sTrack: Secure Tracking in Community Surveillance

Chun-Te Chu, Jaeyeon Jung, Zicheng Liu and Ratul Mahajan
Microsoft

Abstract--We present sTrack, a system that can track objects across distributed cameras without revealing any visual information to peering cameras except whether an object was seen by both parties. To achieve this challenging privacy goal, we build on recent advances in secure two-party computation and multi-camera object tracking. Starting from two distance-metric learning techniques that are foundational for many computer vision tasks, we derive a new technique that is much more suited for secure computation because it increases the computation that cameras do locally and simplifies the computation that they do jointly. At the same time, the tracking accuracy of our technique is similar or better than the original techniques. We implement our approach in a tracking system that uses a new Boolean circuit for secure matching. Experiments using real datasets show that the performance overhead of private tracking is small, adding only a few seconds of delay compared to non-private tracking.

1. INTRODUCTION

Camera-based surveillance is widely employed to fight against crime such as burglaries and vandalism. In conventional surveillance systems such as those deployed in an airport, subway, and corporate buildings, tracking across cameras is extremely useful for crime investigation. For example, when a criminal investigator spots a suspicious person entering a building from one of the surveillance videos, the investigator wants to know where and when this person left the building. Since a building usually has multiple entry and exit points, tracking across the cameras at these points is needed. In the past decade, tremendous progress has been made on cross-camera tracking technologies [13][33][34][40], and many commercial surveillance systems have the cross-camera tracking capability. Since all cameras have the same owner (a government agency or a company), there is little need to isolate the contents of different cameras.

However, privacy concerns severely limit the applicability of multi-camera tracking technologies to community surveillance, where residential neighborhoods must be monitored. Householders in many communities, including those in high-crime neighborhood, are loath to law enforcement deploying surveillance cameras in areas because of trust and privacy issues [4]. A primary concern is that the camera feeds contain many aspects of their life that have nothing to do with crime (and are potentially embarrassing).

People are mindful of not only their own privacy but also embarrassment that unfettered sharing, with the police or each other, can cause to their neighbors. Suppose an expensive bicycle is missing from a householder's front yard. The victim went through his surveillance video and found the images of the kids who took the bicycle. The victim does not recognize the kids and would thus like to trace to which home the kids fled. One way of doing this is to do a neighborhood-wide broadcast (or sharing with the police) of the images, but this may reveal the kids' identities to the entire neighborhood (or the police) and will potentially embarrass the parents publicly, which is not intended by the victim. Note that the only information that is needed to track the kids is whether certain other cameras have seen the same kids around the same time. Based on this information, the victim can locate where the kids live and privately inform the parents. The victim may also choose to warn other neighbors whose houses the kids may have scoped out.

The concerns around sharing camera feeds are unfortunate as many households already have security cameras, and processing scenes captured across them would enable a community-wide surveillance system to track suspicious objects across cameras that individual cameras alone are not able to infer (e.g., a person knocking on several doors, a car cruising the neighborhood).

Our goal is to enable individual cameras that currently operate in silo to cooperate in order to track suspicious objects (e.g., unknown people or cars traveling through the neighborhood) without revealing any information about the objects except matching outcomes. We assume that residents have full access to the video feed from their own cameras. Each time the camera identifies an object of interest (e.g., a car or a human), it communicates with peer cameras to determine whether the object has also been seen by them within a pre-defined time window in order to track the object. This determination can be made using a distance function over the object features (e.g., color histogram) that individual cameras extract. However, private information is revealed if the cameras directly share the feature vectors to compute the distance. For instance, it is possible to identify the object if color histogram is shared.

To address this privacy risk, we leverage secure two-party computation for privately matching objects. Secure two-party computation allows two participants to compute a publicly known function over their inputs, without revealing any information about each other's input except the output of the function (and anything that can be inferred from the output). Much progress [14][16][17][18][37][38] has been made recently towards reducing the CPU and memory overhead of Yao's original proposal [28].

In this paper, we explore if secure two-party computation can be applied to multi-camera object tracking problems, such that the accuracy is high and overhead is low. We make three contributions. First, we analyze state-of-art approaches for multi-camera tracking [9][31] for their overhead if they were implemented securely. We derive efficient Boolean

circuits for them and find that the overhead of these approaches is impractically high because of the complexity of the computation. This result is not surprising since the approaches were not designed with secure computation in mind.

Second, we present a new technique that significantly lowers the complexity of a Boolean circuit construction of the distance function for secure computation than the two state-of-the art techniques. Our technique has low overhead because it refactors the problem such that cameras do the complex portions of the computation (e.g., matrix multiplication) locally and the portions that are computed jointly are highly simplified (e.g., histogram intersection instead of Euclidean distance). Experiments with four real datasets show that the overhead of our technique, measured using the number of gates in the circuit, is 6-200 times lower than existing approaches and its accuracy is comparable or higher.

Like the other techniques that we analyzed, our technique is based on distance metric learning, where the distance function is learned through training. Distance metric learning is used in many other domains (e.g., matching hand-written text, face recognition, and matching biometric features). Thus, although we evaluate our approach for multi-camera object tracking, we believe that it also applies to other domain, rendering them privacy-preserving.

Third, we implement our technique and secure two-party computation into a distributed community surveillance system, presenting the first prototype of privacy-preserving multi-camera tracking.[1] Our experiments show the performance overhead of secure matching is minimal, adding only a few seconds latency compared to a completely non-private surveillance system.

2. BACKGROUND AND RELATED WORK

Our work builds on three themes of work—privacy-preserving video surveillance, secure two-party computation, and object tracking across cameras. In this section, we discuss (1) how previous work that combined computer vision and security influenced our work; (2) how the secure computation with garbled circuits are integrated into our system; and (3) how our approach compares with the state of the art in multi-camera object tracking.

2.1. Privacy Concerns in Video Surveillance

To our knowledge, our work is the first that focuses on privacy-preserving multi-camera object tracking. Much previous effort has focused on privacy concerns with respect to rogue operators of video surveillance deployed in the public area. Senior et al. [23] present an architecture that generates obfuscated images and summary data to hide privacy-intrusive details from unauthorized operators. Upmanyu et al. [26] propose image transformation techniques that enable a few surveillance tasks (e.g., change detection, face detection) on obfuscated images. Avidan et al. [1] introduce a face detection system that integrates secure computing techniques to protect the privacy of face images and the face detection algorithm. SCiFI [22] offers secure computation of face recognition that can be used for detecting known suspects. Although these proposed techniques are useful for each target setting, they are complementary to our effort and cannot be applied directly for multi-camera object tracking in which cameras do not trust each other.

2.2. Secure Two-Party Computation

A key contribution of our work is a new distance metric learning approach that is suitable for efficient, secure two-party computation. Our approach simplifies secure computation to be only composed of addition, min, and comparison functions of small dimensional vectors instead of matrix multiplication of large dimensional feature vectors. This allows us to use a general, circuit-based approach for secure computation without sacrificing performance, instead of using custom protocols (as is done in SCiFI [22]).

We implement secure matching using Huang et al.'s library [14] which contains garbled circuits for commonly-used functions such as addition, subtraction, and comparison. Our garbled circuits implement the optimizations proposed by Kolesnikov et al. [16]. Our system, therefore, benefits from previous works [14][16][37] and also shares their security assumption (semi-honest adversaries). As new techniques are developed to improve the security or efficiency of the circuits for secure two-party computation (e.g., [18]), our system will inherit these benefits.

2.3. Object Tracking across Cameras

Multi-camera object tracking consists of two subtasks—extracting features of objects and matching the feature vectors of two objects to determine if they are the same.

Feature extraction: We focus on tracking people and cars across cameras. Many computer vision algorithms use face or license plate features for identifying human or cars. However, in an uncontrolled environment like a neighborhood, these features are unreliable since cameras may not always capture a clear face or license plate. Hence, we use the whole-body appearance as the basis of features. Global features are easy to compute and are shown to be effective. In particular, color histograms and texture features of the whole-body or different parts of the body were used in various computer vision algorithms [8] [12] [30].

Feature matching: Once feature vectors are available, matching functions compute the distance between them to determine object similarity. To overcome the limitation of traditional metric functions such as Euclidean and Hamming distance, recent computer vision algorithms employ machine learning approaches to "learn" a distance metric that captures the relationship of training data [6][31]. Once the metric is learned (through training), distance evaluation is essentially equivalent to performing distance calculation in the transformed feature space. The metric learning approaches enable effective matching to be performed using simple features like color histograms. For instance, Zheng et al. [31] and Dikmen et al. [6] showed that using color histograms and the learned metric, their algorithms performed better than the existing

[1] We are making our prototype available to the research community. Details elided to preserve the anonymity of the submission.

methods that rely on traditional metrics. We build on these works to show that 1) privately implementing current methods incurs high overhead, and 2) distance computation can be transformed such that its overhead is significantly lowered and accuracy is not impacted.

