photo. On average, each additional face requires about 130ms to detect.

Next we break down the time cost for different modules in our system, as shown in Table 4. Besides face detection, another major source of time cost is face feature extraction with SeetaFace, which is about 427 milliseconds per face. The time costs of all other modules, including face normalization, background noise reduction and face orientation determination, are negligible.

Overall, the performance evaluation show that our system has good efficiency based on limited computational capability in mobile phones. Note that on the camera side, we may implement part of the system as a service, which does photo processing in the background. In this way, users can keep using their devices during seconds of processing delay.

4.4 Real-world Testing

To evaluate iRyP in the real world scenarios, we have further prototyped it on Android 7.0 by providing a photographer app and a bystander app. The photographer app was the main implementation of iRyP's core methods, including receiving BLE advertisement messages, taking pictures, face detection, face normalization, pHash generation, and policy enforcement. The bystander app implemented the privacy profile broadcast feature with the ability to change the payload of BLE advertisement messages based on the input of a privacy policy and pHash values. In the deployment, we used a Samsung Galaxy S8 Plus as the photographer's device to capture pictures. Other Android devices, including Samsung Galaxy S8 Plus and Motorola Nexus 6 were used as the bystanders' devices to broadcast privacy profiles.

Dataset: In our test, we asked 14 friends to "act" as the intended subjects and bystanders to be photographed in different scenarios. We notified the friends in advance and obtained their consent to take photos for personal use and would not share them to the public (including not disclosing their faces in this paper). Specifically, five friends registered to our system with their privacy profiles – two deployed the "no photographing" policy and three "photographer only" (meaning although the photographer can view their faces, sharing is not allowed). The others were not users of our system, so we applied default policies.

We used Samsung Galaxy S8 plus to take 100 pictures in different locations, including classroom, playground, parking lot, lobby, and hallway, to demonstrate various light conditions and backgrounds. Scenarios of "acted" bystanders passing by were also included in this experiment. There were totally 270 faces in the 100 pictures, and 118 faces belonged to the five registered people. Out of these 118 faces, 52 were front faces, 29 right faces, and 37 left faces.

To test face detection rate and face match accuracy, we use the Samsung Galaxy S8 Plus to run tests on all the pictures taken. Before we started the process of face detection, we made sure that the photographer app received the privacy profile BLE messages of all the registered people sent from the Android devices running the bystander apps.

Figure 5 demonstrates two real-world usage cases of our Android app for photographers. Each case shows the screenshots of three major steps: (1) loading and processing picture in our app; (2) sharing picture through our app; (3) shared picture seen from a target app. Some faces have been intentionally masked with a movie star's face for protecting their privacy from readers.

Case 1 (the upper row) includes two of our registered people. The Who_List policy we received from person A (second from the left) is "01", which stands for "photographer only", and the policy we received from the other person B (third from the left) is "00", which stands for "none". Accordingly, during photographing, B's face was blurred right away before the picture was saved to the flash memory. Later on, when we loaded the picture from the flash memory via our app (the leftmost image of the upper row), B's face was already blurred (other faces were temporarily masked with the same movie star's face). When we pressed the "share" button (the middle image), A's face was also blurred to reflect her privacy policy. In this case, we chose to share the picture over Instagram.

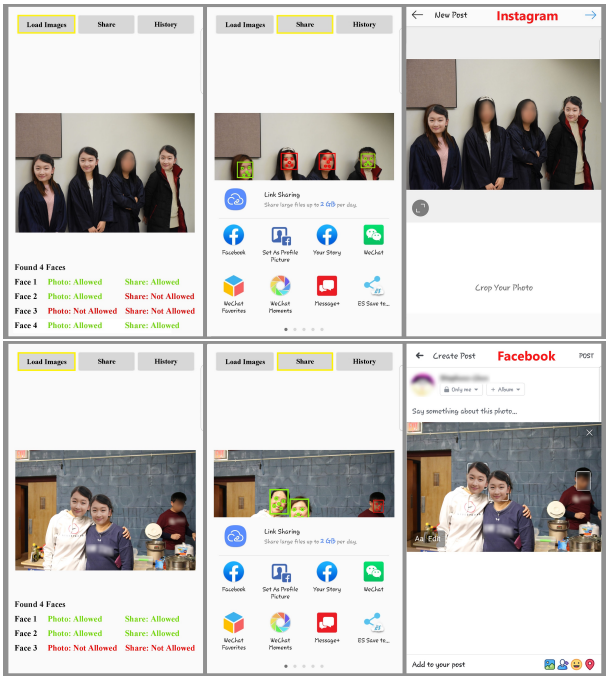


Figure 5: Two sets of screenshots showing the entire process of using our photographer app for loading and sharing a photo to a third-party app while enforcing people's privacy policies. The upper row is to share a photo to Instagram and the lower row is to Facebook.

