Toward privacy-preserving crowd-flow estimation based on camera detections

Fatemeh Marzani, Thijs van Ede, Geert Heijenk, Maarten van Steen
University of Twente
Enschede, The Netherlands
f.marzani,t.s.vanede,geert.heijenk,m.r.vansteen@utwente.nl

ABSTRACT

An important aspect of crowd monitoring is knowing how many people we are dealing with. Sometimes, knowing the size of a crowd in a single location and at a specific moment is enough. Matters become problematic when counting the same people across different locations or counting them over longer periods of time. In those cases, we need to identify and later reidentify a person, which immediately leads to privacy concerns. Until recently, crowd-flow estimation solutions relied on unique identifiers from carry-on devices such as smartphones. However, privacy improvements, such as randomization of MAC-addresses, have made these techniques mostly ineffective. We propose a privacy-preserving crowd-flow estimation pipeline based on camera detections. In our approach, each camera processes detected faces locally, extracts facial features, and converts them into binary identifiers that preserve similarity. Different observations of the same person may still produce slightly different binary identifiers, so we apply a Sample-Then-Lock fuzzy extractor to derive the same random identifier from sufficiently similar observations. The original facial image is then deleted. These identifiers are inserted into Bloom filters and encrypted, allowing set cardinality to be computed obliviously and directly on encrypted data, enabling the system to count people across locations and over time without revealing any identities. Our initial evaluation on the SCface dataset shows that the fuzzy extractor achieves a true-acceptance rate of 0.90, with modest computational overheads: 265.9 ms for identifier generation, 442.8 ms for reproduction, and 18.7 ms to encrypt a single Bloom filter. These results demonstrate that the pipeline is feasible for privacy-preserving crowd-flow estimation.

KEYWORDS

Privacy-preserving analytics; Crowd monitoring; Smart cities; Fuzzy Extractors; Fully Homomorphic Encryption (FHE)

1 INTRODUCTION

Data collected on crowd behavior supports applications such as urban planning [1], tourism management [2], and public safety and security [3]. These applications often require more than a count of people in a single location. They also need crowd-flow information that captures how many individuals move between locations or reappear over time. In that case, the system must determine whether two observations correspond to the same person. This requires identifying and later re-identifying a person, which immediately raises privacy concerns: re-identification entails storing and linking information that can single out individuals, enabling large-scale tracking and unauthorized profiling, and raising concerns under privacy regulations such as the EU General Data Protection Regulation [4] (GDPR). Existing privacy-preserving crowd-flow estimation systems typically rely on one central assumption: each individual can be represented by a unique and stable identifier that is observed again whenever the same person reappears. Until recently, this assumption was reasonable for identifiers broadcast by carry-on devices, most notably the fixed MAC addresses transmitted by smartphones in WiFi probe requests. Stanciu et al. [5] showed that such identifiers can support privacy-preserving counting by encoding observation sets as Bloom filters and estimating set intersections under homomorphic encryption, without revealing the underlying identifiers in plaintext. Their work demonstrated that accurate crowd-flow estimation is possible when stable person-specific identifiers are available. However, this assumption no longer holds in many practical deployments. Modern mobile operating systems now use MAC-address randomization, which prevents repeated observations of the same device from being reliably linked. As a result, carry-on devices can no longer be treated as a dependable source of stable identifiers for crowd-flow estimation.

Camera-based detections provide an alternative source of observations, but using facial data directly creates serious privacy risks. The system must count how many people appear in one or more observation sets without revealing their facial data. This introduces two problems that must both be addressed. First, the observed facial data must be protected. Facial images and embeddings should remain local to the camera, be used only to derive an identifier, and be discarded immediately after processing. Second, different facial observations of the same individual must produce the same global identifier, despite variations in pose, illumination, expression, occlusion, image quality, and camera viewpoint.

