TargetFinder: Privacy Preserving Target Search through IoT Cameras

Youssef Khazbak
The Pennsylvania State University
University Park, Pennsylvania 16802
ymk111@psu.edu

Junpeng Qiu
Facebook
Menlo Park, California 94025
qjpchmail@gmail.com

Tianxiang Tan
The Pennsylvania State University
University Park, Pennsylvania 16802
txt51@psu.edu

Guohong Cao
The Pennsylvania State University
University Park, Pennsylvania 16802
gxc27@psu.edu

ABSTRACT

With the proliferation of IoT cameras, it is possible to use crowdsourced videos to help find interested targets (e.g., crime suspect, lost child, lost vehicle, etc.) on demand. Due to the ubiquity of IoT cameras such as dash mounted cameras and phone camera, the crowdsourced videos have much better spatial coverage compared to only using surveillance cameras, and thus can significantly improve the effectiveness of target search. However, this may raise privacy concerns when workers (owners of IoT cameras) are provided with photos of the target. Also, the videos captured by the workers may be misused to track bystanders. To address this problem, we design and implement TargetFinder, a privacy preserving system for target search through IoT cameras. By exploiting homomorphic encryption techniques, the server can search for the target on encrypted information. We also propose techniques to allow the requester (e.g., the police) to receive images that include the target, while all other captured images of the bystanders are not revealed. Moreover, the target's face image is not revealed to the server and the participating workers. Due to the high computation overhead of the cryptographic primitives, we develop optimization techniques in order to run our privacy preserving protocol on mobile devices. A real-world demo and extensive evaluations demonstrate the effectiveness of TargetFinder.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; • **Computer systems organization** → *Sensors and actuators*; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*; • **Networks** → Mobile networks;

KEYWORDS

Privacy preserving, crowdsourcing, on-demand, target search, IoT cameras, worker selection.

IoTDI '19, April 15–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6283-2/19/04…$15.00
https://doi.org/10.1145/3302505.3310083

ACM Reference Format:
Youssef Khazbak, Junpeng Qiu, Tianxiang Tan, and Guohong Cao. 2019. TargetFinder: Privacy Preserving Target Search through IoT Cameras. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3302505.3310083

1 INTRODUCTION

The ubiquity of IoT cameras such as dash mounted cameras, drone cameras, Google glass, and phone cameras, has enabled many useful applications such as law enforcement [29], traffic monitoring [25] etc. One potential application is related to finding interested targets (e.g., crime suspect, lost child, lost vehicle, etc.), where videos can be crowdsourced on demand from the public to help search for targets in real time. Currently, police use surveillance cameras for target search. However, surveillance cameras often have limited coverage due to deployment cost. On-demand crowdsourcing can significantly enhance the effectiveness of target search. For example, after a crime or a terrorist event, police can create a crowdsourcing task to search for the suspects/targets with information such as photos of the targets or other biometric information. The on-demand crowdsourcing system assigns mobile users (workers) to search for targets. Then workers enable their IoT cameras to capture their surrounding area, upload some photos or video clips to the server, and in return receive payments. Based on the crowdsourced videos or photos, the police can find the targets' locations or moving trajectories, etc.

However, such system may raise privacy concerns. First, to locate the target, workers may be provided with photos of the target, and then crowdsourced videos can be processed locally on the workers' device using content-based techniques [33] to filter out parts of the videos that do not contain the target. Although this design works well if the target is a lost child, a wanted criminal or terrorist, it may raise privacy concerns in many other cases. For example, during police investigation, the identity of the targets should not be released. Hence, providing workers with photo of the target may create privacy issues and may even compromise the investigation. Second, IoT cameras capture videos/photos in public places which can be misused to track bystanders. Such tracking can be a major concern for unwilling bystanders who feel that their privacy is being violated.

Although the problem of preserving the privacy of bystanders has been studied for surveillance cameras [24, 30, 32], most of

them have been proposed for fixed infrastructure where the privacy preserving algorithm is run on powerful servers to detect bystanders and hide their faces. Some other researchers [8, 39] consider content-based modification of captured images or video frames, guided by a privacy policy to preserve individual privacy preferences. However, they assume that each capturing device is associated with a trusted cloudlet located close to that device to run computation intensive tasks, which may not always be possible especially for users with strong privacy requirements. Moreover, none of them considers the problem of target search, and do not have the requirement of preserving the privacy of targets as well as the privacy of bystanders. Preserving the privacy of the target is also hard, since the requester (e.g., the police) may not even want to release the photo of the target to the server, which is honest but curious and has the incentive to reveal the target or bystanders information in order to infer sensitive information about them. As a result, offloading the processing of captured images or video frames, which may contain sensitive information, to the server may not work well. Then, how to process these images or video frames on resource limited mobile devices in a privacy preserving manner becomes a challenge.

In this paper, we design and implement TargetFinder, a privacy preserving system for target search through IoT cameras. In TargetFinder, we propose efficient image processing techniques that can run on mobile devices, which allow the task requester (e.g., police) and the workers to process images and videos locally on their mobile devices to extract target's face information and bystanders' face information. To preserve the privacy of the target and the bystanders, we propose to encrypt the target's face information and the bystander's face information before sending them to the server. By exploiting homomorphic encryption techniques [10], the server can search for the target on encrypted information. After matching the target face to the detected faces, oblivious transfer [12] is applied to allow a requester to receive images that include the target, while all other captured images of the bystanders are not revealed. Moreover, the target's face image is not revealed to the server and the participating workers. Due to the high computation overhead of the cryptographic primitives, we develop optimization techniques in order to run our privacy preserving protocol on mobile devices.

In summary, the contributions of this paper are as follows.

- We design and build an on-demand target search system called TargetFinder, which uses videos crowdsourced from IoT cameras to help search for the target.
- We propose a privacy preserving protocol for target search, which preserves the privacy of the target and the bystanders. To speed up the protocol running time, several optimization techniques are designed and evaluated.
- We have implemented a prototype of TargetFinder, and demonstrate its effectiveness based on real experiments.

The rest of the paper is organized as follows. In Section 2, we present the system overview. The system design and implementation are described in Section 3. In Section 4, we evaluate the system performance. Section 5 describes related work and Section 6 concludes the paper.

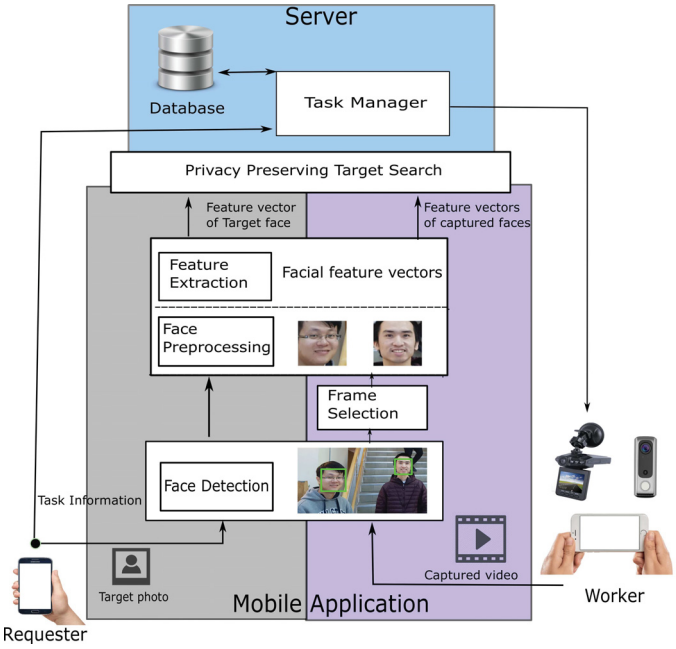


Figure 1: The TargetFinder system.