The Instagram sharing page shows that the faces were already blurred according to their sharing policies.

Case 2(the lower row) shows two people in the foreground whose policies allowed for both sharing and photography. The person who was a bystander in the picture is shown in the back with his side face blurred because of his "no sharing no photography" policy. Upon loading the picture from the flash memory, the bystander's face was already blurred. In case 2, we choose to share the picture over Facebook, which is demonstrated in the rightmost screenshot with the bystander's face already blurred.

Face Detection Rate: Out of 100 pictures, 98 pictures had all the faces detected by the SeetaFace detector on Android at a detector resizing resolution of 800*600. The two faces that failed to be detected were a face at an extreme angle with only a visible left eye and a face that was significantly smaller due to far distance. Setting the resizing resolution to 1200*900 fixed the missed detection of the smaller face but at a cost of nearly doubling the processing time. Resizing resolution of 600*450 was also tested, but it did poorly in detecting all the faces (69 out of 100), and only had a marginal detection performance boost.

Face Match Rate: According to the outcomes from the previous Unknown or Known People Face Matching Tests (Section 4.2), we set the threshold hamming distance to 10 for front faces, and 8 for side faces. Overall, the results of our real-world evaluation largely confirmed the previous evaluation results with unknown and known people face matching tests, as shown in Table 5. For front faces, we found that in certain light conditions, the shades on the face could make the face look drastically different from the face used for registration, and this could even be a difficult job for normal people to recognize. For side faces, we found that the accuracy was sensitive to face angles. In cases that the face turned at a much larger angle than the registration faces, it was almost impossible to match.

Table 5: Real-world face matching results

	Precision	Recall	F-score
Front	0.9565	0.8462	0.8980
Side	0.9375	0.7143	0.8108

Power Usage: The battery consumption for the photographer app was roughly 1% (i.e., 35 mAh) when processing 8 pictures with 27 faces in total.

For the bystander app, power consumption for sending out BLE messages depends on the beacon interval. Based on BLE specification, the advertising interval can be set to a value between 20ms and 10.24s. Clearly, there is a trade-off between power consumption and device discovery latency. In fact, we can select from three different modes, provided by Android APIs. They are low latency, balance, and low consumption. Considering our application scenarios, we choose the balance mode, which sets the interval to approximately 1000 millisecond. We tested the power consumption for BLE broadcasting, and the overhead is negligible.

5 DISCUSSIONS

Security and Privacy Analysis: In Section 3.3, we mentioned that pHash is not reversible, and it is not a facial biometric. Hence, unlike a system that shares facial biometric directly for matching, using pHash will not introduce risks for real-world user authentication. However, will our scheme enable user location tracking, because Bob keeps broadcasting his profile?

Certainly, an attacker Trudy can physically follow Bob to easily continuously track him, e.g., based on any WIFI/BLE beacon messages (in our case a beacon includes the facial hashes) [7]. This is not particular to iRyP. Instead, we consider the special scenario where Trudy wants to track Bob without physically following him. She may download Bob's photos online or secretly take a photo of Bob, and then computes face hashes for Bob. When she is outside, she uses a device to record all privacy profiles broadcast via BLE channels from her surrounding (or by collusion someone else does it and shares the collected privacy profiles with her). Then she tries to match face hashes to see if Bob is in a certain location. Will this attack be effective in our system?