Existing approaches therefore leave an important gap. Face-recognition systems can match different observations of the same person, but they expose sensitive and linkable biometric data when they store or compare facial embeddings. Privacy-preserving crowd-flow systems protect aggregate computations, but they assume that repeated observations produce exactly the same identifier. We address this gap with a privacy-preserving framework for crowd-flow estimation from camera detections. In this framework, each camera first converts facial embeddings into similarity-preserving local identifiers, so observations of the same person remain close in distance while observations of different people remain separated. Because repeated observations may still produce local identifiers, fuzzy extractor derives the same identifier from sufficiently similar representations. This process keeps facial images, embeddings, and local identifiers inside the camera. The camera then inserts the identifiers into Bloom filters and encrypts them, allowing the server to estimate footfall and crowd-flow cardinalities without accessing biometric data or plaintext identifiers. We evaluate the

framework on the SCface dataset and analyze its accuracy, security, and computational and storage overhead. With three reference samples per identity, With our approach, 90% of repeated observations of the same person are mapped to the same global identifier. To the best of our knowledge, this is the first framework that supports privacy-preserving crowd-flow estimation from facial observations across cameras and time.

The remainder of this paper develops and evaluates this contribution. Section 2 reviews related work and positions our framework with respect to privacy-preserving crowd monitoring and biometric processing. Section 3 defines the system model and the privacy and security requirements that guide the design. Section 4 presents the proposed framework, including stable identifier generation and encrypted crowd-flow computation. Section 5 evaluates the framework in terms of accuracy, security, and computational and storage overhead. Section 6 discusses the limitations and directions for future work, and Section 7 concludes the paper. The source code for evaluating our methods is publicly available.[1]

2 RELATED WORK

Footfall Estimation from Visual Data. Early camera-based crowd-monitoring systems primarily focused on footfall estimation, that is, counting the number of people present at a single location without linking observations across locations or time. Chan et al. [6] estimate crowd density directly from visual features, avoiding explicit detection and tracking while reporting only aggregate density for a single viewpoint. The Cerberus system [7] follows an edge-computing approach in which visual analytics are performed locally on the device and only aggregate footfall counts are exported. Chen et al. [8] use federated learning to train a crowd-counting model without sharing raw image data between participants. These systems limit the disclosure of visual information, but they address single-location counting rather than crowd-flow estimation. They therefore do not provide a mechanism for determining how many individuals move between cameras or reappear across different time intervals.

Privacy-Preserving Face Processing. A separate line of work studies privacy-preserving computation over facial biometric data. Ren et al. [9] present RAPOO, a privacy-preserving facial-expression recognition framework based on secure multiparty computation, showing that facial features can be processed without revealing them to any individual party. Ahmed et al. [10] combine facial and voice biometrics for crowd-surveillance applications, but focus on identifying individuals rather than producing aggregate movement statistics. These works demonstrate that privacy-preserving processing of facial data is possible, but their goal is recognition or classification. They do not address settings involving multiple cameras and time spans, in which observations must be linked only to estimate aggregate crowd flows.

Crowd-Flow Estimation with Stable Identifiers. The closest prior work on privacy-preserving crowd-flow estimation uses stable identifiers as set elements and protects them with Bloom filters and homomorphic encryption. Stanciu et al. [5] introduce a framework

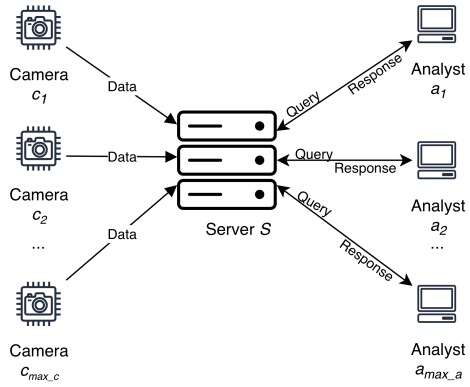


Figure 1: Service model.

in which observation sets are encoded as Bloom filters and processed under homomorphic encryption, allowing a server to estimate the size of intersections without learning the underlying identifiers. Marzani et al. [11] extend this approach for sparsely populated regions by adding controlled uncertainty through detection sampling, reducing the risk of inferring individual movements when crowd density is low. Formal differential-privacy guarantees for low-density crowd-monitoring regions are studied separately in [12].