3. GOALS AND OVERVIEW

Our target setting is a cooperative community of homes, where each home has one or more security cameras. These homes are interested in jointly monitoring the neighborhood (as in neighborhood watch program) by combining the views across all the cameras. The power of this combination stems from the fact that collectively the cameras can infer patterns that individual cameras cannot, such as, a car cruising the neighborhood (a common pattern that neighborhood watch volunteers look out for). One way to accomplish this goal is for each home to send its feed to a central party (e.g., law enforcement or a corporation), and for this party to run one of the existing surveillance algorithms on the combined input. However, many householders loathe sharing their camera data with third-parties and prefer a communal solution [4].

Thus, we investigate an alternative, *peer-to-peer* approach in which cameras are completely autonomous and communicate with each other to determine if they saw the same object. We want to enable this determination in a privacy-preserving manner, i.e., a camera should not be able to infer anything about what other cameras see, except whether they also saw a particular object. Privacy is desirable even in this cooperative setting as cameras capture many facets of residents' lives (e.g., times they come home, who visits them, etc.) that they may not feel comfortable sharing. Sharing of raw camera images (or feature vectors) would reveal such facets to the neighbors. It also runs the risk of those images becoming broadly available due to lax security in a neighbor's system.

The focus of this paper is developing a practical technique using which two cameras can determine if they saw the same object without ever sharing raw images. We show that this is possible to accomplish, with acceptable performance and without any degradation in object matching accuracy compared to non-private matching. Our approach does not require any trusted third parties and the tracking system can grow organically as more people join.

We aim to deploy the system in the neighborhood community area where the households are willing to collaborate in order to maintain the public security of the community. Therefore, just like other works in privacy-preserving vision applications [1][22][26][41][42][43], we assume an honest-but-curious (semi-honest) adversary model in our work. This model suffices for our setting where people in the community area follow the regulations for keeping the community safe. Our intent is that the tracking system does not become an easy conduit for privacy violations by curious neighbors.

While sTrack provides the basic private matching capability, it is not a complete neighborhood watch solution yet. Such a solution would also alert users based on matching results and handle possible matching errors. As we show later, these errors are inherent in the state of art vision algorithms. (sTrack does not make them more likely.) One way to handle matching errors is to release raw images of suspicious objects that match, after seeking users' approval. This way the owners can determine if something embarrassing is being shared. When used in this way, sTrack ensures that users need to verify only a subset of images that do match. Extending sTrack into a complete neighborhood watch solution is a subject of future work.

We now present the high level architecture and key components of sTrack. Later sections describe our system in more detail.

Table 1. Gate counts for basic garbled circuits

basic circuits		non XOR gates
addition	$ADD(l)$	l [16]
subtraction	$SUB(l)$	l [16]
multiplication	$MUL(l)$	$2l^2 - l$ [16]
comparison	$CMP(l)$	l [16]
multiplexer	$MUX(l)$	l [17]
minimum	$MIN(l)$	$2l$ [16]

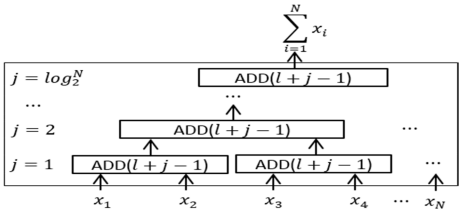

Figure 1. ADD*(N, l) Garbled Circuit Construction

Object detection and tracking within a camera: Individual camera process input frames and extracts objects. For each frame, they keep track of time, location, and feature vectors of all the objects appearing in it. Since the same object can appear in multiple frames, we associate these appearances to track objects across frames.

Feature vector preparation: The system prepares an input feature vector of an object under two conditions: (1) when an object exits the view--within-camera tracking allows us to differentiate between entering and exiting objects, and (2) when the system (or user) identifies an object of interest and wants to sends a matching request to peer cameras. All prepared input feature vectors are stored in a database for matching. Preparing input feature vectors involve two steps: i) extracting real-valued feature vectors (color histograms in our case) from an object, and ii) transforming them based on the parameters learned in the training stage and quantizing for secure distance evaluation. All computations thus far take place locally and do not involve any information exchange with peering systems.

Secure matching: A camera issues a secure matching request to peers when the system records that a suspicious object has entered the view (i.e., Condition (2) in the feature vector preparation process). The output of secure matching function is binary—match or no-match—based on two input feature vectors. An individual system may use its own crite-

ria to deem an object as suspicious or labeled as such by a human. In our current prototype, a system issues a secure matching request for all objects and to all peering cameras. The requester becomes a client and the responding system becomes a server for secure function evaluation. Upon receiving a request, the responding system looks up its database to count the number of candidate objects to match. The system then opens the same number of communication channels to perform secure function evaluation with the requester. The following sections discuss in detail how our system implements these computer vision tasks and how we design a new distance learning technique to make the secure matching function evaluation efficient.

4. ANALYZING EXISTING TECHNIQUES

Tracking objects across cameras is done by matching objects seen by different cameras. Formally, two cameras collaborate to evaluate a distance function $D(\mathbf{x}_i, \mathbf{x}_j)$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the feature vectors extracted by them for the objects to be matched. If the distance is smaller than a threshold, it is deemed that the objects are the same. The matching accuracy depends on the distance function $D(\cdot,\cdot)$.

This section considers two recent techniques to learn a distance function $D(\cdot,\cdot)$. It presents our analysis of how each function can be decomposed into local and joint computation. The decomposition is important as the complexity of only the joint portion determines the complexity of secure computation. We construct efficient garbled circuits for each technique and find that they have high overhead. We also consider a simple variant of one of the techniques; we find that this variant has low complexity, but poor accuracy. Based on these insights, we present in Section 5 a new distance metric learning technique, called GCDC, which greatly reduces the complexity of the joint computation while being highly accurate.

4.1. Background: GC-based for Secure Computation

Yao's garbled circuit approach provides a foundation to construct a secure two-party protocol for computing a function represented as a Boolean circuit [28]. Security proofs are available for GC protocols against semi-honest (i.e., honest but curious) adversaries [20]. Software packages are available for constructing GC protocols [14][21][38][39]. We focus on optimizing the matching function, $D(\cdot,\cdot)$, such that it allows efficient Boolean circuits. Hence, our work complements and benefits from many recent works that focus on optimizing GC systems [14][16][17][18][38][39].

The secure computation protocol starts with a Boolean circuit for $D(\mathbf{x}_i, \mathbf{x}_j)$. One party (the circuit generator) prepares a garbled version of the circuit, and the second party (the circuit evaluator) obliviously computes the circuit's output. Garbling involves generating keys to each wire and the garbled truth table for each gate. Thus, the number of gates determines the complexity of a GC, impacting the performance of generating and evaluating the GC. One exception is XOR gates which come "free" as shown by Kolesnikov et al. [17], requiring no crypto operations.

Table 1 shows the size, measured in terms of the number of non XOR gates, of efficient circuit constructions for basic functions that compute on two l-bit integers. A key point to bear in mind is that the size of a GC increases roughly in the order of $O(l^2)$ for multiplication whereas it increases linearly for other basic circuits. See references [16][17] for details of how these circuits can be constructed with free XOR gates.[2] In constructing GCs, we use efficient circuits from prior work [14][16][32] whenever applicable.

Moreover, we construct an efficient circuit for a commonly-used function in our domain, summation of N l-bit integers, denoted as $ADD^*(N,l) = \sum_{j=1}^{N} x_j$, where x_j is an l-bit integer. Figure 1 shows the GC construction. ADD circuits are connected in a hierarchical fashion to minimize table size. We assume that N is in power of 2. Appendix C describes an algorithm to construct $ADD^*(N,l)$ when N is not in power of 2. The output of $ADD^*(N,l)$ is a $(l + log_2 N)$-bit integer. Since the number of non XOR gates of $ADD(l)$ circuit is $|ADD(l)| = l$ as shown in Table 1, the size of $ADD^*(N,l)$ is

$$|ADD^*(N,l)| = \sum_{j=1}^{log_2 N} \frac{N}{2^j} \cdot |ADD(l+j-1)| = Nl + N - l - 1 - log_2 N\,. \quad (1)$$

4.2. Background: Distance Metric Learning

Metric learning has been shown to be highly effective at improving matching accuracy [6][31]. It can be described as follows. Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the original real-valued feature vector extracted from the object i, where d is the number of dimensions. Let $\mathbb{O}^p$ denote the positive training set consisting of matched feature vector pairs, and $\mathbb{O}^n$ denote the negative training set consisting of unmatched feature vector pairs. The goal of metric learning is to learn a distance function $D(\cdot,\cdot)$ that can discriminate positive and negative examples. That is, we would like $D(\mathbf{x}_i, \mathbf{x}_j)$ to be small when $\mathbf{x}_i$ and $\mathbf{x}_j$ correspond to the same object and large otherwise. In the following techniques, learning $D(\cdot,\cdot)$ is equivalent to learning the parameter $\mathbf{W}$.

4.3. PRDC

Zheng et al. propose [31] a distance metric earning method called PRDC (person re-identification by Probabilistic Relative Distance Comparison), for object matching. Its distance function is:

$$D_1(\mathbf{x}_i, \mathbf{x}_j) = \left|\mathbf{x}_i - \mathbf{x}_j\right|^T \mathbf{W}\,\mathbf{W}^T \left|\mathbf{x}_i - \mathbf{x}_j\right| = \left\| \mathbf{W}^T \left|\mathbf{x}_i - \mathbf{x}_j\right| \right\|^2, \quad (2)$$

where $|\cdot|$ is the entry-wise absolute function; $\|\cdot\|$ is *L2* norm; $\mathbf{W} = [\,\mathbf{w}_1 \ldots \mathbf{w}_L\,] \in \mathbb{R}^{d \times L}$. They learn a matrix $\mathbf{W}$ consisting

[2] The minimum circuit (which takes two l-bit integers and outputs one of the input values that is less than or equal to the other input value) is not discussed in [16]. However, it can be easily constructed by connecting one l-bit multiplexer circuit and one l-bit comparison circuit.