To answer this question, we note that while pHash is effective in distinguishing Bob from other people in the same view, where the number of people is often small, it is prone to many collisions in a large population (in the above attack, it corresponds the number of people Trudy has encountered over time). Table 6 shows the number of pairwise collisions among 379 unique faces increases with the distance threshold d. Even with $d = 5$, there are 731 collisions. Note that one may use a very small d (e.g., $d = 1$) to reduce false positives, but this will introduce high false negatives, because in iRyP Bob

uses a private set of photos to generate his face hashes. As such, using face hash as a unique identifier for user location tracking will not be effective.

Improving Matching Accuracy: Since the brightness and contrast may introduce pHash distance between faces of the same person, we may slightly expand the profile generation process to mitigate the impact. Specifically, Bob may generate two sets of face identifiers, one based on a relatively darker setting and the other based on a brighter setting. Depending on the actual brightness (smart devices often have light sensors to measure the brightness level of the environment), it may adopt and transmit the proper face identifiers.

6 RELATED WORK

Our research is motivated by the findings from a number of user studies [6, 10, 13], that (1) users mostly like to deal with privacy issues right after photo capturing to avoid the burdensome of photo management in a later time; (2) bystanders do expect that recorders can respect their privacy and first get their consent; (3) the privacy policies should be customized by people in the view. Next, we focus the literature review on the technical solutions for privacy respecting camera systems. As a malicious attacker can easily spy on others, neither the existing systems nor our system aims to defeat determined attacks.

Place Avoider [27] relied on visual tags and general patterns to recognize a space as sensitive. Roesner et al. [24] also proposed their venue-based solution. The idea is to enable the objects associated with sensitive places to "transmit" the pre-defined privacy policy by advertising messages or using visual labels. In the work of Wu et al. [29], low-cost sensors embedded in smartphones and smart watches can be used to discriminate the sensitive space from others. Bo et al. [8] proposed *privacy.tag*, a system for preserving privacy of the subjects in photos. In their system, people use QR code printed in their T-shirts to express their privacy policies. Photographer's Camera scans the QR code to learn people's privacy policies. While interesting, this approach is not intuitive or convenient to deploy. Raval et al.[21] proposed to pick up capture policies using visual marks. However, the privacy preferences of human subjects in photos are not the objective of their study. Jung et al. [17] proposed a method for transmitting privacy policies of bystanders through infrared technology and recognizing gestures. This approach however requires extra equipment for users' camera devices. He et al. [12] proposed an image perturbation technique to "encrypt" the sensitive areas in an image, and it supports popular image transformations. Ilia et al. [14] proposed Face/Off, a new access control mechanism to prevent the privacy leakages from photos in social network such as Facebook. Similarly, HideMe [18] is a plugin to existing photo sharing OSNs for preserving users' privacy. Different from our work, which is a comprehensive system for mobile camera and covers both the photo-taking phase and sharing phase, the above two systems are designed for the sharing phase only and all processing is done in PCs or in the cloud.

The work most relevant to ours is [6] *I-Pic*, a system to protect bystanders' privacy. In I-Pic, each person (bystander) has an online agent called bystander agent, which trains a classifier based on bystander's photographs. Each camera user has an online agent called capture agent, which talks to a bystander agent for secure matching. Unlike our system, the I-pic platform relies on the cloud computing platform to help match faces, and the camera needs a GPU to process images. It is not clear who would have the incentive to provide cloud services for supporting large-scale deployment of such systems. In our system, we piggyback three face hashes and the privacy policy into the advertisement messages. Without cloud support, our system cannot perform many advanced AI and crypto algorithms. However, it has the advantage of providing purely edge-based, in-situ privacy protection, which makes it readily deployable.

7 CONCLUSIONS

In this paper, to address the privacy concerns of people who are non-willingly photographed by others, we presented iRyP, an efficient privacy-respecting system for mobile devices. We designed several efficient and effective mechanisms to boost up the accuracy of face matching based on perceptual hashing algorithm. The biggest advantage is that it largely reduces the required data for face feature information while still providing high accuracy. We implemented an Android-platform based prototype, and performed various experiments. We showed that our system has high usability and good accuracy. In our future work, we will migrate the advertisement of policy messages to wearable devices such as smartwatches.