2 SYSTEM OVERVIEW

In this section, we first present the system overview and then introduce the threat model. Finally, we describe the basic idea of our privacy preserving protocol.

2.1 TargetFinder Overview

TargetFinder helps locate targets using corwdsourced videos from IoT cameras such as vehicle mounted cameras, Google glass, or phone cameras. The videos can be processed at the IoT camera or at the worker's mobile device such as smartphones. As shown in Figure 1, TargetFinder consists of a server and a mobile app that runs on the requesters' and workers' mobile devices. A requester can be someone who wants to pay for getting information about the target. The requester creates a task about the target which includes some information such as the task description, geographical information such as possible locations of the target, and the photo of the target. If privacy is not a concern, the photo of the target will be shown to the workers. In this paper, we focus on tasks with privacy concerns, and thus the photo will not be shown to the workers. The photo of the target will be used by the privacy preserving algorithm so that only matched video frames are sent to the requester to protect the privacy of the target and the bystanders.

With the photo of the target, *Face Detection*, *Face Preprocessing*, and *Feature Extraction* are executed to detect the target's face and extract its feature vector. On the worker side, after videos are captured, they are processed to select some video frames, referred to as captured photos. Based on these captured photos, *Face Preprocessing*, and *Feature Extraction* are executed to process faces, and extract their feature vectors. Then the proposed *Privacy Preserving* protocol is used for target search which preserves the privacy of the target and the bystanders.

On the server side, each task has a *Task Manager* which stores the task information, tracks the progress of the task, selects participant workers, and sends notifications to the selected workers. The *Task Manager* notifies workers by pushing the task notification to their mobile devices. This is implemented by using the Firebase Cloud Messaging (FCM) service [6] provided by Google.

2.2 Threat Model

TargetFinder preserves the privacy of the target and bystanders by considering the following threat model. The server is honest but curious who has the incentive to reveal the target or bystanders information in order to infer sensitive information about them. The requesters and workers are assumed to follow the privacy preserving protocol, but may try to learn extra information from the received information. We assume that the requesters and the server do not collude.

TargetFinder addresses the privacy concerns of the target and the bystanders by employing cryptographic primitives to privately match the target face to the captured faces. The requester only receives the matched photos that include the target, not other photos. Moreover, the target information is not revealed to the server and the participating workers such that only the requester knows the identity of the target, and can discover his location.

2.3 Basic Idea of the Privacy Preserving Solution

To search for the target in the captured images while preserving the privacy of both the target and the bystanders, we propose the following privacy preserving protocol. First, the requester generates a public and private key pair, and creates a task to search for the target with the photo of the target. The feature vector of the target is extracted, and encrypted using the requester's public key. Then, he sends the public key and the encrypted feature vector to the server. The server selects and notifies participating workers, and then broadcasts the public key to all selected workers. On the worker side, images are processed, and feature vectors are extracted. The extracted feature vectors are encrypted using the public key, and sent back to the server. For each detected face, the server computes the encrypted dot product of the following two vectors: the encrypted feature vector of the detected face and that of the target. The encrypted dot products are returned to the requester. Then, the requester decrypts and selects photos that include the target based on a predetermined threshold. Finally, the requester can initiate the oblivious transfer protocol to retrieve the selected photos from the server.

To compute the dot product on encrypted feature vectors, it is required to do additions and multiplications over ciphertexts. While partially homomorphic encryption is in general more efficient than Somewhat-Homomorphic encryption (SHE) [10], it only allows additions or multiplications on encrypted data, but not both at the same time which is required for computing the dot product. Thus, we use SHE based on ideal lattices [10] which allow the server to compute the dot product on ciphertexts produced by the requester and the workers. However, it has high computation overhead as it does additions and multiplications over ciphertext represented by ring of polynomials, which will be addressed in this paper.

3 TARGETFINDER DESIGN AND IMPLEMENTATION

In this section, we describe the design and implementation of TargetFinder's two major components: image processing and privacy preserving target search. Image processing will be used in our privacy preserving protocol, and hence we first describe it.

3.1 Image Processing

In TargetFinder, face detection is executed first to detect faces in every video frame. Then, frames with the most number of detected faces are selected, and finally features are extracted from these frames. The following describes the details of these techniques: face detection, frame selection, face preprocessing, and feature extraction.

Face Detection: The first step is to detect faces in images or video frames. There are several face detection libraries available on Android. For example, the Android SDK includes a face detection API. However, it is not accurate and may miss the detection of many faces. In TargetFinder, we use the face detection algorithm called Haar feature-based cascade classifier from the OpenCV library [7]. It has much higher accuracy compared to the Android face detection API, and allows flexible configurations with various parameters, such as image scaling, minimum size of a face, etc..

Frame Selection: Extracting feature vectors of all detected faces from all video frames takes a long time and enormous computational power. Moreover, there are usually minor changes between consecutive video frames and the target will appear in many consecutive frames, and hence sampling video frames at a lower rate works well on capturing the targets. Therefore, to remove the redundancy, we select only one frame every second, and the frame with the most number of detected face is selected. Based on the computation capability of the IoT camera, more frames may be selected. It is possible that frames can be better selected based on other factors such as the face image quality. Since the focus of this paper is the privacy preserving protocol, we will leave it as future work.

Face preprocessing: For the selected frames, the faces are preprocessed before extracting feature vectors. Preprocessing is needed because the detected faces may have various sizes, positions, and orientations which affect the results of the feature extraction algorithm. In preprocessing, image grayscale conversion, resizing, and face alignment are performed. The face alignment aligns the landmarks of the face (e.g. eyes, nose and mouth) to appear at roughly the same position in the final output images. In TargetFinder, we use Dlib [5] to detect the face landmarks, and transform the landmarks to the positions specified by the feature extraction algorithm.

Feature Extraction: There are several feature extraction algorithms, such as Eigenfaces [38], and Light convolution neural network (CNN) [40]. Eigenfaces represents a human face with a set of eigen vectors, while Light CNN represents a human face using a 256 dimensional feature vector extracted from a 29-layer CNN. Light CNN has several advantages over other algorithms. First, it can achieve a much higher face verification accuracy (i.e., 99%). Moreover, the size of the Light CNN model is only 33MB, much smaller than many other CNN models such as the VGG Network [34] which is about 580MB. This makes it more suitable to be

run on smartphones. Thus, we use Light CNN for feature extraction, and the Android ported Caffe deep learning framework [3, 16] is used in our implementation to run Light CNN.

After extracting the feature vector of the target face and those of the captured faces, the feature vector of target face is compared with that of each detected face. The comparison is by computing the dot product of the two vectors. If the dot product is above a predetermined threshold, the detected face matches the target face. In our system, such comparison is done privately as described in Section 3.2 to preserve the privacy of the targets and the detected faces.

Performance Optimization: The extracted feature vectors are used in the privacy preserving protocol. Such protocol is based on cryptographic primitives which are computationally intensive, and the running time depends on the dimension of the feature vectors. To reduce the running time, we reduce the dimension of the 256-dimensional feature vectors using principal component analysis which linearly projects high dimensional data into a lower dimensional space. The projection is done by multiplying original feature vectors by a transformation matrix to transform the feature vectors into b-dimensional space. To obtain the transformation matrix, we first extract facial feature vectors, and then perform singular value decomposition on the extracted vectors. The transformation matrix is represented by the b most significant components. With such dimensionality reduction, we can speed up the execution of the cryptographic protocol with the cost of face recognition accuracy. We study such tradeoff in the performance evaluations (Section 4) to choose a reasonable b-dimensional space.

To improve the performance of image processing, we use multi-thread to process multiple frames simultaneously. As the JNI code of the Caffe framework is not thread-safe, we modify the code to allow each thread to hold its own instance of the neural networks for feature extraction. As the feature extraction is the most computational intensive task, we use one thread to run face detection, one thread for frame selection, and multiple threads to preprocess faces and run multiple instances of Light CNN to extract feature vectors in parallel.