Table 2. PRDC computation

	intermediate steps	circuits needed
1	$\lvert \mathbf{x}_i - \mathbf{x}_j \rvert$	d subtraction d absolute value
2	$\mathbf{W}^T \lvert \mathbf{x}_i - \mathbf{x}_j \rvert$	$d \times L$ multiplication $(d-1) \times L$ addition
3	$\lVert \mathbf{W}^T \lvert \mathbf{x}_i - \mathbf{x}_j \rvert \rVert^2$	L multiplication $(L-1)$ addition

of a small set of d-dimensional basis ($L \ll d$) in order to separate different objects well.

Note that $\lVert \mathbf{W}^T \lvert \mathbf{x}_i - \mathbf{x}_j \rvert \rVert^2$ needs to be computed privately (i.e., jointly) as it involves both feature vectors. As a result, the computation of the function in secure two-party computation involves subtraction, absolute value calculation, multiplication, and addition. Moreover, the computation often operates in high dimensional space, i.e., d is usually quite large, which adds much overhead. Table 2 shows the computational complexity of the secure two-party computation implementation of $D_1(\mathbf{x}_i, \mathbf{x}_j)$.

We now construct an efficient GC for PRDC. We assume that each input element is quantized to be an l-bit integer (We discuss in Section 6 how we picked the quantization level, l, based on experimental results). The first step of Table 2 requires computing the absolute value of the subtraction of two l-bit integers. This circuit can be constructed by connecting 2Sorter(l) (which takes two l-bit integers, x and y, and outputs $min(x,y)$ and $max(x,y)$) [32] and one regular subtraction circuit. So, the Step 1 requires d numbers of 2Sorter(l) and $SUB(l)$ circuits. The Step 2 of Table 2 performs $d \times L$ multiplications of two l -bit integers (which are the outputs of the Step 1). Then, it adds d elements of the multiplication results, which are $2l$-bit integers. The Step 3 of Table 2 first performs L multiplications of two $(2l + log_2\, d)$-bit integers and the output of each multiplication, which is a $(4l + 2\, log_2\, d)$-bit integer is summed up (L elements total) and the output which is a $(4l + 2\, log_2\, d + 2\, log_2\, L)$ -bit integer is compared against a predefined threshold estimated in the training stage. In sum, the number of non XOR circuits is

$$\begin{aligned} |PRDC| &= d \cdot |2Sorter(l)| + d \cdot |SUB(l)| + dL \cdot |MUL(l)| \\ &\quad + L \cdot |ADD^*(d, 2l)| + L \cdot |MUL(2l + log_2\, d)| \\ &\quad + |ADD^*(L, 4l + 2\, log_2\, d)| \\ &\quad + |CMP(4l + 2\, log_2\, d + log_2\, L)| \\ &= 2dl^2L + dlL + 8l^2L + 3dl + dL + 2L(log_2\, d)^2 \\ &\quad + 8lL\, log_2\, d - 1\,. \end{aligned} \quad (3)$$

Since d is large and multiplication is expensive in secure two-party computation, PRDC incurs high overhead.

One possible remedy is to get rid of the absolute-value function as shown in (4)

$$D_2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}\mathbf{W}^T (\mathbf{x}_i - \mathbf{x}_j) = \lVert \mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j \rVert^2, \quad (4)$$

Table 3. PRDC w/o absolute computation

	intermediate steps	circuits needed
1	$\mathbf{W}^T\mathbf{x}_i$, $\mathbf{W}^T\mathbf{x}_j$	(local computation)
2	$\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j$	L subtraction
3	$\lVert \mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j \rVert^2$	L multiplication $(L-1)$ addition

where $\mathbf{W} = [\,\mathbf{w}_1 \ldots \mathbf{w}_L] \in \mathbb{R}^{d \times L}$ and $L \ll d$. Note that $\mathbf{W}^T\mathbf{x}_i$ and $\mathbf{W}^T\mathbf{x}_j$ can be computed locally by each camera (without secure two-party computation). As a result, the computational complexity is reduced significantly (Table 3). However, we find that matching accuracy also degrades significantly (Section 6). In constructing an efficient GC for this function, we again assume that $\mathbf{W}^T\mathbf{x}_i$ is quantized to be an l-bit integer and construct the circuit for each step of Table 3. We omit the details of circuit construction as they are similar to above. The circuit size is

$$\begin{aligned} |PRDC\_woABS| &= L \cdot |SUB(l)| + L \cdot |MUL(l)| + |ADD^*(L, 2l)| \\ &\quad + |CMP(2l + log_2\, L)| \\ &= 2l^2L + 2lL + L - 1. \end{aligned} \quad (5)$$

4.4. MCC

In order to lower complexity without sacrificing accuracy, we consider a different approach, MCC (Metric learning by Collapsing Classes) [9], which has been shown to have good performance for visual matching [31]. Since the method in [9] requires large amount of computation in the training stage, especially if the feature space is high dimensional, we use Principal Component Analysis (PCA) [15] for dimensionality reduction. Assume $\hat{\mathbf{x}}_i \in \mathbb{R}^r$ ($r \ll d$) is the new feature vector after PCA,

$$\hat{\mathbf{x}}_i = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}}), \quad (6)$$

where $\mathbf{P} \in \mathbb{R}^{d \times r}$ is projection matrix that maps the original d-dimension space into r dimensions ($r \ll d$); $\bar{\mathbf{x}} \in \mathbb{R}^d$ is the mean vector over the training data. The distance function is expressed as

$$D_3(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) = (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)^{\mathrm{T}} \mathbf{W}\mathbf{W}^T (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j) = \lVert \mathbf{W}^T\hat{\mathbf{x}}_i - \mathbf{W}^T\hat{\mathbf{x}}_j \rVert^2, \quad (7)$$

where $= [\,\mathbf{w}_1 \ldots \mathbf{w}_q] \in \mathbb{R}^{r \times q}$, $r \ll d$, and $q \leq r$. Similar to (4), $\mathbf{W}^T\hat{\mathbf{x}}_i$ and $\mathbf{W}^T\hat{\mathbf{x}}_j$ can be computed locally without invoking secure two-party computation. In addition, the dimensionality reduction is also performed locally. However, secure computation still needs expensive multiplication circuits (Table 4).

In constructing a GC for MCC, we again assume that $\mathbf{W}^T\hat{\mathbf{x}}_i$ is quantized to be an l-bit integer and construct the circuit for each step of Table 4. This circuit is similar to PRDC_woABS but with L replaced by q. The circuit size is

$$\begin{aligned} |MCC| &= q \cdot |SUB(l)| + q \cdot |MUL(l)| + |ADD^*(q, 2l)| \\ &\quad + |CMP(2l + log_2\, q)| \\ &= 2l^2q + 2lq + q - 1. \end{aligned} \quad (8)$$

Table 4. MCC computation

	intermediate steps	circuits needed
1	$\hat{\mathbf{x}}_i = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})$	(local computation)
2	$\mathbf{W}^T\hat{\mathbf{x}}_i,\ \mathbf{W}^T\hat{\mathbf{x}}_j$	(local computation)
3	$\mathbf{W}^T\hat{\mathbf{x}}_i - \mathbf{W}^T\hat{\mathbf{x}}_j$	q subtraction
4	$\left\| \mathbf{W}^T\hat{\mathbf{x}}_i - \mathbf{W}^T\hat{\mathbf{x}}_j \right\|^2$	q multiplication $q-1$ addition

5. OUR TECHNIQUE: GCDC

Based on experience with the analysis above, we developed a new technique called GCDC (Garbled Circuit Distance Computation). Our goal is to push most computation to be local and minimize joint computation. We observe that the distance function $D_3(\cdot,\cdot)$ is equivalent to computing the Euclidean distance in another space specified by $\mathbf{W}$. In order to avoid the multiplication operation, we replace the Euclidean distance function with the histogram intersection function after linearly projecting to a new space by $\mathbf{W}$. The histogram intersection function (9) has also been shown as an effective metric in classification and pattern recognition [19][29] and it is defined as

$$HI(\mathbf{h}_i, \mathbf{h}_j) = \sum_{k=1}^{q} min(h_i^k, h_j^k), \tag{9}$$

where $\mathbf{h}_i = [h_i^1 \dots h_i^q]^T \in \mathbb{R}_+^q$, $\mathbf{h}_j = [h_j^1 \dots h_j^q]^T \in \mathbb{R}_+^q$ are the vectors with nonnegative entries. The new distance function is:

$$\begin{aligned} D_4(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) &= -HI(\mathbf{W}^T\hat{\mathbf{x}}_i + T\mathbf{1},\ \mathbf{W}^T\hat{\mathbf{x}}_j + T\mathbf{1}) \\ &= -\sum_{k=1}^{q} min(\mathbf{w}_k{}^{\mathrm{T}}\hat{\mathbf{x}}_i + T,\ \mathbf{w}_k{}^{\mathrm{T}}\hat{\mathbf{x}}_j + T), \end{aligned} \tag{10}$$

where $\mathbf{W} = [\mathbf{w}_1 \dots \mathbf{w}_q] \in \mathbb{R}^{r\times q}$, $r \ll d$ and $q \le r$, and $T \ge 0$ is a predefined parameter that ensures the nonnegativity of $\mathbf{w}_k{}^{\mathrm{T}}\hat{\mathbf{x}}_i + T$ for any feature vector $\hat{\mathbf{x}}_i$. Note that although the value of $D_4(\cdot,\cdot)$ seems to be negative, we can always add a positive constant scalar to make it a valid distance metric, and it will not change the formulation of our metric learning, so we discard the scalar here and in the following discussion. The secure two-party computation complexity of distance function $D_4(\cdot,\cdot)$ is shown in Table 5. Observe that significant computation is local and joint computation uses simple operations.