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their helpful feedback. This project was in part supported by NSF grant CNS-1618684. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

REFERENCES

[1] [n.d.]. Bluetooth Low Energy. https://en.wikipedia.org/wiki/Bluetooth_Low_Energy/. Accessed November 26, 2017.
[2] [n.d.]. OpenCV https://opencv.org/ Accessed Nov 12,2017. ([n. d.]).
[3] [n.d.]. Perceptual hashing. https://en.wikipedia.org/wiki/Perceptual_hashing. Accessed November 26, 2017.
[4] [n.d.]. pHash. https://www.phash.org. Accessed November 26, 2017.
[5] [n.d.]. Seetaface Android. http://blog.csdn.net/wuzuyu365/article/details/53468631. Accessed November 26, 2017.
[6] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. 2016. I-pic: A platform for privacy-compliant image capture. In *ACM Mobisys*.
[7] Johannes K. Becker, David Li, and David Starobinski. 2019. Tracking Anonymized Bluetooth Devices. *PoPETs* 2019, 3 (2019), 50–65. https://doi.org/10.2478/popets-2019-0036
[8] Cheng Bo, Guobin Shen, Jie Liu, Xiang-Yang Li, YongGuang Zhang, and Feng Zhao. 2014. Privacy. tag: Privacy concern expressed and respected. In *ACM SenSys*.
[9] Loris D'Antoni, Alan M Dunn, Suman Jana, et al. 2013. Operating System Support for Augmented Reality Applications.. In *HotOS*.
[10] Tamara Denning, Zakariya Dehlawi, and Tadayoshi Kohno. 2014. In situ with bystanders of augmented reality glasses: Perspectives on recording and privacy-mediating technologies. In *ACM conference on Human factors in computing systems*.
[11] Bluetooth Special Interest Group. [n.d.]. Specification of the Bluetooth system. https://www.bluetooth.com/specifications/bluetooth-core-specification. Accessed November 26, 2017.
[12] Jianping He, Bin Liu, Deguang Kong, Xuan Bao, Na Wang, Hongxia Jin, and George Kesidis. 2016. PUPPIES: Transformation-Supported Personalized Privacy Preserving Partial Image Sharing. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, Toulouse, France, June 28 - July 1, 2016*. 359–370. https://doi.org/10.1109/DSN.2016.40
[13] Roberto Hoyle, Robert Templeman, Steven Armes, Denise Anthony, David Crandall, and Apu Kapadia. 2014. Privacy behaviors of lifeloggers using wearable

Table 6: Number of Pairwise Collision

Distance Threshold	5	6	7	8	9	10	11	12	13
Pairwise Collision Number	731	1384	2415	3805	5621	7766	10447	13605	17091

cameras. In *ACM UbiComp*.
[14] Panagiotis Ilia, Iasonas Polakis, Elias Athanasopoulos, Federico Maggi, and Sotiris Ioannidis. 2015. Face/off: Preventing privacy leakage from photos in social networks. In *ACM CCS*.
[15] Suman Jana, David Molnar, Alexander Moshchuk, Alan M Dunn, Benjamin Livshits, Helen J Wang, and Eyal Ofek. 2013. Enabling Fine-Grained Permissions for Augmented Reality Applications with Recognizers.. In *USENIX Security Symposium*. 415–430.
[16] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. 2013. A Scanner Darkly: Protecting user privacy from perceptual applications. In *IEEE Symp. on Security and Privacy*.
[17] Jaeyeon Jung and Matthai Philipose. 2014. Courteous glass. In *ACM UbiComp*.
[18] Fenghua Li, Zhe Sun, Ang Li, Ben Niu, Hui Li, and Guohong Cao. 2019. HideMe: Privacy-Preserving Photo Sharing on Social Networks. In *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*. 154–162. https://doi.org/10.1109/INFOCOM.2019.8737466
[19] Xin Liu, Meina Kan, Wanglong Wu, Shiguang Shan, and Xilin Chen. 2017. VIPLFaceNet: an open source deep face recognition SDK. *Frontiers of Computer Science* 11, 2 (2017), 208–218.
[20] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 94–101.
[21] Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Lanodn P Cox. 2016. What you mark is what apps see. In *ACM MobiSys*.
[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *IEEE CVPR*.
[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*.
[24] Franziska Roesner, David Molnar, Alexander Moshchuk, Tadayoshi Kohno, and Helen J Wang. 2014. World-driven access control for continuous sensing. In *ACM CCS*.
[25] Terence Sim and Li Zhang. [n.d.]. Controllable Face Privacy. In *11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015, Ljubljana, Slovenia, May 4-8, 2015*.
[26] Christopher Smowton, Jacob R Lorch, David Molnar, Stefan Saroiu, and Alec Wolman. 2014. Zero-effort payments: Design, deployment, and lessons. In *ACM UbiCompu*.
[27] Robert Templeman, Mohammed Korayem, David J Crandall, and Apu Kapadia. 2014. PlaceAvoider: Steering First-Person Cameras away from Sensitive Spaces.. In *NDSS*.
[28] Matthew Turk and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3, 1 (1991), 71–86.
[29] Muchen Wu, Parth H Pathak, and Prasant Mohapatra. 2015. Enabling privacy-preserving first-person cameras using low-power sensors. In *IEEE SECON*.
[30] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. [n.d.]. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*.