3.2 Privacy Preserving Target Search

In this section, we first describe the cryptographic primitives and then introduce our privacy preserving protocol for target search.

3.2.1 Cryptographic Primitives

Somewhat-Homomorphic encryption (SHE): SHE allows additions and multiplications over ciphertexts, while partially homomorphic encryption allows only addition or multiplication over ciphertexts. Thus, by using SHE, the server can compute the vector dot product on ciphertexts produced by the requester and the workers. In TargetFinder, we use the SHE scheme based on ideal lattices described in [10]. In the following, we describe its basic idea.

The plaintext m resides in the ring of polynomials with coefficients in $\mathbb{Z}_t$, i.e., $m \in R_t = \mathbb{Z}_t[x]/(x^d + 1)$, and the ciphertext space is in $R_p = \mathbb{Z}_p[x]/(x^d + 1)$. The parameters p and t are positive integers, where $p > t > 1$, and they define the upperbound of the ciphertext and plaintext coefficients respectively. Let χ be the error distribution in R_p, then the private key (sk) is obtained by sampling $s \leftarrow \chi$; i.e., $sk = s$. To generate the public key, we first sample uniformly at random $a_1 \leftarrow R_p$, and sample small error $e \leftarrow \chi$. Then the public key is $\mathbf{pk} = [a_0 = -(a_1 s + te), a_1]$. Given this public key and a message m, the encryption is performed by sampling $u, g, h \leftarrow \chi$ and outputs the following ciphertext: $\mathbf{c} = [c_0, c_1] = [a_0 u + tg + m, a_1 u + th]$.

SHE allows additions and multiplications on ciphertexts. Given two ciphertexts $\mathbf{c} = [c_0, c_1]$ and $\mathbf{c'} = [c'_0, c'_1]$, we can add them to obtain a ciphertext $\mathbf{c_{add}} = [c_0 + c'_0, c_1 + c'_1]$. Similarly, the product of the two ciphertexts is $\mathbf{c_{mlt}} = [c_0 c'_0, c_0 c'_1 + c'_0 c_1, c_1 c'_1]$. The homomorphic multiplication increases the number of ring elements in the ciphertext. Hence, the encrypted vector dot product returned to requester includes one extra polynomial, i.e., $\mathbf{c_{mlt}} = [c_0, c_1, c_2]$, and the decryption can be computed using the private key sk as follows: $m = (c_0 + c_1 s + c_2 s^2)\ mod\ t$.

k-out-of-n Oblivious Transfer: The k-out-of-n oblivious transfer protocol, denoted OT_n^k, is a two party protocol that allows a requester to privately choose k out of n images from the server. The requester learns k images and nothing else, and the server learns nothing about the chosen k images. In our implementation, we use the OT_n^k scheme [12] which is as follows. A requester constructs a k-degree polynomial function with roots equal to the selected indices, and then hides this polynomial in another random k-degree polynomial function. The masked polynomial coefficients are sent to the server. Because of the random polynomial function, the server cannot learn about the selected indices from the masked polynomial coefficients, and it can only use such coefficients to encrypt the images and send them back to the requester. Finally, the requester can only decrypt images of the selected indices. All other $n - k$ images are unknown to the requester and cannot be decrypted.

Table 1: Table of notations

Parameter	Description
pk	SHE public key
sk	SHE private key
d	Degree of the polynomial
z	Number of selected workers
$\mathbf{f^t}$	Feature vector of the target
F^{w_i}	Extracted feature vectors of worker w_i
q	Dimension of the feature vector
n	Number of images captured by selected workers
k	Number of images selected by the requester

3.2.2 Privacy Preserving Target Search

Our privacy preserving protocol works as follows. First, the requester generates a public and private key pair, and creates a task to search for the target with the photo of the target. The feature vector of the target is extracted, and encrypted using the requester's public key. Then, he sends the public key and the encrypted feature vector to the server. The server selects and notifies participating workers, and then broadcasts the public key to all selected workers. On the worker side, after selecting video frames, the captured photos (frames) are processed and feature vectors are extracted. The extracted feature vectors are encrypted using the public key,

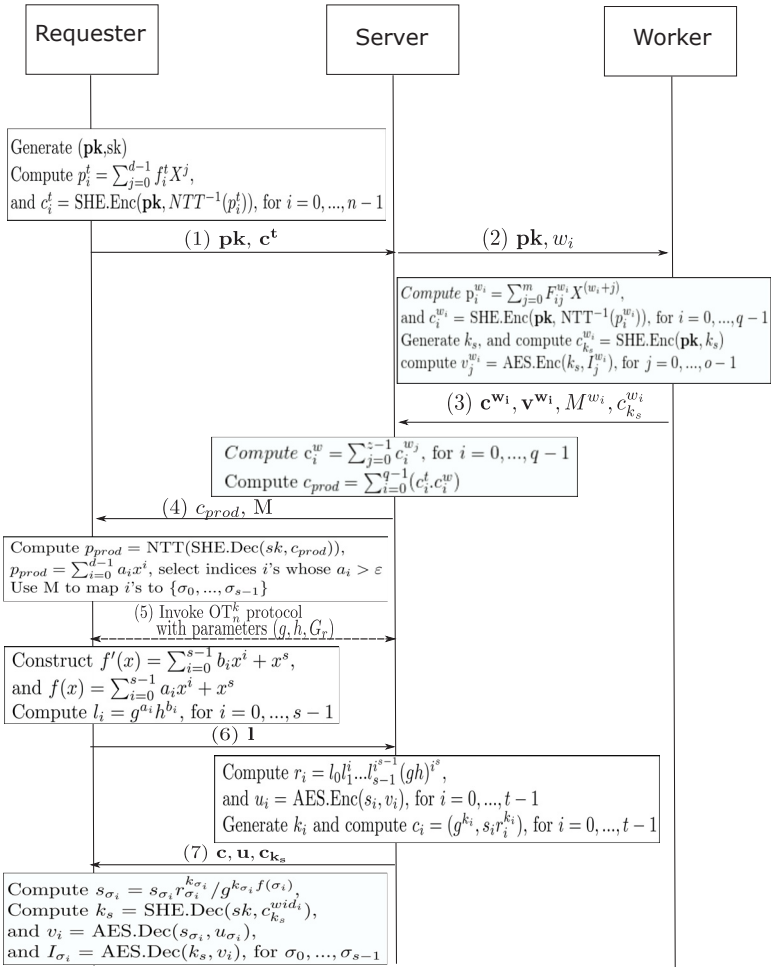


Figure 2: Our privacy preserving protocol

and sent back to the server. For each detected face, the server computes the encrypted dot product of the following two vectors: the encrypted feature vector of the detected face and that of the target. The encrypted dot products are returned to the requester. Then, the requester decrypts and selects photos that include the target based on a predetermined threshold. In case the target is found in some photos, the requester initiates the oblivious transfer protocol to retrieve these photos from the server. The used parameters are summarized in Table 1.

Baseline protocol: A straightforward implementation of this protocol incurs high computational overhead and high bandwidth consumption for several reasons. First, a worker may capture multiple faces, and hence needs to encrypt several feature vectors. To encrypt m feature vectors with q dimensions, mq encryptions are needed to generate mq encrypted polynomials. Second, the server needs to compute encrypted dot products of the feature vectors. For z selected workers, each with m feature vectors, zm dot product calculations are required, and zm encrypted dot products, represented by a polynomial of d coefficients, are returned to the requester. Then, the requester needs to decrypt zm encrypted dot products. This results in high computational overhead at the server and the requester, and high bandwidth consumption between the requester and the server.