However, just defining $D_4(\cdot,\cdot)$ is not enough. For it to be a valid approach, we must also develop an efficient way to learn $\mathbf{W}$. Our approach for doing so is motivated by [9]. Our goal is to learn a linear projection matrix, such that the pairs in set $\mathbb{O}^p$ have small distances, and the pairs in set $\mathbb{O}^n$ have large distances. Define a conditional distribution over points $i \ne j$ such that

$$p^{\mathbf{W}}(j|i) = \frac{e^{-D_4(\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j)}}{Z_i} = \frac{e^{-D_4(\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j)}}{\sum_{k\ne i} e^{-D_4(\hat{\mathbf{x}}_i\hat{\mathbf{x}}_k)}} \qquad i \ne j. \tag{11}$$

Ideally, if all the pairs in set $\mathbb{O}^p$ have small distance, and all the pairs in set $\mathbb{O}^n$ have large distances, the distribution would become "bi-level", that is,

Table 5. Our GCDC computation

	intermediate steps	circuits needed
1	$\hat{\mathbf{x}}_i = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})$	(local computation)
2	$\mathbf{W}^T\hat{\mathbf{x}}_i + T\mathbf{1},\ \mathbf{W}^T\hat{\mathbf{x}}_j + T\mathbf{1}$	(local computation)
3	$\min(.)$	q min operation
4	$\sum_{k=1}^{q} \min(.)$	$(q-1)$ addition

$$p_0(j|i) \propto \begin{cases} 1 & if\ \ (\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) \in \mathbb{O}^p \\ 0 & if\ \ (\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) \in \mathbb{O}^n \end{cases} \tag{12}$$

Therefore, the cost function $f_1(\mathbf{W})$ is defined as

$$\begin{aligned} f_1(\mathbf{W}) &= \sum_i KL[p_0(j|i) \,|\, p^{\mathbf{W}}(j|i)] \\ &= \sum_i \sum_{j\ne i} p_0(j|i) \times log(\tfrac{p_0(j|i)}{p^{\mathbf{W}}(j|i)}), \end{aligned} \tag{13}$$

where $KL[\cdot\,|\,\cdot]$ is K-L divergence [3] which measures the distance between two distributions. Substituting (11) and (12) into (13), we obtain,

$$f_1(\mathbf{W}) = \sum_{\substack{i,j, \\ (\hat{\mathbf{x}}_i,\hat{\mathbf{x}}_j)\in\mathbb{O}^p}} D_4(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) + \sum_i log(Z_i) \tag{14}$$

We introduce a regularization term $f_2(\mathbf{W})$ to bound the values of $\mathbf{W}^T\hat{\mathbf{x}}_i$ in such a way that we can always find a nonnegative scalar T to make all the entries in $\mathbf{W}^T\hat{\mathbf{x}}_i + T\mathbf{1}$ nonnegative for all i (Appendix A).

$$f_2(\mathbf{W}) = \sum_{k=1}^{q} \mathbf{w}_k{}^T\mathbf{w}_k = Tr(\mathbf{W}^T\mathbf{W}) \tag{15}$$

Moreover, to satisfy the equality of self-distance, that is,

$$D_4(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k) = D_4(\hat{\mathbf{x}}_l, \hat{\mathbf{x}}_l) \qquad \forall\ k \ne l \tag{16}$$

an additional term $f_3(\mathbf{W})$ is added (Appendix B),

$$f_3(\mathbf{W}) = \sum_i (\sum_{k=1}^{q} \mathbf{w}_k{}^T\hat{\mathbf{x}}_i)^2 \tag{17}$$

The final objective function is the sum of the above three terms:

$$J(\mathbf{W}) = f_1(\mathbf{W}) + f_2(\mathbf{W}) + f_3(\mathbf{W}) \tag{18}$$

The metric learning problem is formulated as finding $\mathbf{W}$ that minimizes the objective function $J(\mathbf{W})$:

$$\widehat{\mathbf{W}} = \arg\min_{\mathbf{W}} J(\mathbf{W}) \tag{19}$$

Gradient descent method is employed to solve the optimization problem. The gradient vector is

$$\begin{aligned} \tfrac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} &= \sum_i \sum_{j\ne i} ((p_0(j|i) - p^{\mathbf{W}}(j|i)) \times g(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j, \mathbf{w}_k)) \\ &\quad + 2\mathbf{w}_k + 2\sum_i ((\sum_{l=1}^{q} \mathbf{w}_l{}^T\hat{\mathbf{x}}_i)\,\hat{\mathbf{x}}_i), \end{aligned} \tag{20}$$

where

$$g(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j, \mathbf{w}_k) = \begin{cases} -\hat{\mathbf{x}}_i & if\ \ \mathbf{w}_k{}^T\hat{\mathbf{x}}_i < \mathbf{w}_k{}^T\hat{\mathbf{x}}_j \\ -\hat{\mathbf{x}}_j & if\ \ \mathbf{w}_k{}^T\hat{\mathbf{x}}_i > \mathbf{w}_k{}^T\hat{\mathbf{x}}_j \\ -\frac{1}{2}(\hat{\mathbf{x}}_i + \hat{\mathbf{x}}_j) & if\ \ \mathbf{w}_k{}^T\hat{\mathbf{x}}_i = \mathbf{w}_k{}^T\hat{\mathbf{x}}_j \end{cases}$$

We now describe an efficient GC for GCDC. We first discuss preprocessing steps (Steps 1 and 2 of Table 5) and then discuss circuit size. Given any real-valued feature vec-

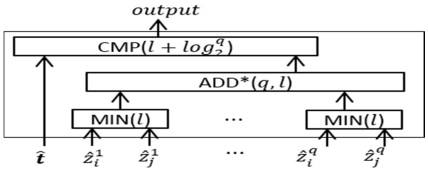

Figure 2. Garbled Circuit of GCDC

tor $\mathbf{x}_i \in \mathbb{R}^d$, it will be transformed into $\mathbf{z}_i \in \mathbb{R}^q$ with much lower dimension based on PCA and linear mapping,

$$\mathbf{z}_i = \mathbf{W}^T\mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}}) + T\mathbf{1}, \quad (21)$$

where $\mathbf{W}$, $\mathbf{P}$, $\bar{\mathbf{x}}$ and T are the parameters obtained in the training stage, and each party knows them. Since the secure two-party computation only takes integer input, each entry of $\mathbf{z}_i$ is first normalized to a real number from 0 to 1 and further quantized into an l-bit integer value to form a q-dimensional input vector $\hat{\mathbf{z}}_i$. Finally, the matching between input vectors $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$ is performed through secure two-party computation. Through the secure two-party computation [14], there is no information leakage, so the privacy is guaranteed. GCDC simplifies the GC needed for secure function evaluation, as it only requires integer comparison (CMP), min (MIN), and addition (ADD). The circuit requires two vectors, $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$ (q l-bit integers) and the comparison threshold, $\hat{t}$. Figure 2 shows the GC of GCDC method. The circuit size is

$$|GCDC| = q \cdot |MIN(l)| + |ADD^*(q,l)| + |CMP(l + log_2 q)|$$
$$= 3ql + q - 1\,. \quad (22)$$

6. COMPARING THE TECHNIQUES

We now use four real datasets to evaluate GCDC and the three techniques in Section 4. We show that 1) GCDC has comparable or better accuracy, and 2) the number of gates for GCDC is 6x fewer than MCC and over 200x fewer than PRDC. PRDC_woABS has fewer gates than GCDC but has very poor accuracy.

Test datasets: We use two public datasets, VIPeR [11] and i-LIDS [24][25], and two of our own datasets, which we call *human dataset* and *car dataset*. VIPeR is the world largest publicly available person re-identification dataset. It consists of the well-cropped snapshot images of 632 people under outdoor scene, e.g., Figures 3(a) and 3(b). For each person, two images are captured under different viewing angles and lighting conditions which make the appearance vary and increase the difficulty of the re-identification. i-LIDS is extracted from a multiple-camera tracking video scenario [25] captured in an airport area. We use the subset iLIDS-AA, which has images of 100 people. Example images are shown in Figures 3(c) and 3(d).