APPENDIX A

Here we evaluate the sensitivity of facial hashing to the change of illumination and facial expression. This is because, in reality, the face images taken for profile generation likely differ from those taken on-site because of the changes of these factors. While such changes will also affect the accuracy of other face matching algorithms, here we focus on evaluating their impact on our facial hashing algorithm, given the input of normalized faces.

7.1 Sensitivity to Illumination

For illumination sensitivity test, we change both the brightness and the contrast levels of a given face image (Figure 7a). Specifically, we use two variables B and C, initially both set to 1.0 for the original image and varying between $[0.1, 2.0]$ at the interval of 0.1, to control the changes of brightness and contrast levels. We use a heatmap to show the results, as in Figure 6. The columns represent the contrast levels, and the rows represent the brightness levels. For each cell, the value represents hamming distance value. Darker blue color indicates bigger distances, whereas lighter yellow color indicates a smaller distance.

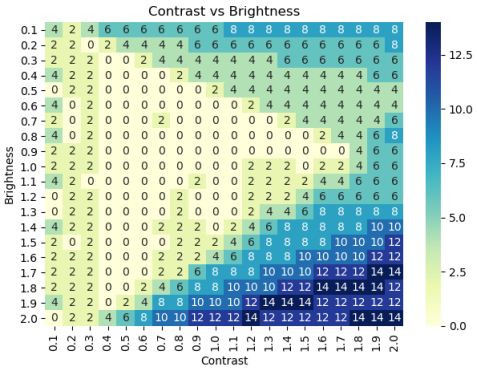
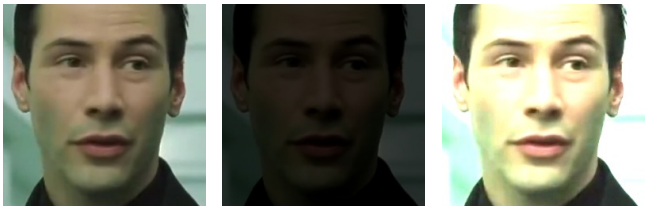


Figure 6: Heatmap of Hamming Distance under Different Brightness and Contrast

From the heatmap, we can observe that in most cases, the change of brightness and contrast levels only introduce a small hamming distance from the original image. Especially, it is not very sensitive to darkness (an example in Fig. 7b), but is sensitive in very bright and high contrast cases (an example in Fig. 7c). The reasons are as follows. In Step 6 of pHash computation, the binary output for each pixel (1 or 0) is based on whether the pixel value is above the average or not. A darker image has a smaller pixel average, but that does not affect much the polarity of each pixel with respect to the average. However, when changing to a very bright and high contrast case, many pixels of the original image would reach the maximum pixel value of 255 (hence truncated to 255) during the calculation. This not only introduces high frequency values in the DCT matrix, but also cause skewness of averages.