Optimized protocol: To reduce the high computational overhead of the baseline protocol, we propose the following optimizations. First, for a worker w_i, the number of encryptions can be reduced from mq to q as follows. Let matrix F^{w_i} store the m feature vectors, each with q dimensions. To encrypt the feature vectors, we first generate a polynomial from each row of the matrix, to have q polynomials, each storing m feature values. Then, the requester's public key is used to encrypt the q polynomials. Thus, we only have q encryptions.

Second, at the server, we pack the ciphertext to compute the dot product in parallel for all zm feature vectors, hence reducing the computational overhead. After computing the dot products, only a single encrypted polynomial is returned to the requester and hence reducing the bandwidth consumption. The requester decrypts it to get a polynomial of degree d, where each coefficient represents a dot product value. More specifically, the ciphertext is packed into q polynomials. For a polynomial, each coefficient represents an

element of different feature vector. Thus, to independently compute the dot product of each feature vector with that of the target, we need coefficient wise homomorphic operations. Polynomial additions are naturally coefficient wise, but polynomial multiplications result in convolution product of the coefficients. To transform convolution products into coefficient wise products on rings, we use the Number Theoretic Transform (NTT) which is a discrete Fourier transform specifically for finite rings. To transform convolution products on the encrypted polynomials into coefficient wise products on the plaintext polynomials, we apply an inverse-NTT to plaintexts before encryption, and then an NTT to the plaintext after decryption. These transformations do not affect polynomial additions as they are linear transformations.

Third, encrypting/decrypting captured images with SHE involves computations of many polynomials, and hence inefficient. To address this problem, we use a hybrid approach, where images are encrypted and decrypted using AES symmetric-key, and only the short AES key is encrypted and decrypted using SHE.

Figure 2 shows the protocol, which is described as follows.

(1) The requester generates a SHE public and private key pair, denoted as **pk** and sk. Then, she extracts the feature vector of the target with q dimensions, denoted by $\mathbf{f^t}$. Each element of the feature vector is represented by a polynomial of degree d as follows. For each f_i^t, we have $p_i^t = \sum_{j=0}^{d-1} f_i^t X^j$. Inverse-NTT ($\text{NTT}^{-1}$) is applied to the polynomials and result is encrypted with **pk**, i.e., $c_i^t = \text{SHE.Enc}(\mathbf{pk}, \text{NTT}^{-1}(p_i^t))$, and $\mathbf{c^t} = [c_0^t, ..., c_{q-1}^t]$. Then, **pk**, and $\mathbf{c^t}$ are sent to the server.
(2) After notifying the participant workers, the server broadcasts the public key **pk** and a random index w_i to each participating worker.
(3) Worker w_i extracts the matrix F^{w_i} which represents the feature vectors of detected faces in the captured photos. Each row of the matrix F^{w_i} is represented by a polynomial, i.e., $p_i^{w_i} = \sum_{j=0}^{m} F_{ij}^{w_i} X^{w_i+j}$. Similar to the requester, the inverse-NTT is applied to the polynomials and the result is encrypted with **pk**; i.e., $c_i^{w_i} = \text{SHE.Enc}(\mathbf{pk}, \text{NTT}^{-1}(p_i^{w_i}))$. In addition, she encrypts each captured image $I_j{}^{w_i}$ using AES symmetric key k_s as follows: $v_j^{w_i} = \text{AES.Enc}(k_s, I_j^{w_i})$, and encrypts k_s using **pk**, i.e. $c_{k_s}^{w_i} = \text{SHE.Enc}(\mathbf{pk}, k_s)$. Then she sends to the server the following: $\mathbf{c^{w_i}}$, $\mathbf{v^{w_i}}$, $c_{k_s}^{w_i}$, and a matrix that maps each image to its extracted feature vectors denoted by M.
(4) For each dimension i of the q dimensions of the feature vectors, the server packs all z workers' ciphertexts by adding them, i.e. $c_i^w = \sum_{j=0}^{z-1} c_i^{w_j}$. Then it computes all encrypted dot products in parallel as follows: $c_{prod} = \sum_{i=0}^{q-1} c_i^t . c_i^w$, and sends c_{prod} and M to the requester.
(5) The requester decrypts c_{prod} and applies NTT to get a polynomial with a dot product for each coefficient, i.e., $p_{prod} = \text{NTT}(\text{SHE.Dec}(sk, c_{prod}))$. The feature vectors with dot product value larger than a threshold ϵ, matches that of the target, and will be selected. Then she uses M to find the indices of the images containing these feature vectors, denoted as $\{\sigma_0, ..., \sigma_{k-1}\}$. To securely retrieve these k images, she initiates oblivious transfer protocol OT_n^k with the server with system parameters (g, h, G_r).
(6) The requester constructs a polynomial function of degree k: $f'(x) = \sum_{i=0}^{k-1} b_i x^i + x^k$, where $f'(i) = 0$, iff $i \in \{\sigma_0, ..., \sigma_{k-1}\}$. Then she generates a random polynomial $f(x) = \sum_{i=0}^{k-1} a_i x^i + x^k$ which is used to compute $\mathbf{l} = [l_0, ..., l_{k-1}]$, where $l_i = g^{a_i} h^{b_i}$. Then she sends $\mathbf{l}$ to the server.
(7) For each image i, the serve computes $r_i = l_0 l_1^i ... l_{k-1}^{i^{k-1}} (gh)^{i^k}$, generates a secret key k_i, and encrypts the image using a hybrid approach. The hybrid approach first encrypts the image using AES encryption key s_i, i.e., $u_i = \text{AES.Enc}(s_i, v_i)$, and encrypts s_i using the public key, i.e., $c_i = (g^{k_i}, s_i r_i^{k_i})$. The server sends the following to the requester: the encrypted images $\mathbf{u}$, the encrypted symmetric keys $\mathbf{c}$ and the encrypted symmetric keys c_{k_s}.

Finally, for every index in $\{\sigma_0, ..., \sigma_{k-1}\}$, the requester can get the symmetric key s_{σ_i} by computing $s_{\sigma_i} = s_{\sigma_i} r_{\sigma_i}^{k_{\sigma_i}} / g^{k_{\sigma_i} f(\sigma_i)}$. The selected images u_{σ_i} can be decrypted with the symmetric keys s_{σ_i}, $c_{k_s}^{w_i}$, and the SHE private key sk.

3.2.3 Privacy Analysis

In this section, we present an analysis of our privacy preserving protocol in order to show its effectiveness in preserving the privacy of the target and the bystanders.

The SHE scheme relies on the hardness of the Ring Learning with Errors problem [23], and hence provides strong security guarantee. Thus, the server with a SHE public key can not learn the feature vectors of the target and bystanders from the ciphertext. In addition, the SHE scheme provides semantic security which means that it is not computationally possible to know if two different ciphertexts are obtained from the same plaintext. Therefore, the server cannot learn extra information from the requester and workers' encrypted feature vectors. Moreover, the server cannot infer target location, although it knows workers' locations where captured photos are taken. This is because the server does not know the dot product values, i.e. the server does not know which photos may include the target. If it knows such information, it can infer which worker captures the target, and hence can know the target location.TargetFinder was presented under the assumption that the sever is honest-but-curious, however, with some minor modifications, it can be resistant to some active attacks performed by the server. In the case where server is malicious, it can perform man-in-the-middle attack in order to obtain the feature vectors of bystanders. The server can perform the attack by constructing SHE public and private keys, and provides the constructed public key instead of requester's SHE public key to workers. In this case, server can decrypt workers' encrypted feature vectors. However, this attack can be detected by allowing workers to communicate to requester to check that requester's SHE public key was not altered.