To collect our datasets, we set up two cameras in our building pointing at two different adjacent streets and collect multiple video clips.[3] The clips are captures at different

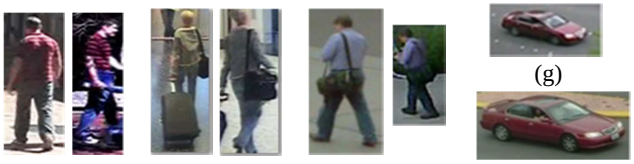
(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3. Examples of the snapshots in the database. (a) and (b) are from VIPeR database, and (c) and (d) are from iLIDS dataset. In these two datasets, the images are already normalized into the same size. (e) and (f) are from our human database, and (g) and (h) are from our car dataset.

times of the day and total 207 minutes. Because the cameras are at different locations, the size and perspective of the objects they capture is different. Some cropped images are shown in Figures 3(e)-(h). From the video clips that we captured, we form two datasets, one with humans and one with cars. For the human dataset, we extract 114 people from 100 minutes video clips. For the car dataset, we extract 83 cars from 40 minutes video clips.

Experimental methodology: To focus on the effectiveness of various distance metric learning methods, we ran the experiments by using manually-cropped snapshots of objects (e.g., Figure 3), so the within-camera tracking errors do not affect the results. For VIPer and iLIDS dataset, the cropped objects are already provided; for our two datasets, we manually cropped objects from video frames. In Section 8.1, we will quantify the end-to-end error of our system in realistic settings where the objects are extracted automatically by sTrack.

For each dataset, we randomly select 80 objects (68 for our car dataset) as the training set and 15 objects as the testing set in each trial. The training set and testing set do not overlap, i.e., each object only appears either in training set or testing set. In the training set, a pair of snapshots of each person or car under different views form the positive set $\mathbb{O}^p$, and all pairs of images of different people are the negative set $\mathbb{O}^n$. They are used in learning the parameters mentioned in Section 5. While doing the testing, we evaluate the matching functions against 15 objects based on the learned parameters. However, in real settings, a camera may capture objects at a faster (slower) rate if pointed at a busy (quite) street. We use 15 to represent a moderately busy street. We report the average of ten trials for each dataset.

As mentioned earlier, for humans, we use color histogram as the input feature because it is resilient against the scale of the object. For cars, the feature is extracted in the similar manner except that the multiple stripes are extracted based on the principal axis of the car. In this way, the feature representation is rotation invariant.

[3] The cameras deployed in our experiment recorded videos of passers-by, making them implicit human subjects in our video data set. To allay privacy concerns, one author was stationed at the site explaining that the video would be used to evaluate human and car tracking algorithms. While our institution does not require ethics board approval, had we been required to, we would have noted that the observations were made in a public place, are not dissimilar to other surveillance in public places, and that the observations would be used exclusively for training and testing the effectiveness of algorithms-not to identify specific individuals or behaviors.

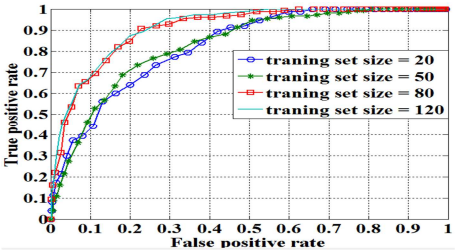

Figure 4. ROC curve with varying training set sizes (VIPeR)

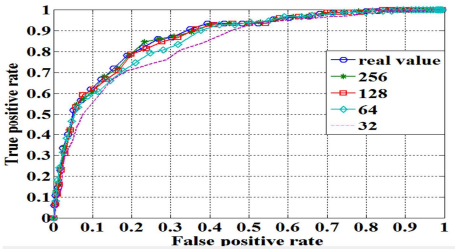

Figure 5. ROC curve with varying quantization levels (iLIDS)

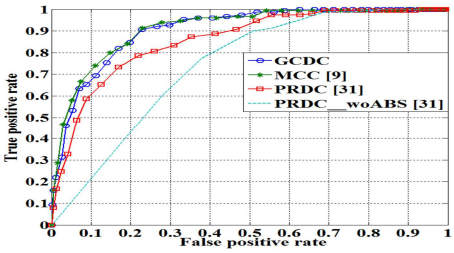

Figure 6. ROC curve of VIPeR database.

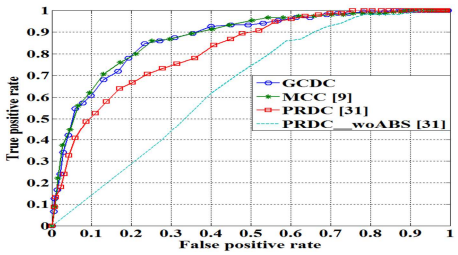

Figure 7. ROC curve of iLIDS database.

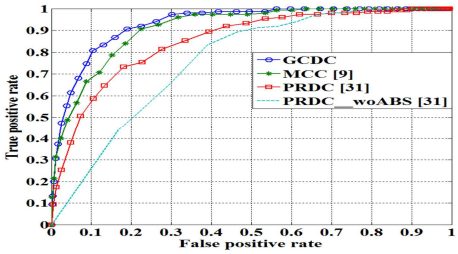

Figure 8. ROC curve of our human database.

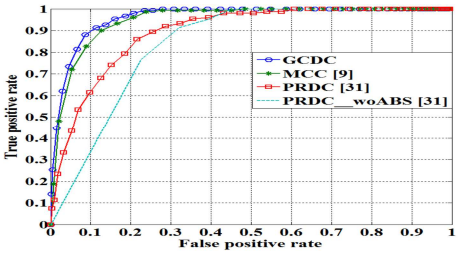

Figure 9. ROC curve of our car database.

We use the Receiver Operational Characteristic (ROC) curve as the accuracy evaluation metric[4]. ROC curve is commonly used for binary classification evaluation. In our case, we adjust the threshold and evaluate the true positive rate (TPR) and the false positive rate (FPR) for each threshold. The more accurate the method is, the ROC curve would be close to the upper-left corner, which corresponds to high TPR and low FPR. For the computation of PCA, we choose the principal components so that the cumulative energy exceeds 0.95. The scalar T in (21) is set as 1.5.

Impact of the training set on accuracy: Before presenting comparison results, we describe how we determined training set size and quantization levels in our experiment (and our system). The size of the training set of course affects matching accuracy. The ROC curves under different training set sizes are presented in Figure 4. It shows that the accuracy goes up as the training set size increases from 20 to 120. However, the accuracy improves only slightly when the size of training set increases beyond 80. We thus use 80 as the size of training data in our experiments.

Impact of quantization on accuracy: To generate integer inputs to a Boolean circuit for secure computation, we quantize the input vector into one of a finite set of prescribed integer values. The larger the quantization level, the smaller the impact on accuracy. Figure 5 shows the ROC under different quantization levels of the tests on iLIDS. We also include the result of using original real value without quantization for comparison. As shown in the figure, the quantization level should be at least 128 levels for good accuracy; that is, 7-bit or above is required to represent each entry of the input vector to secure two-party computation. To minimize the loss of accuracy, we use 256 levels($l = 8$) in all our experiments.

6.1 Accuracy results

First, we present the accuracy results for the VIPeR dataset. Figure 6 shows the ROC curves under different matching functions. Interestingly, the PRDC_woABS method incurs low cost for secure computation, but the accuracy drops significantly compared to the rest. On the other hand, GCDC has similar accuract to MCC while reducing the computational cost (next section).

Figure 7 shows the results for the iLIDS dataset. As for the VIPeR dataset, GCDC has no accuracy degradation. The performance of all methods in the iLIDS dataset is lower than the VIPeR dataset due to occlusions (e.g., the person with a luggage in Figure 3(c)).

Finally, we study our two datasets. Figures 8 and 9 again show that GCDC has comparable accuracy to PRDC and MCC, the two state of the art methods. Since the size of the

[4] When raw distance value between inputs is available (which is not in our case), it is better to use the Cumulative Matching Characteristics (CMC) curve [31] as the accuracy evaluation metric.

Table 6. Summary table for VIPeR dataset
(d =864; L =4; q=71)

VIPeR dataset	non XOR gates	AUC
PRDC	499,623	0.8515
PRDC_woABS	579	0.7468
MCC	10,298	0.9059
GCDC	1,778	0.9017

Table 7. Summary table for iLIDS dataset
(d =864; L =4; q=76)

iLIDS dataset	non XOR gates	AUC
PRDC	499,623	0.8151
PRDC_woABS	579	0.6404
MCC	11,022	0.8697
GCDC	1,902	0.8653

object is larger in our database than in the others, the histogram-based features are more representative to the objects, leading to better performance for most methods than in the VIPeR and iLIDS datasets.

6.2. Trade-off between Accuracy and Overhead

Tables 6-9 summarize the matching accuracy and overhead for different methods. As the measure of overhead, we compute the number of non-XOR gates using the equations (3)(5)(8)(22) and the parameters shown in each table. We use AUC (Area Under the Curve) to represent accuracy characteristics of each method. AUC value ranges from 0 to 1. As ROC curves get closer to the upper-left corner (low false positive rate with high true positive rate), AUC increases. For each table, we shade the cell of the smallest number of non XOR gates or the largest AUC value to make the comparison easier.