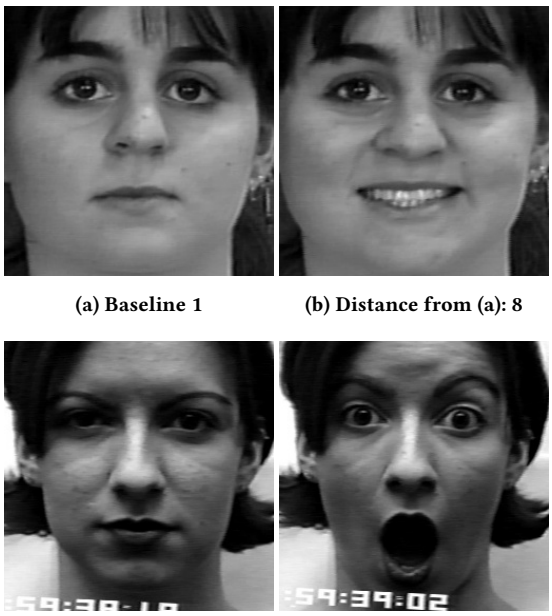
7.2 Sensitivity to Facial Expression

To test the sensitivity of facial hashing under different expressions, we use the CK and CK+ expression dataset [20]. The dataset has different kinds of expression classes. Take the "happy" class as an example. It has many groups of continuous photos of a person from natural expression to smiling or laughing. The CK and CK+ dataset is very big, so we *randomly* choose some groups of photos from the following classes, including anger, disgusted, fear, happy, sadness, and surprised. As Figure 7 shows, there are totally 217 photos for testing (anger 40, disgusted 31, fear 22, happy 48, sadness 42, and



(a) The original image under the brightness 1.0 and the contrast 1.0 **(b) The distance is 0 under the brightness 0.2 and the contrast 0.3** **(c) The distance is 14 under the brightness 2.0 and the contrast 2.0**

Figure 7: An Example showing how brightness and contrast affect hamming distance.



(a) Baseline 1 **(b) Distance from (a): 8**

(c) Baseline 2 **(d) Distance from (c): 8**

Figure 8: Expression test examples

surprised 34). For each selected photo, the face is detected and normalized according to our standard procedure, Then for each group of photos, we choose the first photo as the baseline and compute the hamming distance between the baseline photo and every other photo of the group. Here we set 10 or 11 as the threshold distance for face matching. Figure 7 shows the total number of photos in each class and the number of fails. The overall accuracy is 0.86, which indicates the facial hashing is good to match different expressions at the given threshold value. The main reason for failures is because of the extreme facial expressions in the database.

Figure 8a and Figure 8b show a successful match in the "happy" class. The distance is 8, which is mostly caused by the open mouth. Figure 8c and Figure 8d show another successful match in the "surprised" class. The distance is 8, which is also mainly because of the widely opened mouth and raised eyebrows.

Table 7: Expression test classes composition and accuracy

Class	Total # Faces	# Fails	Accuracy
anger	40	0	1.00
disgusted	31	0	1.00
fear	22	7	0.68
happy	48	7	0.85
sadness	42	6	0.86
surprised	34	11	0.68
Total classes	217	31	0.86

The above evaluation on the sensitivity of our facial hashing shows that with proper thresholds as the hamming distances, facial hashing indeed exhibits robustness to some degrees of changes of illumination and expression. In reality, these two factors, together with facial orientation, may jointly contribute to hamming distance, either by increasing it further or reducing it by mutually canceling out some effects. As a result, our system will not set a threshold distance by simply adding up the threshold values of different impacting factors. There is no dataset for a joint evaluation; therefore, we show the overall performance using real-world testing cases (Section 4.4), which naturally combine multiple factors together.

APPENDIX B

Offline and In-Situ Policy Generation To use iRyP, users may generate their privacy profiles offline and share with their family members and friends in advance. For Alice, she can add their privacy profiles (i.e., 24 bytes generated from three face photos) into a local database for future use. In this way, there will be no need for realtime profile sharing via BLE. For passers-by, since the encounter may only happen once, there is no need to store their profiles in Alice's camera. As such, the profile database will remain very small.

In practice, it may happen that Alice needs to add someone into her profile database *in-situ* (or temporally change the privacy policy of an existing person for special photos, which can be treated the same way). For example, Alice wants to take a photo of her new friend Carol, who is not using iRyP. Different from Bob who is a passerby, here Carol is the intended object for photographing, although her privacy policy is not in Alice's camera. In this case, Alice's camera may apply a default policy of her selection (e.g., "photographing and sharing with common friends"). However, if Carol prefers Alice to not share his face, Alice can manually change the privacy policy for Carol in the augmented section of the photo file through the iRyP client application, as described in Section 3.4.