In our system, a requester with SHE public and private key can decrypt any SHE ciphertext. The requester is given only encrypted dot product results, and hence cannot use it to infer feature vectors of bystanders. Moreover, using oblivious transfer protocol, the requester can learn only the matched photos, and not any other photos. In the case where requester is malicious, he can manipulate the photo selection threshold. By lowering the threshold, he can allow the selection of many photos that may not contain the target,

and hence breach the bystanders privacy. However, we assume requesters follow the protocol correctly.

Finally, in TargetFinder, workers do not receive any target information, and they do not know whether the captured photos include the target as the requested photos by the requester are unknown to them. Hence, they cannot learn the identity of the target and cannot discover his location.

3.2.4 Implementation Issues

In order to implement SHE on smartphone, we port the Java Lattice Based Cryptography (jLBC) [1] library to Android. However, the jLBC library runs very slow on Android which severely affects the performance. SHE requires polynomial additions and multiplications. In jLBC, the addition of two polynomials of degree d can be done in $O(d)$ time, but the multiplication of two polynomials takes $O(d^2)$ time, which is too long.

We propose the following optimizations to speed up the SHE implementation. First, we speed up polynomial multiplications by implementing NTT to reduce the time complexity from $O(d^2)$ to $O(d \log d)$. Although the NTT algorithm can be found in [11, 27], we implement it in Java and port it to Android. Second, we use the `BigInteger` implementation from OpenJDK 9.0, which has a major improvement on the speed of multiplication of `BigInteger` numbers. To further speed up SHE, we use multiple threads to encrypt multiple polynomials simultaneously.

To implement the oblivious transfer protocol on smartphone, we compute the coefficients of the polynomial with roots equal to the selected indices using the Efficient Java Matrix Library (EJML), and use the Bouncy Castle [2] cryptographic library for the AES implementation.

On the server side, we implement SHE in C++ and use the NTT-based Fast Lattice library (NFLlib) [9]. We implement the oblivious transfer protocol and use the MPI library provided by libgcrypt [4] for both big integer and prime number operations. We use the AES implementation in libgcrypt.

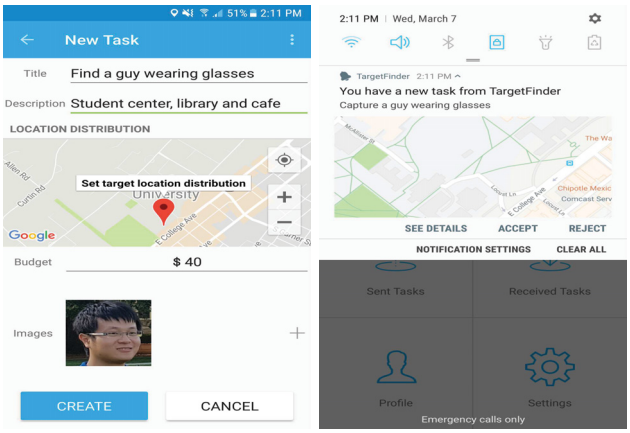
4 PERFORMANCE EVALUATIONS

In this section, we use a real world demo to show how TargetFinder works, and evaluate the performance of the two major components: image processing and privacy preserving target search.

4.1 Experimental Setup

We implement the TargetFinder mobile app on Android, and develop a Sinatra application on the server. The server is run on a desktop with an Intel i7 processor and 32 GB RAM, and the mobile app is deployed on LG Nexus 5X and Samsung Galaxy S8 smartphones. Figure 3 shows the user interface of TargetFinder.

The following parameters are used in our privacy preserving protocol. For SHE, parameters are chosen to ensure at least 128-bit security. The polynomials are of degree $d = 4096$, with coefficients of size $\log_2 p = 124$ bits, and the plaintext feature elements are of size $\log_2 t = 23$ bits. For the oblivious transfer protocol, G_r is set to be the group of quadratic residues modulo a safe prime r, and all operations are modulo m. r and m are set to 1024 bit. For the AES encryption, we use a symmetric key of size 128 bit.



(a) Task Creation (b) Task Notification

Figure 3: TargetFinder User Interface.

Table 2: Details of the datasets

Dataset name	Number of tasks	Number of images	Category	Number of faces
Campus	8	447	With target	71
			Without target	479
			Total	550
Mall	8	460	With target	67
			Without target	876
			Total	943
Downtown	8	492	With target	74
			Without target	1171
			Total	1245
All	24	1399	With target	212
			Without target	2526
			Total	2738

We recruited 11 graduate students and 3 undergraduates to use our TargetFinder mobile app. Students use our mobile app to take videos/images when they stand still or sit in any public place, and there is no significant difference from normal smartphone users. In many cases, the bystanders do not know whether the workers are taking videos/images, or just reading news. Three datasets were collected based on the following three location areas: university campus, shopping mall and downtown. At each location area, eight tasks are created. For each task, we record all interactions between the workers and the TargetFinder app. When a worker starts recording a video, *Frame Selection* is used to select the frame with the highest number of detected faces. When the video recording finishes, all selected video frames (with resolution of 1920×1080) are uploaded to the server. We manually label the faces into two categories: targets' faces and irrelevant faces. Those labeled faces are used as the ground-truth for our experiments. The details of the three datasets are provided in Table 2.