These approaches show how to protect crowd-flow computation once stable identifiers are available. However, they rely on the assumption that each observation can be mapped to the same identifier whenever the same person is observed again. In the original setting, such identifiers were derived from fixed MAC addresses broadcast by smartphones during WiFi probe requests. This assumption has become fragile because modern operating systems randomize MAC addresses by default. More importantly for our setting, these methods cannot tolerate noisy biometric observations: two slightly different bit strings are treated as two different individuals. As a result, they cannot be directly applied to face-based crowd monitoring, where embedding variation caused by pose, illumination, occlusion, image quality, and camera viewpoint is unavoidable.

Positioning of the Work. Privacy-preserving footfall counting from camera data is feasible, but how to preserve privacy in the case of crowd-flow estimation from faces remains an open research question. Existing crowd-flow systems protect the computation once stable identifiers are available, but they do not explain how such identifiers can be obtained from noisy biometric observations in the first place. This is precisely the gap our work addresses: we derive stable identifiers directly from noisy face-derived representations and use them as protected set elements for aggregate crowd-flow estimation. To the best of our knowledge, we present the first privacy-preserving crowd-flow-estimation pipeline performed from face detections.

3 SYSTEM MODEL

We consider a crowd-monitoring system in a public space with multiple surveillance cameras. The objective of the system is to estimate

[1] https://anonymous.4open.science/r/fuzzyExtractorToBF-C67E/README.md

pedestrian footfall and crowd flows while preventing the disclosure of biometric information related to observed individuals. We start by introducing the camera-based crowd-monitoring environment and the representation of observations within the system. We then model the insights offered by the system as statistical counts for pedestrian footfall and crowd-flow estimation. Finally, we describe the entities involved in the process and define the accuracy, privacy, and security requirements that must be satisfied throughout the system.

3.1 Crowd-Monitoring Environment

We consider a crowd-monitoring environment consisting of a set of surveillance cameras $C = c_1, c_2, \ldots, c_n$ deployed across a monitored area. The cameras continuously observe individuals moving through their fields of view and collect observations that can later be used to estimate pedestrian footfall and crowd flows. Time is divided into a sequence of discrete epochs $E = e_1, e_2, \ldots, e_m$, where each epoch corresponds to a predefined observation interval. The epochs are ordered in time such that e_m occurs after e_{m-1}. During an epoch e, a camera c may observe multiple individuals within its field of view. For each observed individual, the camera extracts a face embedding from the captured facial image. Repeated observations of the same person may produce different embeddings. Therefore, an embedding cannot directly serve as a unique identifier.

Each camera converts the embedding into a similarity-preserving local identifier. Local identifiers from the same person remain close in Hamming distance, but they may not be identical. The system therefore cannot use them directly as a set element. Instead, it applies a fuzzy extractor, illustrated in Figure 2, to derive a stable identifier that is reproduced from sufficiently similar local identifiers. This global identifier provides the exact consistency required for set-cardinality and intersection computations across cameras and epochs. Let $D_{c,e}$ denote the set of global identifiers corresponding to all individuals observed by camera c during epoch e. The collection of sets $\{D_{c,e} \mid c \in C, e \in E\}$ forms the basis for all crowd-monitoring statistics supported by the system.

3.2 Statistical Counts for Pedestrian Dynamics

Observations made by cameras can be used to derive various statistics describing pedestrian dynamics. In this work, we focus on two fundamental crowd-monitoring situations:

(1) The crowd of individuals present at a particular location during a given period of time, referred to as *footfall*.
(2) The movement of individuals between locations over time, referred to as *crowd flow*.

We formally define them in terms of counts over observation sets.

Definition 3.1 (Footfall). Let $D_{c,e}$ denote the set of unique global identifiers generated from observations made by camera c during epoch e. We define the footfall associated with camera c during epoch e as the count obtained by computing $|D_{c,e}|$. The footfall represents the number of unique individuals observed by camera c during the corresponding epoch.

Definition 3.2 (Crowd Flow). Let D_{c_1,e_1} and D_{c_2,e_2} be observation sets generated by cameras c_1 and c_2. We define the crowd flow from (c_1, e_1) to (c_2, e_2) as $\left|D_{c_1,e_1} \cap D_{c_2,e_2}\right|$. This value represents the number of individuals observed by camera c_1 during epoch e_1 and later observed by camera c_2 during epoch e_2.