The followings are highlights of the comparison results:

- PRDC_woABS results in the smallest number of non XOR gates for three datasets, but it has very poor accuracy. Thus, we do not consider it to be a viable method.
- Both PRDC and MCC provide accurate object matching results. GCDC provides similarly accurate results for all four datasets and even outperforms PRDC and MCC for two datasets.
- The number of non XOR gates for GCDC is on average 261 times smaller than PRDC and 6 times smaller than MCC.

Thus, we conclude that GCDC is a promising alternative to PRDC or MCC for secure object matching. In the next section, we show how GCDC performs in a real prototype system.

7. sTrack: DESIGN & IMPLEMENTATION

We now describe our prototype of the sTrack community surveillance system. We first describe how we implement various key tasks and then how we integrate individual components into a working system.

Table 8. Summary table for the human dataset
(d=864; q=69; L=3 for PRDC; L=15 for PRDC_woABS)

human dataset	non XOR gates	AUC
PRDC	379,902	0.8473
PRDC_woABS	2,175	0.7643
MCC	10,009	0.9062
GCDC	1,729	0.9272

Table 9. Summary table for the car dataset
(d=864; q=54; L=3 for PRDC; L=4 for PRDC_woABS)

car dataset	non XOR gates	AUC
PRDC	379,902	0.8877
PRDC_woABS	579	0.8395
MCC	7,831	0.9471
GCDC	1,351	0.9567

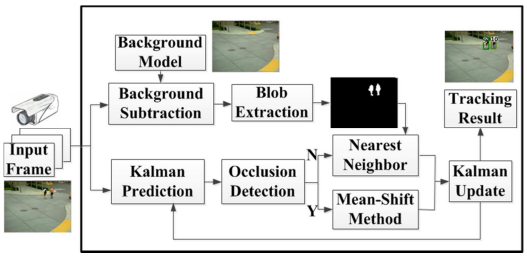

Figure 10. Tracking objects within a single camera module.

7.1. Implementation of Key Tasks

Each site (i.e., camera) in our system is configured with information about one or more peer sites against which object matching should be done. This information includes the peer's location relative to the site and how to contact it (IP address and port) when matching is needed. Each site independently keeps track of objects' information, including location, entry and exit timestamps, and feature vectors.

Tracking within a camera: We employ state of the art computer vision algorithms for background subtraction and estimating object states between frames. All of this computation involves only local processing (i.e., no information exchange needed), and thus does not affect secure computation. Figure 10 shows the flowchart of the modules that we developed for tracking objects within a single camera. For each new frame, we first perform background subtraction followed by blob extraction. The background subtraction module determines whether the pixel belongs to the background or foreground based on a statistical background model represented as a Gaussian distribution per pixel. After background subtraction, we can extract the blobs that represent the foreground objects [10]. The output is a binary image (0=background, 1=foreground).

The tracking module keeps tracking the states of the objects including their location, size, and velocity. Given the extracted blobs of a new frame, we use Kalman filter [27] to estimate the current state of each tracked object. The module then determines whether a tracked object is under occlusion

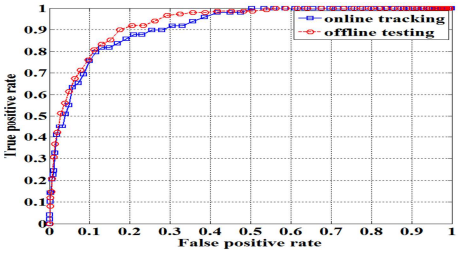

Figure 11. Online tracking performance.

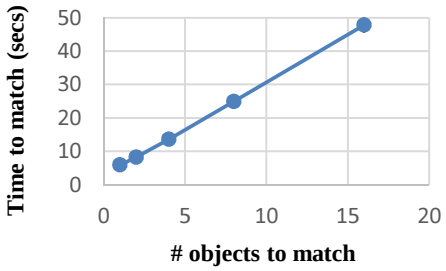

Figure 12. Mean and standard deviation of the time to securely match objects. Standard deviation is too small to show when # of objects to match is less than 10.

by checking if its predicted state is consistent with the observation in the new frame. If the object is not occluded, the result from blob extraction is deemed reliable, and the nearest blob is selected as the corresponding measurement. Otherwise, mean-shift method [5] is used to obtain the measurement. By using these measurements, we update the Kalman filter and obtain the current state of the object.

Each camera performs tracking as above and stores entry and exit time stamps, tracking history, and feature vectors of all the objects in a local database. This stored information is used when the camera performs the evaluation of the matching function between a local object and objects captured by other cameras.

Generating feature vectors: Since color information has been demonstrated to be one of the most robust features that is invariant to different scales and perspectives [31][12], we adopt the color histogram as the basis of our feature vectors. One can always include more features, e.g., texture, edge features or even biometric features like faces, to enhance the accuracy, and our proposed metric learning method can still be applied without change. As in [31], we divide an object into six horizontal regions. In each region, the color histograms of RGB, YCbCr, and HSV color spaces are extracted. Each channel has 16 bins, and the histograms are concatenated into the feature vector in a 864-dimension feature space. The feature vectors are transformed into the valid input vectors which have low dimensional integer entries before the matching function is evaluated via secure two-party computation.

Tracking object across cameras: When an object enters the view of Site A, the feature vector is extracted and the matching process is initialized against all peer sites. (We currently issue match requests for all objects entering the view, but in the future, it could be limited to suspicious objects such as those that the camera has never seen before or those specified by users.) For these processes, we term the initiator of the process (Site A) as the client and other sites as servers. Each server performs a match against each of the objects that appeared within a 10 minute time window. The length of the time window can be determined based on the user's interest or the prior knowledge about the topology of the cameras. For instance, in a community area with moderate size, it normally takes no more than 10 minutes for a people walking acorss the cameras. The output (obtained by both parties) is a binary value indicating if the client's object matched against any of the server's objects. Note that in the secure two-party computation, neither the server nor the client learns the real distance value but only a binary value representing the match or nonmatch, so the feature vectors from the other party cannot be inferred, i.e., there will be no information leakage.

This lets Site A learn where the object came from. Match requests can also be issued sometime after the object exits the camera's view, to learn where the object went. But we do not implement it yet and instead rely on incoming match requests from other peers to learn this information. Based on the results of incoming and outgoing match requests, each site can independently infer the activity pattern of suspicious objects that come under its view.

Training: As described earlier, matching requires an offline training phase to learn the distance metric. This learning is done in a pairwise manner. That is, a set of parameters is learned for each pair of cameras, rather than learning global parameters. As inputs for the learning, we use objects that the site owners have whitelisted for the purposes of learning out of a pool of all objects seen by the camera. As explained earlier, a set of around 80 objects is enough for robust learning.

7.2. System Integration

Our system is developed as an application on top of HomeOS [7]. HomeOS applications are written primarily in C#, but for performance reasons, we implemented image processing functionality (e.g., background subtraction, blob extraction, and tracking) as a C++ library. This allows us to perform these functions in real time even on weak PCs such as netbooks. The secure matching functionality is implemented in Java by extending Huang et al.'s library [14]. This code is invoked as a separate process that takes the feature vector that needs to be secretly matched as input. For each potential match, there is one process on the client side that communicates with a process on the server side. These two processes communicate with each other to determine if their respective input feature vectors match.

When the server receives a match request from the client, it starts one process per potential match. Each process is started with a feature vector that belongs to a different object and uses a different TCP port for communication. Then the server returns to the client the list of ports where each process is listening. Upon receiving this list, the client starts

one process for each port. The feature vector that is input to client-side processes is the same, corresponding to the object that triggered the matching process. The client and server read the outputs (binary values) of these processes to determine if any of the pairwise matches yielded a positive result.

This way of doing things reveals to the client how many (but not which) objects were seen by the server in the matching time window. If this information is sensitive, to hide it, the server can always initiate matches against a fixed number of objects. This fixed number should be an upper bound on the number of objects that can be seen in the matching time period. When fewer objects have been seen, the remaining processes can be supplied random feature vectors as input.

8. EXPERIMENTAL RESULTS

This section evaluates our prototype for its accuracy of tracking and the computation time of secure object matching.

8.1. Object Matching Accuracy

In Section 6, what we studied was the accuracy of matching objects across cameras assuming that object extraction in individual cameras was perfect. In practice, however, object extraction may not be perfect as extra pixels (which do not belong to the object) may be attributed to the object or some object pixels may be missing. In this section, we quantify the end-to-end accuracy of our system, which includes inaccuracies due to imperfect object extraction. For this experiment, we use the video clips from our datasets with total length 47 minutes as the input to our system. The system has two sites and each site is fed the video gathered by one of the cameras in the datasets. As in the actual system, each site performs independently single camera tracking within its own view and store objects' information (e.g., feature vectors and timestamps) in its local database. The sites perform object matching using parameters from offline training.

Figure 11 shows the matching accuracy. We see that there is only a slight loss in accuracy compared to offline testing due to errors caused by tracking within a single camera. These errors result from the cluttering background, severe occlusion, and abrupt change of the lighting condition.