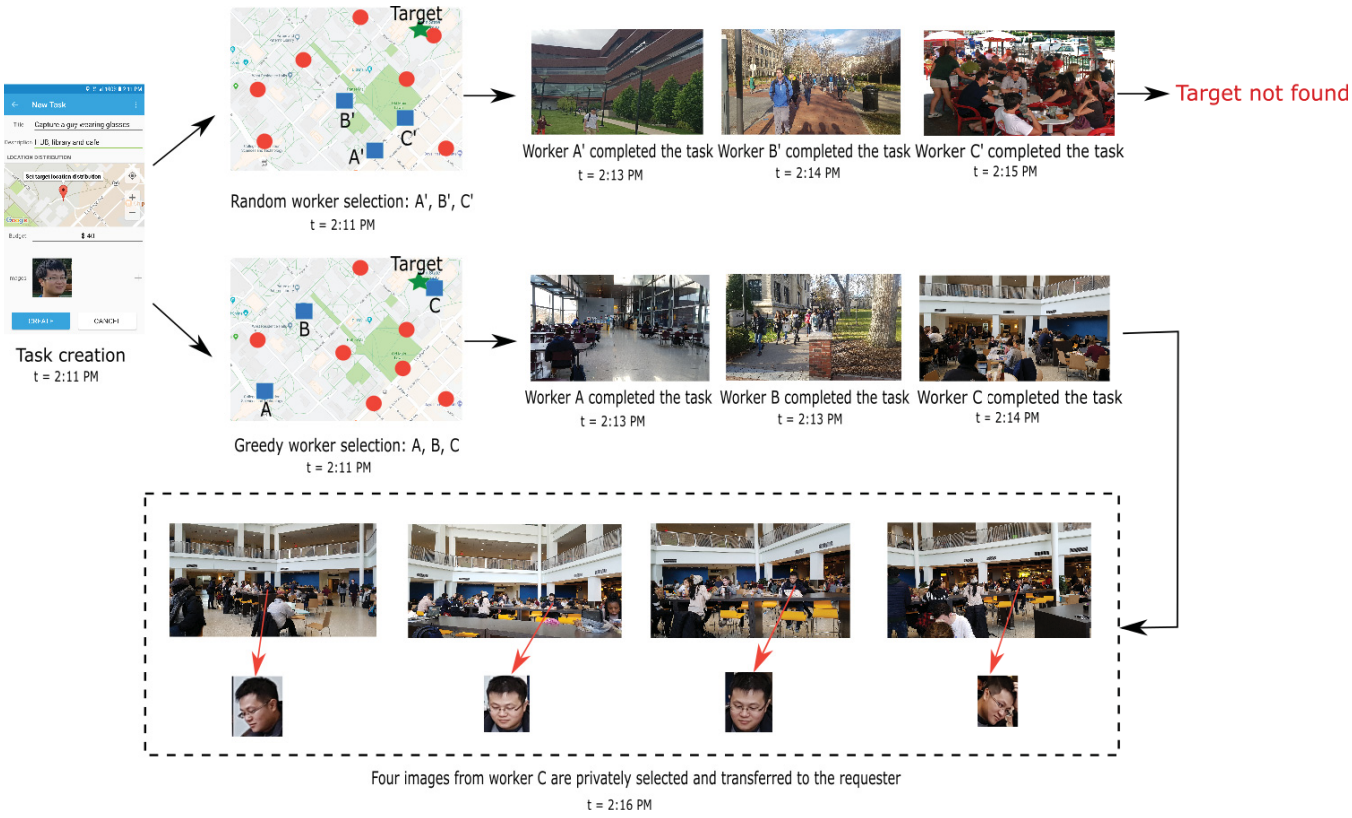


Figure 4: A real world demo

4.2 Worker Selection

In TargetFinder, the selection of workers to perform the task is a challenge. There may be a large number of candidate workers around the searching area and asking all of them to perform the task has high cost. Hence, we design a worker selection scheme that selects a subset of workers to complete the task.

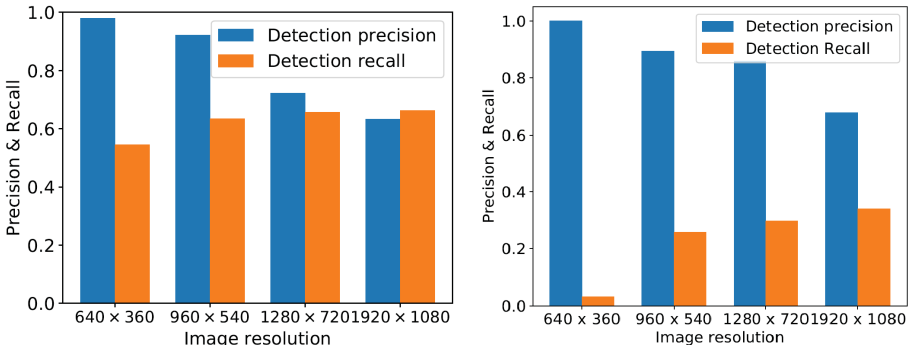
TargetFinder asks a requester to provide some information about the target such as possible target locations and the probability of the target being at each location. Such information can be obtained based on application knowledge. For example, in law enforcement, the police know the possible locations that the suspect will appear. In our system, a worker covers a square area centered at his location, where points closer to the center have higher probability of being covered, i.e., less likely to be occluded. Thus, the worker coverage and target location are represented by a discrete probability distributions. In TargetFinder, each task has a budget which is the maximum payment that a requester is willing to pay for the task, and each worker receives a payment after executing the task. The described worker selection problem can be easily proved to be NP-hard. Specifically, the 0-1 Knapsack problem can be reduced to the worker selection problem.

Hence, we propose a greedy algorithm to solve the following optimization problem: how to select a limited number of workers to maximize the probability of finding the target. Initially, we consider all workers that are located within the target area. Workers are ordered by their probability of finding the target, and the worker with the highest probability is selected. Then all remaining workers are reconsidered to select the next worker. Specifically, we consider each candidate worker along with the already selected worker to calculate their total probability of finding the target, and the one with the highest total probability is selected. It is important to jointly consider the remaining workers and the already selected worker because their coverage can overlap. The selection process continues until the payment needed for the selected workers exceeds the total payment, or no more worker is left.

4.3 A Real-World Demo

Figure 4 illustrates how TargetFinder works. A task is created by a requester who wants to search for her friend on the university campus, and she is willing to pay at most $40 for this task. She has a photo of her friend, and she also knows that he may be located in one of the following locations on campus: student center, library, and cafe. By default, her friend's photo is considered as sensitive information that should be protected. To help search for the target, she provides extra information about the target; i.e., a guy wearing glasses.

After the server receives the task, it selects three workers: A, B, and C, based on our worker selection algorithm. As shown the Figure 4, Worker A is located in the cafe, Worker B is located in front of the library, and Worker C is inside the student center.



(a) OpenCV (b) Android SDK

Figure 5: Precision and recall of different face detection approaches on different image resolutions

After selecting these workers, the server notifies them by pushing a notification to their mobile devices.

After receiving notifications, the workers accept the task, open the camera interface provided by TargetFinder mobile app, and starts capturing videos of their surrounding area. Workers A and B completed the task at 2 : 13 PM, while Worker C completed the task at 2 : 14 PM. Then, the privacy preserving target search protocol is run to search the target from the captured photos (selected video frames). Four matched photos are found that include the target face. These matched photos are received by the requester at 2 : 16 PM. Note that the workers and the server cannot see these photos since they are encrypted, and the requester can only see these four matched photos, not other captured photos.

As a comparison, Figure 4 also shows that the random worker selection algorithm is used to select three workers: A', B', and C'. As can be seen, these workers are not close to the target. Therefore, their captured images do not include the target and they can not find the target.

4.4 Performance of Image Processing

In this subsection, we evaluate the performance of image processing, which has three main components: face detection, face preprocessing, and feature extraction.

As shown in Table 3, to process a single image with one face on a Nexus phone, the average running time is 0.486 + 0.172 + 0.826 = 1.484s.

Table 3: Running time of image processing

	Detection (ms)	Preprocessing (ms)	Feature Extraction (ms)
Nexus	486	172	826
S8	92.4	137	440

4.4.1 Face detection. Table 4 shows how the performance of the face detection can be affected by different image resolutions. In order to get the measurements, we scale the images of the datasets into different resolutions, and measure the average running time on images with different resolutions. As can be shown, when the image resolution increases, the running time increases.

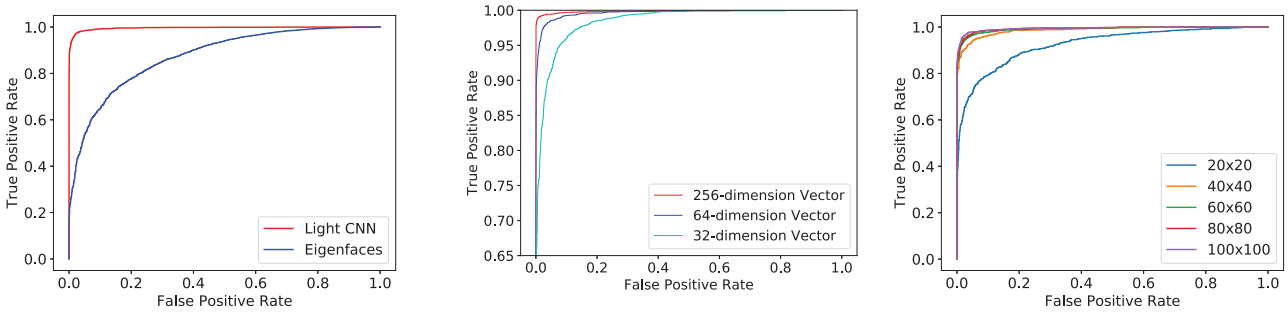
Table 4: Running time of face detection on images with different resolutions

Image resolutions	Nexus (ms)	S8 (ms)
640 × 360	331	16.7
960 × 540	354	28.7
1280 × 720	370	48
1920 × 1080	486	92.4

Figure 5 shows the precision and recall of different face detection approaches on our dataset, where Precision = true positives/(true positives + false positives), and Recall = true positives/(true positives + false negatives). Although both precision and recall are important, it is hard to achieve both. For our application, missing a target (false negatives) is much worse, and then recall is more important. However, the recall of Android SDK is too low compared to that of openCV, thus we choose openCV in TargetFinder. As shown in the figure, as the image resolution increases, more faces can be detected, but the number of false positives increases faster than the number of true positives, and hence the precision decreases and the recall increases.