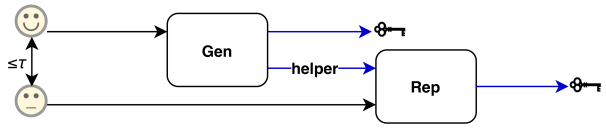


Figure 2: Fuzzy-extractor operation. Gen produces a random key and public helper data from b, while Rep reproduces the same key from b' when $\mathrm{dist}(b, b') \leq \tau$.

A crowd-monitoring system concerned with pedestrian dynamics should support the computation of both footfall and crowd-flow statistics.

3.3 Service Model

Let us now model, from an architectural point of view, the entities taking part in the process. We distinguish three types of entities involved in the system. Figure 1 shows the entities involved in the system.

Cameras. Cameras observe individuals within their fields of view and perform all biometric processing locally. For each detected face, a camera extracts a face embedding, generates a local identifier, and assigns a global identifier. Although the global identifier does not directly reveal the individual's real-world identity, it acts as a persistent pseudonym that can link observations across cameras and epochs. Cameras therefore do not share plaintext identifier sets. Instead, they encode each observation set as a Bloom filter, encrypt it, and send only the protected representation to the server.

Server. This entity stores protected data received from cameras and processes crowd-monitoring queries. Upon receiving a query, the server assembles the required data and performs the corresponding computations. The server operates only on protected representations and does not require access to biometric information in plaintext.

Analysts. These are entities interested in pedestrian-dynamics insights. Analysts submit footfall and crowd flow queries. They need to process received responses in order to find out the desired statistical counts. .

3.4 Security and Privacy Requirements

We assume that cameras operate correctly and follow the prescribed protocol. Cameras are trusted to perform local processing of biometric observations, including face-embedding extraction, local-identifier generation, and global-identifier assignment. Cameras discard raw facial images, face embeddings, and local identifiers at the end of each epoch. The server is assumed to be *honest-but-curious*; it correctly executes the protocol but may attempt to infer information from the protected data it stores and processes. Authorized analysts are trusted to issue legitimate crowd-monitoring queries but should not obtain information beyond the aggregate

statistics returned by the system. Finally, we assume that the server does not collude with cameras or analysts.

Under these assumptions, the system must satisfy the following requirements.

Accuracy. The estimated footfall and crowd-flow counts should closely match the true number of individuals observed within the monitored environment.

Irreversibility. A helper record is public auxiliary information that allows the fuzzy extractor to reproduce a stable identifier from a sufficiently similar observation. An adversary who gains access to identifiers, or protected data structures should not be able to reconstruct the original facial image, face embedding, or local identifier.

Server Obliviousness. The server should learn neither biometric information nor individual identifiers and should process only protected data necessary for query execution.

Limited Disclosure. Authorized analysts should learn only the aggregate counts corresponding to the queries they submit and should not be able to infer information about specific individuals or their movements

4 OUR CONSTRUCTION

We now describe how the proposed system derives crowd-monitoring statistics from facial observations while satisfying the requirements defined in Section 3.4. Figure 3 provides an overview of main steps, which the following subsections describe in detail.

4.1 Face Detection

For each epoch, cameras continuously capture images within their field of view and detect the presence of individuals. Face detection is performed locally at the camera, and only detected facial regions are processed further. Raw images are discarded after processing and are never transmitted outside the camera.

Each detected face is subsequently transformed into a face embedding that serves as the basis for identifier generation.

4.2 Face Embedding Extraction

For each detected face, a camera extracts a face embedding using a pre-trained face-recognition model. Let $v \in \mathbb{R}^d$ denote the resulting embedding vector, where d is the embedding dimension. We use pre-trained AdaFace model with an IR-101 backbone [13] for this purpose. Face embeddings provide a compact numerical representation of facial characteristics and enable similarity comparisons between observations. Embeddings obtained from the same individual are generally close in the feature space but are not necessarily identical. Consequently, embeddings cannot be directly used as identifiers, since repeated observations of the same individual may produce different embedding vectors. Therefore, embeddings must be transformed into a more robust representation before they can be used for crowd-monitoring statistics.