8.2. Performance of Secure Matching

We now benchmark the performance of secure matching in our system. For this benchmark, we use two netbooks with 1 GHz processor and 2 GB memory that run Windows 8. We use this relatively cheap and weak computer on purpose; we expect that many camera sites will not be equipped with a powerful computer and will instead use small, headless PCs or with embedded processors like smart cameras. We configure the netbooks to use each other as peers for matching. Using a network emulator, we introduce a round-trip network delay of 100 ms between the netbooks to mimic real-world situations in which sites may not have low-latency network paths between them.

Table 10. Secure computation overhead for matching a pair of objects: Note that total is greater than the sum of individual overheads since we only report a few expensive steps.

	circuit evaluator	**circuit generator**
circuit preparation	638 (ms)	655 (ms)
OT preparation	1,442 (ms)	1,765 (ms)
circuit garbling & evaluation	765 (ms)	816 (ms)
total	3,722 (ms)	3,875 (ms)

We measure the total time from the client issuing the match request to it recovering the result of the match. Thus, this time includes the time for initial handshake in which the client learns about the ports on which the server processes are running and the time to start client and server processes. We use randomly selected feature vectors for matching. Matching performance does not depend on the values in the feature vector; it only depends on the size of the feature vector, which is independent of the objects being matched. We use $q = 128, l = 8,$ and $\hat{t} = 9280$ for the experiments. Thus, the number of non XOR gates is 3,199 using Equation (22). We conduct ten different trials with each number of objects and plot the mean the standard deviation across those trials. Figure 12 shows the results as a function of the number of objects against which a match is performed (i.e., the number of objects that the server saw in the specified time period). We see that the time it takes to securely match objects increases linearly with the number of objects that need to be matched. For single-object matches, the time is roughly 4 seconds. For matching 4-16 objects, the total time amounts to roughly 3 seconds per match. Thus, even if as many as 20 objects need to be matched, secure matching will complete in under a minute.

This level of performance, which was obtained on a relatively weak computer, can lead to a practical, real-time surveillance system. Consider the following back-of-the-envelope analysis. Let the community be such that one new object enters every X minutes on average, each camera needs to consult N neighboring, and that the time period of interest is Y minutes. In this community, a camera will issue $\frac{N}{X}$ requests per minute for matching, where each request will involve $\frac{Y}{X}$ object matches. Thus, a camera needs to perform $2 \times \frac{Y}{X} \times \frac{N}{X}$ matches per minute. The factor of two is due to the fact that a camera will answer match requests as well. With $X = 5, N = 10, Y = 10$, this number is 8 per minute, well within the performance bounds of our system.

To shed light on where time is spent during the match process, Table 10 shows the breakdown of the time to match one pair of objects. Since our secure computation protocol is built on [14], we used the same parameters as in [14] (e.g., 80-bit wire labels and $|q| = 128$ and $|p| = 128$ for the Naor-Pinkas oblivious-transfer (OT) protocol [35]). As the table shows, a large portion of the overhead (1.4 – 1.8 seconds) results from the computation intensive OT preparation

step since the netbooks only have a moderate CPU. Since [14] implements oblivious transfer extension [36], which enable a pair of sites to run any number of oblivious transfers at a small cost after preparation, we can hide this setup latency by having a pair of sites go through this OT preparation only once (instead of every object match instance as is currently implemented). We find that the computational overhead of local computation of GCDC (e.g., matrix multiplications) is negligible and so is the bandwidth overhead of secure computation. On average, machines exchanged 542 packets for secure matching.

9. DISCUSSION

This section discusses limitations of the current prototype and remaining challenges in realizing privacy-preserving distributed community surveillance systems.

Proving the security of the entire system: We assume that each site is semi-honest and pairwise object matching can be secured using secure two-party computation. However, the surveillance system as a whole involves multiple pairwise object matching tasks across many pairs of sites. Thus, our system is secure only if the secure two-party computation protocol is composable. We defer the investigation of this issue to future work.

Scaling secure object matching to a large number of peers: As shown in Section 8, the performance overhead of secure matching increases linearly in the number of objects to match. Hence, as more neighbors join the surveillance system, the overhead will increase, requiring individual sites to equip with better CPUs and more memory. However, like some computer vision works [33][34], we can potentially utilize the topology of cameras and the traveling time between the cameras to improve matching efficiency. Based on the object's time and spatial information, cameras only need to send matching requests to the neighboring cameras since the object is likely to travel to the areas covered by these cameras. Plus, given the average traveling time between two cameras, the matching time window can be adjusted accordingly. According to the recent development of the video surveillance research [44], the topology information and the average traveling time can be obtained without human efforts. This way, we can not only make community surveillance systems to scale better but also reduce false positives.

Weakness against querying attacks: Secure object matching can stop nosy neighbors from collecting the other party's input feature vectors to de-identify objects. However, it cannot prevent them from tracking the appearance of a certain object in others' camera by continuously querying other cameras with the feature vector of the target object. In an ideal setting, a participating camera should be only allowed to initiate matching for the object that is actually captured by the camera. However, in practice, this is almost impossible to ensure e.g., an adversary can launch an "analogue" attack by staging a photograph. We can guard against such nosy queries to some extent by requiring that neighbors who issue a disproportionally large number of queries share raw footage of the object if there is a match. We believe that such targeted sharing of footage of suspicious objects with other neighbors who have also seen the object poses few privacy concerns.

On combining computer vision and privacy: The most challenging aspect of our work was that it needed deep understanding of both vision algorithms and Boolean circuit construction. Individual authors were experts in at most one of these areas. One of our hopes with this work is to lower the bar for developing privacy-preserving algorithms for a range of vision-related problems. Our analysis and circuit construction of various algorithms should help privacy researchers analyze and modify other vision algorithms for secure computation and vision researchers to develop new algorithms that permit low-overhead secure computation.

10. CONCLUSION

We presented a distributed surveillance system that can track objects (e.g., human, cars) across multiple cameras without leaking any visual information about the objects other than the binary matching output. Our system is based on a new distance metric learning approach that, compared to state-of-art approaches, has 1) 6-200 times lower complexity of a Boolean circuit for secure two-party computation, and 2) similar or better tracking accuracy. Experiments using our prototype show that the performance overhead of private tracking is moderate, taking around 3 seconds even on a netbook. Distance metric learning is a powerful approach for matching that applies to a range of other domains (e.g., face recognition, matching hand-written text, and biometric matching); we hope that our work will spur future research into privacy-preserving matching techniques and systems in such domains.

11. REFERENCES

[1] S. Avidan and M. Butman, "Blind vision," *ECCV*, May, 2006.

[2] M. Bauml and R. Stiefelhagen, "Evaluation of local features for person re-identification in image sequences," *IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance*, Sep, 2011.

[3] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[4] A.J. Brush, J. Jung, R. Mahajan, and F. Martinez, "Digital Neighborhood Watch: Investigating the Sharing of Camera Data amongst Neighbors", *CSCW*, Feb, 2013.

[5] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. PAMI*, vol. 25, no. 5, pp. 564-577, May 2003.

[6] M. Dikmen, E. Akbas, T. S. Huang, and N. Ahuja, "Pedestrian recognition with a learned metric," *ACCV*, 2010.

[7] C. Dixon, R. Mahajan, S. Agarwal, A.J. Brush, B. Lee, S. Saroiu, and P. Bahl, “An operating system for the home,” *Proc. NSDI*, 2012.

[8] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, “Person re-identification by symmetry-driven accumulation of local features,” *Proc. CVPR*, 2010.

[9] A. Globerson and R. Roweis, “Metric learning by collapsing classes,” *Proc. NIPS*, 2005.

[10] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, Springer, ch.6, 2006.

[11] D. Gray, S. Brennan, and H. Tao, "Evaluating Appearance Models for Recognition, Reacquisition, and Tracking," *Proc. IEEE International Workshop on Performance Evaluation for Tracking and Surveillance (PETS)*, October, 2007.

[12] D. Gray and H. Tao, “Viewpoint invariant pedestrian recognition with an ensamble of localize features,” *Proc. ECCV*, 2008.

[13] http://www.videosurveillance.com/neighborhoods.asp. Last accessed in November 2012.

[14] Y. Huang, D. Evans, J. Katz, and L. Malka, “Faster secure two-party computation using garbled circuits,” *20th USENIX Security Symposium*, Aug, 2011.

[15] I. Jolliffe, “Principal component analysis,” John Wiley & Sons, Ltd, 2005.

[16] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, “Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima”, *Proc. CANS* 2009.

[17] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications”, *Proc. ICALP*, 2008.

[18] B. Kreuter, a. shelat, and C.-H. Shen, “Billion-Gate Secure Computation with Malicious Adversaries”, *Proc. USENIX Security*, 2012.

[19] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” *Proc. CVPR*, 2006.

[20] Y. Lindell and B. Pinkas, “A proof of security of Yao’s protocol for two-party computation”, *Journal of Cryptology*, 22(2):161–188, 2009.

[21] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, “Fairplay — A Secure Two-Party Computation System”, *Proc. USENIX Security*, 2004.

[22] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, “SCiFI-A system for secure face identification,” *IEEE Symposium on Security and Privacy*, 2010.

[23] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Y.-L. Tian, A. Ekin, J. Connell, C. F. Shu, and M. Lu, “Enabling Video Privacy through Computer Vision”, *IEEE Security and Privacy* 3, 3 (May 2005), 50-57.