4.4.2 Feature extraction. Figure 6a compares the performance of Light CNN and Eignfaces. In the comparison, both approaches are used to extract feature vector of the target face and that of the captured faces. The extracted feature vectors are used to run face recognition to identify the target face in the captured faces. As shown in Figure 6a, Light CNN significantly outperforms Eignfaces. With true positive of 95%, the false positive of Light CNN is only 0.02%, but it is about 70.0% for Eigenfaces. Thus, we choose Light CNN in TargetFinder.

Figure 6b illustrates the effect of feature vector dimension on the performance of face recognition. when the true positive rate is set to be 0.99, the false positive rate for 64-dimension vectors and 32-dimension vectors are 0.08 and 0.25 respectively. Thus, we choose 64-dimension feature vector in TargetFinder.



(a) Light CNN and Eigenfaces. (b) Feature vectors with different dimension (c) Light CNN with different face image size

Figure 6: The performance of face recognition.

Table 5: Performance of SHE (per feature vector)

Protocol	Phone	Key generation (ms)	Encryption (ms)	Decryption (ms)	NTT (ms)	NTT^{-1} (ms)
Baseline	Nexus	6995	106960	2250	180	183
	S8	315.7	36416	650	84	96
Optimized	Nexus	6995	2139.2	2241	180	183
	S8	315.7	728.32	642	84	96

Table 6: Performance of oblivious transfer protocol

k	n	Requester				Server
		Nexus		S8		
		Setup (ms)	Dec. (ms)	Setup (ms)	Dec. (ms)	Enc. (ms)
2	100	9.9	85	6.5	50	560.7
10	100	28.2	213.5	11.7	82.71	637.7
10	500	28.2	213.5	11.7	82.71	3381
100	500	173	1226	70.78	260.63	53035.5

Table 7: Performance of AES

Image resolution	Smartphone				Server
	Nexus		S8		
	Enc. (ms)	Dec. (ms)	Enc. (ms)	Dec. (ms)	Enc. (ms)
640×360	38	43.9	37.1	39.3	1.75
960×540	81.3	85.9	70.8	73.6	3.78
1280×720	145.1	155.3	113.7	121.7	6.72
1920×1080	315.5	398.2	245.35	261	16.17

Figure 6c shows the impact of face image size on the ROC curve of face recognition. We select face images larger than 100×100, and downsample them into sizes of 100×100, 80×80, 60×60, 40×40 and 20×20. Then all face images are resized to 128×128, which is the input image size of the Light CNN. As shown in the figure, the accuracy of face recognition is similar when the image size is larger than or equal to 40×40. However, the accuracy significantly drops when the face image size changes from 40×40 to 20×20, for example, with the true positive rate of 90%, the false positive rate increases from 16.02% to 41.2%. Thus, the parameters in our face detection are tuned to only allow faces with image size equal to or larger than 40×40 to be detected.

4.5 Performance of the Privacy Preserving Target Search

In this section, we evaluate the computation and communication overhead of our privacy preserving protocol.

4.5.1 Computational Overhead. Table 5 shows the running time of the major operations of the privacy preserving protocol. As shown in the table, the optimized protocol outperforms the baseline protocol in encrypting and decrypting the feature vectors. For example, on S8, the optimized protocol encrypts the feature vectors with running time of 728.32 ms per feature vector, whereas the baseline protocol encrypts the feature vectors with running time of 36416 ms per feature vector. The optimized protocol is more efficient, because it packs the feature vectors into much less number of polynomials compared to the basic protocol.

Table 6 shows the computation time of the oblivious transfer protocol, where n is the number of images captured by the selected workers, and k is the number of images selected by the requester. The setup operation includes constructing and hiding the k-degree polynomial function. As k increases, the computation time increases, because the setup, encryption and decryption operations depend on k. We also observe that the computation time on S8 is less than that on Nexus, because S8 has higher processing and memory capabilities. Based on our data collection experiments, k is in the range of $2 - 10$ images. Thus, the computational time of our protocol on the server side may be up to 3381 ms, which is acceptable in our system.

Table 7 shows the computational time of AES on images with different resolutions. For different image resolutions, the computational time varies. By increasing the image resolution, the computational time increases. Compared to SHE, the computational time of AES is quite small which motivates our design choice of using AES scheme to encrypt and decrypt long images data.

Table 8: Communication Overhead of SHE (per 50 feature vectors)

Scheme	Requester		Worker	
	Upload (KB)	Download (KB)	Upload (KB)	Download (KB)
Baseline	8060	9316	396940	124
Optimized	8060	202	8076	124

4.5.2 Communication Overhead. Table 8 shows the communication overhead of SHE. In the optimized protocol, workers upload less amount of data compared to the baseline protocol, because workers in the optimized protocol pack feature vectors in less number of polynomials which reduces the communication overhead. In addition, the download data size is also reduced in the optimized protocol. In the baseline protocol, for z workers each with m feature vectors, requesters need to download all the zm encrypted dot products of size 9316KB. In the optimized protocol, requesters need to download only one polynomial of 202KB, which includes the dot products of all feature vectors.

Note that our optimized protocol is based on reducing the number of encryptions, packing the ciphertext, and using AES symmetric-key for encryption/decryption of images. As these optimization techniques do not include any approximations or changes on the original plaintext, it has the same accuracy as the baseline protocol.

5 RELATED WORK

Recently, researchers have built systems to crowdsource photos and videos from mobile devices such as Mediscope [18], GigaSight [33], SmartPhoto [42, 43], VideoMec [41], FlierMeet [14], and SecureFind [35]. With these mobile crowdsourcing systems, privacy concerns are becoming more and more important, and researchers have proposed privacy preserving techniques to mitigate such concerns [19, 20, 22, 37]. To *et al.* [37] presented a framework to protect the location privacy of the workers participating in crowdsourcing tasks. In [22], a privacy preserving scheme was proposed to hide workers' identities by ensuring that the server cannot link any submitted task to the reporting worker, or link multiple tasks submitted by the same worker. In [35], the authors proposed a privacy-preserving object-finding system via mobile crowdsourcing. In their system, a unique Bluetooth tag is attached to every valuable object such as personal asset, child, pet, or elderly, and it is used to help workers to find the object. However, their solution depends on special hardware, e.g., Bluetooth tags, which is not possible in our system since the suspect will not wear such special tags. Different from them, TargetFinder searches targets utilizing efficient image processing techniques that can run on resource constrained devices while preserving the privacy of the target and the bystanders.

Preserving the privacy of bystanders has been studied for surveillance cameras [13, 24, 30, 32]. In [30, 32], the authors proposed privacy preserving solutions to hide faces of bystanders, while allowing the detection of their actions and movements. In [24], the authors consider more general case of hiding region of interests which include privacy-sensitive information. However, most of these solutions are proposed for fixed infrastructure where the privacy preserving algorithm is run on powerful servers.

Researchers have considered content-based modification of captured images or video frames, guided by a privacy policy to preserve individual privacy preferences [8, 39]. In [8], the authors proposed I-Pic, a platform for privacy preserving image capture, which allows users to respect each others' individual privacy preferences, while capturing photos. In I-Pic, captured images are automatically modified according to the privacy choices of individuals captured. When a user chooses to hide his face, I-Pic modifies the captured image to blur that user's face before sharing the image. In [39], the authors proposes a mechanism for preserving the privacy of bystanders in video streams by selectively blurring faces according to specified policies. However, these works assume that each capturing device is associated with a trusted cloudlet to run computation intensive tasks, which may not always be available and many users may not trust these cloudlet. Hence, offloading the processing of captured images or video frames, which may contain sensitive information, to the server/cloudlet may not work well. Moreover, none of them considers the problem of target search, and do not have the requirement of preserving the privacy of targets and bystanders.