4.3 Similarity-Preserving Identifier Generation

The purpose of local identifiers is to provide a stable representation of face embeddings that preserves the similarity structure of the underlying embedding space. In particular, observations originating from the same individual should produce local identifiers with a small Hamming distance, whereas observations originating from different individuals should produce local identifiers with a large Hamming distance. Together, these properties ensure that local identifiers correctly reflect the identity structure present in the original embedding space. The generation of local identifiers does not require information exchange between cameras. Each camera applies the same deterministic function independently to its observed embeddings. Formally, given a face embedding $v \in \mathbb{R}^d$, similarity-preserving binarization produces a local identifier b.

Given an embedding vector v, we compute its mean value and use it as a threshold. Each component of the embedding is converted into a binary value according to whether it exceeds the mean. The resulting binary vector $b \in 0, 1^d$ is defined as

$$b_i = \begin{cases} 1, & \text{if } v_i > \mu(v), \\ 0, & \text{otherwise,} \end{cases}$$

where $\mu(v)$ denotes the mean of the embedding components.

This transformation converts each embedding into a compact representation while preserving the relative similarity between observations. As a result, observations of the same individual are expected to produce binary vectors that differ only in a limited number of positions. The global-identifier generation stage handles the remaining differences between local identifiers.

4.4 Global Identifier Generation

Although local identifiers preserve the similarity between observations, they do not guarantee global consistency. Observations originating from the same individual may still produce different local identifiers due to variations in the underlying face embeddings. Consequently, local identifiers cannot be used directly to construct observation sets, as the same individual could contribute multiple distinct elements to the system.

To support crowd-flow estimation, the system must assign the same identifier to every observation of the same person, regardless of the camera or observation time. This requires the generation of global identifiers. Unlike local-identifier generation, achieving global identifiers cannot be performed independently by each camera. Instead, cameras must exchange auxiliary information that enables them to determine whether a newly generated local identifier corresponds to a global identifier that was previously generated elsewhere in the deployment.

To satisfy these requirements, we employ a fuzzy extractor. A fuzzy extractor enables cameras to derive the same global identifier from sufficiently similar local identifiers while revealing negligible information about the underlying biometric representation.

A fuzzy extractor consists of two algorithms: a generation function $\text{Gen}(b) \to (R, P)$ and a reproduction function $\text{Rep}(b', P) \to R$, where $b, b' \in \{0,1\}^d$ are local identifiers. Given b, Gen samples a uniformly random global identifier $R \in \{0,1\}^\lambda$ and produces a public helper record P. If a later observation produces a sufficiently similar local identifier b', Rep uses P to reproduce the same identifier R. An important security requirement of fuzzy extractors is that the identifier R should remain computationally indistinguishable from a uniformly random value given P.

We employ the Sample-Then-Lock construction [14] as our fuzzy extractor. Traditional fuzzy extractors based on error-correction