[24] Bak Slawomir, Corvee Etienne, Bremond Francois, Thonnat Monique, "Boosted Human Re-identification using Riemannian Manifolds", *Image and Vision Computing, Special Issue on Manifolds for Computer Vision*, 2011.

[25] UK Home Office, “The Image Library for Intelligent Detection Systems (i-LIDS): Multiple Camera Tracking (MCT),” 2008.

[26] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C.V. Jawahar, “Efficient Privacy Preserving Video Surveillance”, *Proc. ICCV*, 2009.

[27] G. Welch and G. Bishop, “An introduction to the kalman filter,” *Technical Report, University of North Carolina at Chapel Hill*, 1995.

[28] A. C. Yao, “How to Generate and Exchange Secrets”, *Proc. FOCS*, 1986.

[29] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, “Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition,” *Proc. ICCV*, 2005.

[30] Y. Zhang and S. Li, “Gabor-LBP based region covariance desciptor for person rei-identification,” *IEEE Intl. Conf. on Image and Graphics*, pp. 368-371, 2011.

[31] W. Zheng, S. Gong, and T. Xiang, “Person re-identification by probabilistic relative distance comparison,” *Proc. CVPR* , 2011.

[32] Y. Huang, D. Evans, and J. Katz, “Private Set Intersection: Are Garbled circuits Better than Custom Protocol?”, *Proc. NDSS*, 2012.

[33] A. Gilbert and R. Bowden, “Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity,” *Proc. ECCV*, 2006.

[34] K. Chen, C. Lai, Y. Hung, and C. Chen, “An adaptive learning method for target tracking across multiple cameras,” *Proc. CVPR*, 2008.

[35] M. Naor and B. Pinkas, “Computationally Secure Oblivious Transfer”, *Journal of Cryptology*, 18(1), 2005.

[36] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, “Extending Oblivious Transfers Efficiently,” *Proc. Crypto*, 2003.

[37] M. Naor, B. Pinkas and R. Sumner. Privacy Preserving Auctions and Mechanism Design. Proc. Electronic Commerce, 1999.

[38] L. Malka, "VMCrypt: Modular Software Architecture for Scalable Secure Computation", *Proc. CCS*, 2011.

[39] W. Henecka, S. Kögl,A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "TASTY: Tool for Automating Secure Two-partY computations", *Proc. CCS*, 2010.

[40] O. Javed, K. Shafique, and M. Shah, “Appearance modeling for tracking in multiple non-overlapping cameras,” *Proc. CVPR* , 2005.

[41] S. Avidan, A. Elbaz, and T. Malkin, “Privacy preserving pattern classificaton,” *IEEE Conf. on Image Processing*, 2008.

[42] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," *Privacy Enhancing Technologies (PET)*, 2009.

[43] A. Sadeghi, T. Schneider, and I. Wehrenberg, "Effieiecnt privacy-preserving face recognition," *Information, Security and Cryptology*, 2009.

[44] C. Chu, K. Lee, and J. Hwang, "Self-organized and scalable camera networks for systematic human tracking across nonoverlapping cameras," *ICASSP*, 2013.

APPENDIX A

Denote the original high dimensional feature vector as $\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{id} \end{bmatrix}$. Since we use the histogram-based feature, each entry is a real number from zero to one. Because the principal component matrix $\mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_r]$ is an unitary matrix, the entries of the new vector after dimensional reduction

$$\hat{\mathbf{x}}_i = \begin{bmatrix} \hat{x}_{i1} \\ \vdots \\ \hat{x}_{id} \end{bmatrix} = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}}) \qquad \text{(A.1)}$$

are bounded. Assume $\|\hat{\mathbf{x}}_i\|_2 \le \theta$, and according to Cauchy-Schwarz inequality,

$$(\mathbf{w}_k{}^T \hat{\mathbf{x}}_i)^2 \le \|\mathbf{w}_k\|_2{}^2 \|\hat{\mathbf{x}}_i\|_2{}^2 \le \|\mathbf{w}_k\|_2{}^2 \times \theta^2 \qquad \text{(A.2)}$$

In order to confine the dynamic range of $\mathbf{w}_k$, we add the term $\sum_{k=1}^{q} \mathbf{w}_k{}^T \mathbf{w}_k = Tr(\mathbf{W}^T\mathbf{W})$ as another cost function in our objective function to make $\|\mathbf{w}_k\|_2{}^2$ bounded. If $\|\mathbf{w}_k\|_2{}^2$ is bounded, $(\mathbf{w}_k{}^T \hat{\mathbf{x}}_i)^2$ is bounded; that is, for all the vectors $\hat{\mathbf{x}}_i$, we can always find a nonnegative scalar T such that make $\mathbf{w}_k{}^T \hat{\mathbf{x}}_i + T \ge 0, \forall\, k$.

APPENDIX B

In order to make the distance metric function $D_4(\cdot,\cdot)$ valid, self-distance of all the given feature vectors $\hat{\mathbf{x}}_i$ should be the same.

$$\begin{aligned} D_4(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i) &= -HI(\mathbf{W}^T\hat{\mathbf{x}}_i + T1, \mathbf{W}^T\hat{\mathbf{x}}_i + T1) \\ &= -\sum_{k=1}^{q} min\,(\mathbf{w}_k{}^T\hat{\mathbf{x}}_i + T\,, \mathbf{w}_k{}^T\hat{\mathbf{x}}_i + T\,) \\ &= -\sum_{k=1}^{q} \mathbf{w}_k{}^T\hat{\mathbf{x}}_i + qT \end{aligned} \qquad \text{(B.1)}$$

If we assume $\sum_{k=1}^{q} \mathbf{w}_k{}^T\hat{\mathbf{x}}_i$ equals to zero for all i, $D_4(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k)$ is equal to $D_4(\hat{\mathbf{x}}_l, \hat{\mathbf{x}}_l)$ for all $k \ne l$. Thus, we introduce the cost function $\sum_i (\sum_{k=1}^{q} \mathbf{w}_k{}^T\hat{\mathbf{x}}_i)^2$ in the objective function.

APPENDIX C

We present the algorithm (Algorithm C.1) for constructing the $ADD^*(N, l) = \sum_{j=1}^{N} x_j$ circuit and computing the number of gates when N is not in power of 2.

1. Initial $X = N$, $t = 0$, $s = K - 1$, and *gateN* = 0.
   $N = \sum_{i=0}^{K-1} 2^{\alpha_i}$, $\log_2 N \ge \alpha_{K-1} > \cdots > \alpha_0 \ge 0$
2. **while**$(X > 1)$
3. m = 2^{α_s}
4. construct sub-circuit $ADD^*(\mathrm{m}, l)$
5. *roots[t]* = output of $ADD^*(\mathrm{m}, l)$
6. *gateN* = *gateN* + $|ADD^*(\mathrm{m}, l)|$,
   where $|ADD^*(\mathrm{m}, l)|$ is obtained by Eq.(1)
7. $X = X - \mathrm{m}$
8. $t = t + 1$
9. $s = s - 1$
10. **end while**
11. **if**$(X == 1)$
12. *roots[t]* = the remaining x_j
13. **end if**
14. Initial $r = K - 1$, *right_tree* = *roots[K − 1]*
15. **while**$(r \ge 1)$
16. Connect *roots[r-1]* and right_tree with an ADD$(l + \alpha_{K-r})$ circuit.
17. *right_tree* = output of the above ADD circuit
18. *gateN* = *gateN* + |ADD$(l + \alpha_{K-r})$|
19. $r = r - 1$
20. **end while**

Algorithm C.1. The algorithm of $ADD^*(N, l)$ when N is not in power of 2.

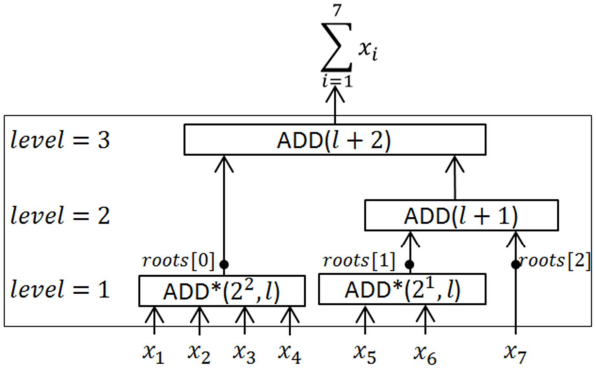

Figure C.1. Example circuit when N = 7

First, we decompose N into the summation of a set of numbers consisting of power of 2.

$K \le \lceil log_2 N \rceil$, $N = \sum_{i=0}^{K-1} 2^{\alpha_i}$, $log_2 N \ge \alpha_{K-1} > \cdots > \alpha_0 \ge 0$. For $i = 0$ to $K - 1$, we compute the summation of 2^{α_i} numbers by constructing sub-circuits $ADD^*(2^{\alpha_i}, l)$. By using $ADD(.)$, the outputs of these sub-circuits are further added up sequentially in the order from the one with smallest number of bit to the one with largest number of bit. The size of $ADD^*(N, l)$ is equal to *gateN*. Figure C.1 shows an example when $N = 7$.