To enable efficient privacy preserving target search on mobile devices, optimized image processing techniques are explored. Face detection and face recognition have been studied for many years in the computer vision community [21, 44]. Most of the state of art algorithms [17, 26, 28, 31, 34] use CNN to improve the accuracy. However, CNN models usually stress the mobile devices by occupying a large amount of memory space and taking much longer time to execute. To address these issues, some researchers propose light weight solutions such as Light CNN[40], and binarized neural network (BNN) [15][36]. In BNN, the authors considered only binary numbers for the weights in the neural network, so that the running time and the memory requirement can be reduced, while the performance is sacrificed. The authors in [40] proposed a Light CNN model which is suitable for mobile devices. Although we use Light CNN for extracting feature vectors, our goal is to combine it with our privacy preserving protocol, and thus we have to optimize its performance when combining with the cryptographic primitives.

6 CONCLUSIONS

In this paper, we presented the design, implementation, and evaluation of TargetFinder, which can be applied for finding interested targets (e.g., suspect) privately anytime and anywhere. We proposed a privacy preserving protocol to protect the privacy of the target and the bystanders, and proposed efficient image processing techniques that can run on mobile devices. The proposed technique allows the requester (e.g., the police) to receive images of the target, while all other captured images of the bystanders are not revealed. Moreover, the target's face image is not revealed to the

server and the participating workers. Due to the high computation overhead of the cryptographic primitives, we proposed several optimization techniques in order to run our privacy preserving protocol on mobile devices. Through real world deployment and extensive performance evaluations, we have demonstrated the effectiveness of TargetFinder on privately locating targets through IoT cameras.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) under grants CNS-1526425 and CNS-1815465.

REFERENCES

[1] 2012. Java Lattice Based Cryptography Library. http://gas.dia.unisa.it/projects/jlbc/index.html
[2] 2013. Bouncy Castle. https://www.bouncycastle.org/
[3] 2016. Caffe Android lib. https://github.com/sh1r0/caffe-android-lib
[4] 2017. Libgcrypt. https://www.gnupg.org/software/libgcrypt/index.html
[5] 2018. Dlib C++ library. http://dlib.net/
[6] 2018. Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging/
[7] 2018. OpenCV library. http://opencv.org/
[8] 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*.
[9] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. 2016. NFLlib: NTT-based fast lattice library. In *Cryptographers' Track at the RSA Conference*. Springer.
[10] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual cryptology conference*. Springer.
[11] Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray CC Cheung, Derek Pao, and Ingrid Verbauwhede. 2015. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2015).
[12] Cheng-Kang Chu, Wen-Guey Tzeng, et al. 2008. Efficient k-out-of-n Oblivious Transfer Schemes. *Journal of Universal Computer Science* (2008).
[13] Liang Du, Meng Yi, Erik Blasch, and Haibin Ling. 2014. GARP-face: Balancing privacy protection and utility preservation in face de-identification. In *IEEE International Joint Conference on Biometrics*.
[14] Bin Guo, Huihui Chen, Zhiwen Yu, Xing Xie, Shenlong Huangfu, and Daqing Zhang. 2015. FlierMeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing. *IEEE Transactions on Mobile Computing* (2015).
[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in neural information processing systems*.
[16] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*.
[17] Huaizu Jiang and Erik Learned-Miller. 2017. Face detection with the faster R-CNN. In *IEEE International Conference on Automatic Face & Gesture Recognition*.
[18] Yurong Jiang, Xing Xu, Peter Terlecky, Tarek Abdelzaher, Amotz Bar-Noy, and Ramesh Govindan. 2013. Mediascope: selective on-demand media retrieval from mobile devices. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.
[19] Youssef Khazbak and Guohong Cao. 2017. Deanonymizing mobility traces with co-location information. In *IEEE Conference on Communications and Network Security (CNS)*.
[20] Youssef Khazbak, Jingyao Fan, Sencun Zhu, and Guohong Cao. 2018. Preserving Location Privacy in Ride-Hailing Service. In *IEEE Conference on Communications and Network Security (CNS)*.
[21] Erik Learned-Miller, Gary B Huang, Aruni RoyChowdhury, Haoxiang Li, and Gang Hua. 2016. Labeled faces in the wild: A survey. In *Advances in face detection and facial image analysis*. Springer.
[22] Qinghua Li and Guohong Cao. 2014. Providing efficient privacy-aware incentives for mobile sensing. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*.
[23] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.
[24] Andrea Melle and Jean-Luc Dugelay. 2014. Scrambling faces for privacy protection using background self-similarities. In *IEEE International Conference on Image Processing*.
[25] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. 2008. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.
[26] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. 2015. Deep Face Recognition.. In *Proceedings of the British Machine Vision*.
[27] Alberto Pedrouzo-Ulloa, Juan Ramón Troncoso-Pastoriza, and Fernando Pérez-González. 2017. Number theoretic transforms for secure signal processing. *IEEE Transactions on Information Forensics and Security* (2017).
[28] Hongwei Qin, Junjie Yan, Xiu Li, and Xiaolin Hu. 2016. Joint training of cascaded cnn for face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.
[29] Hang Qiu, Xiaochen Liu, Swati Rallapalli, Archith J Bency, Kevin Chan, Rahul Urgaonkar, BS Manjunath, and Ramesh Govindan. 2018. Kestrel: Video Analytics for Augmented Multi-Camera Vehicle Tracking. In *IEEE IoTDI*.
[30] Jeremy Schiff, Marci Meingast, Deirdre K Mulligan, Shankar Sastry, and Ken Goldberg. 2009. Respectful cameras: Detecting visual markers in real-time to address privacy concerns. In *Protecting Privacy in Video Surveillance*. Springer.
[31] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*.
[32] Andrew Senior, Sharath Pankanti, Arun Hampapur, Lisa Brown, Ying-Li Tian, Ahmet Ekin, Jonathan Connell, Chiao Fe Shu, and Max Lu. 2005. Enabling video privacy through computer vision. *IEEE Security & Privacy* (2005).
[33] Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, and Mahadev Satyanarayanan. 2013. Scalable crowd-sourcing of video from mobile devices. In *ACM Proceeding of the international conference on Mobile systems, applications, and services*.
[34] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[35] Jingchao Sun, Rui Zhang, Xiaocong Jin, and Yanchao Zhang. 2016. Securefind: Secure and privacy-preserving object finding via mobile crowdsourcing. *IEEE Transactions on Wireless Communications* (2016).
[36] Surat Teerapittayanon, Bradley McDanel, and HT Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *IEEE Pattern Recognition (ICPR)*.
[37] Hien To, Gabriel Ghinita, Liyue Fan, and Cyrus Shahabi. 2017. Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE Transactions on Mobile Computing* (2017).
[38] Matthew A Turk and Alex P Pentland. 1991. Face recognition using eigenfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*.
[39] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. 2017. A scalable and privacy-aware IoT service for live video analytics. In *ACM International Conference on Multimedia Systems (MMsys)*.
[40] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. 2018. A light CNN for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security* (2018).
[41] Yibo Wu and Guohong Cao. 2017. VideoMec: a metadata-enhanced crowdsourcing system for mobile videos.. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.
[42] Yibo Wu, Yi Wang, and Guohong Cao. 2017. Photo crowdsourcing for area coverage in resource constrained environments. In *IEEE INFOCOM*.
[43] Yibo Wu, Yi Wang, Wenjie Hu, and Guohong Cao. 2016. Smartphoto: a resource-aware crowdsourcing approach for image sensing with smartphones. *IEEE Transactions on Mobile Computing* (2016).
[44] Cha Zhang and Zhengyou Zhang. 2010. A survey of recent advances in face detection. *Technical Report MSR-TR-2010-66* (2010).