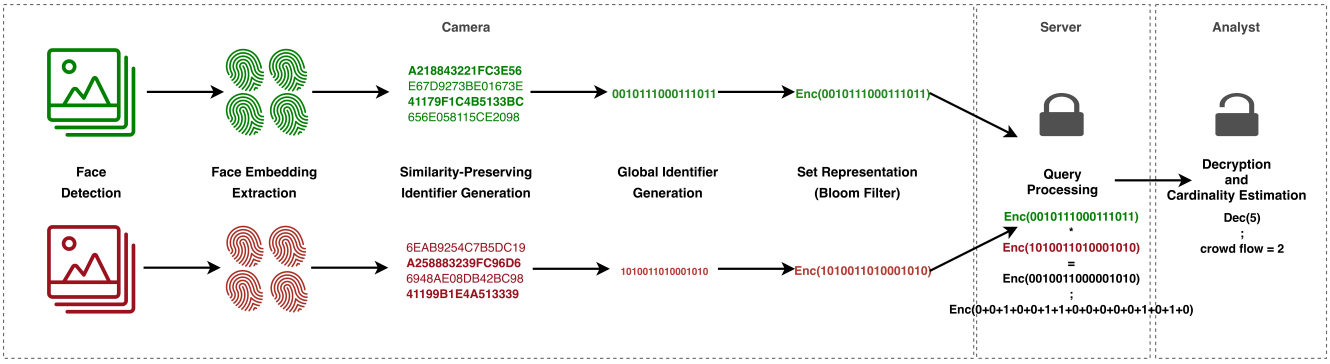


Figure 3: Privacy-preserving crowd-flow estimation pipeline, from local face processing on the cameras to encrypted query processing on the server and aggregate cardinality estimation by the authorized analyst. In this example, the BF parameters are $m = 16$ and $k = 3$, yielding $\hat{n} \approx 2$, corresponding to a crowd flow of two.

codes store correction information in the helper record that is derived directly from b, which consumes entropy in proportion to the error rate. For face embeddings, where bits are correlated and inter-observation variation is non-negligible, this leaves no security margin [15]. Sample-Then-Lock addresses this limitation by replacing global error-correction information with a collection of digital lockers. Rather than storing information that enables correction of errors in the biometric representation, each locker protects the same randomly generated identifier using a different subset of bit positions. As a result, the helper record reveals only the selected subsets and locker parameters, but not the enrolled bit values themselves. The security of the construction is determined by the lowest entropy among the subsets used by the lockers. Therefore, the identifier remains computationally indistinguishable from a random value as long as every selected subset has sufficiently high min-entropy, even when the helper record is fully exposed.

During identifier generation, each locker i uses a subset $I_i \subseteq \{1, \ldots, d\}$ of s distinct bit positions, sampled uniformly without replacement from the d positions of the local identifier b. We write b_{I_i} for the bits of b at these positions. The camera samples a fresh hash key h_i and computes $z_i = \mathrm{HMAC}(h_i, b_{I_i})$. It then constructs the locker value as $p_i = (0^{128} \| R) \oplus z_i$.

All lockers associated with the same local identifier protect the same identifier R, but each locker uses a different subset of bit positions. Together, the lockers form the helper record. During reproduction, the camera computes $z'_i = \mathrm{HMAC}(h_i, b'_{I_i})$ for each locker and evaluates $y_i = z'_i \oplus p_i$. If b' matches b at every position in I_i, then $z'_i = z_i$ and $y_i = 0^{128} \| R$.

The camera detects successful unlocking by checking whether the first 128 bits of y_i are zero. This fixed prefix acts as a recognition value. If the check succeeds, the camera extracts R from the remaining bits. In this way, cameras can reproduce the same global identifier without exchanging the identifier itself.

Reproduction succeeds if at least one locker uses a subset containing none of the positions on which b and b' differ. A camera can increase the probability of successful reproduction by using multiple face observations of the same individual rather than relying on a single observation. While an individual remains in view, the camera may capture several frames and derive one local identifier from each face observation. During generation, the first local identifier is used to generate the global identifier R and a helper record. Additional local identifiers are used to create further helper records that protect the same R. During reproduction, the camera may similarly derive local identifiers from multiple face observations and test them against the available helper records. The camera reproduces R when any of these attempts opens at least one locker. Using multiple observations is a standard technique for improving the true-acceptance rate of biometric fuzzy extractors [16].

4.5 Helper-Record Exchange Design

The fuzzy extractor specifies how a helper record P enables a sufficiently similar local identifier to reproduce the same identifier R. However, it does not specify which cameras receive P or how they access previously generated helper records. This distribution determines the scope of the shared identifier space and the queries that can be answered from the stored observation sets. We consider three designs.

Design 1: Pairwise camera-epoch exchange. In the pairwise design, cameras exchange helper records only for the camera-epoch sets involved in a specific query. For example, to estimate the flow from (c_1, e_1) to (c_2, e_2), camera c_2 uses the helper records generated at (c_1, e_1) when processing its observations during e_2. This creates an identifier space shared by the two observation sets and allows the system to compute $\left|D_{c_1,e_1} \cap D_{c_2,e_2}\right|$.

The main advantage is that cameras exchange and process only the helper records needed for the selected query. This design can reduce computation when the system supports only a small number of predefined queries.

However, the resulting identifiers are specific to the selected camera-epoch combination. The encrypted Bloom filters created for one query cannot necessarily be reused for another query involving different cameras or epochs. Each new query may therefore require another helper-record exchange, another identifier-reproduction step, and new encrypted Bloom filters. Because cameras discard local identifiers after each epoch, the system must determine these queries in advance. The design can support a multiway query such

as $\left|D_{c_1,e_1} \cap D_{c_2,e_2} \cap D_{c_3,e_3}\right|$, but all participating camera-epoch sets must use the same query-specific identifier space.

Design 2: Shared helper storage. In the shared-storage design, each camera maintains a pool of helper records received from the other cameras. For each new observation, the camera derives a local identifier b and tests it against the stored helper records. If a helper record opens, the camera reproduces the corresponding global identifier R. Otherwise, it generates a fresh pair (R, P) and shares the new helper record P with the other cameras.

This design creates one identifier space shared across all cameras and epochs. Each camera therefore constructs only one encrypted Bloom filter per epoch. The system can reuse these encrypted Bloom filters for pairwise and multiway queries, including queries that were not specified in advance.

The disadvantage is that each camera's helper-record pool grows over time. A camera may need to test every new local identifier against many stored helper records, increasing both storage and reproduction time. Compared with Design 1, this design requires more camera-side processing but provides greater query flexibility. In the remainder of this paper, we adopt the shared-helper-storage design.

Design 3: Centralized identifier service. In this design, cameras send each local identifier b to a central identifier service. The service tests b against its stored helper records and returns the corresponding global identifier R. If no helper record opens, it generates a new pair (R, P) and stores the helper record.

This design creates a common identifier space across all cameras and reduces camera-side storage and computation because the identifier service performs the helper-record search. However, it receives biometric-derived local identifiers in plaintext and learns the global identifiers assigned to observations. It can therefore link observations across cameras and epochs. For this reason, we reject this design despite its lower camera-side overhead.

4.6 Observation Set Representation

During each epoch e, camera c collects the global identifiers of all observed individuals in the observation set $D_{c,e} = \{R_1, R_2, \ldots, R_n\}$. The camera removes duplicate identifiers so that each individual contributes only once. The camera represents $D_{c,e}$ as a Bloom filter $BF_{c,e}$ of length m using k hash functions. Bloom filters provide a compact probabilistic representation of sets and support efficient cardinality and intersection estimation through simple bit operations. These operations are particularly suitable for homomorphic encryption, because the server can evaluate them directly over encrypted Bloom-filter bits. Let $t_{c,e}$ denote the number of bits set to one in $BF_{c,e}$. The cardinality of the observation set can be estimated as $\widehat{N}_{c,e} = -\frac{m}{k} \ln\left(1 - \frac{t_{c,e}}{m}\right)$ [17].

Given two Bloom filters BF_{c_1} and BF_{c_2}, their intersection can be approximated using $BF_{c_1 \cap c_2} \approx BF_{c_1} \wedge BF_{c_2}$, allowing the size of the corresponding observation-set intersection to be estimated.

4.7 Homomorphic Encryption

Although Bloom filters do not reveal the stored identifiers directly, a plaintext Bloom filter can still leak information about the represented observation set. For example, a party that knows a candidate global identifier R^* can compute its k hash positions and test whether all corresponding bits are set: $\bigwedge_{j=1}^{k} BF_{c,e}\left[h_j(R^*)\right] = 1$. A positive result indicates that R^* may belong to the observation set $D_{c,e}$, subject to the false-positive probability of the Bloom filter. Repeating this test across cameras and epochs could reveal where and when the same identifier appeared.

To prevent such membership tests, each camera encrypts its Bloom filter using the analyst's public key before sending it to the server. Let $Enc(BF_{c,e})$ denote the encrypted Bloom filter associated with camera c and epoch e. At the end of each epoch, the camera sends only $Enc(BF_{c,e})$ to the server and discards the plaintext Bloom filter.

The server stores the encrypted Bloom filters and uses them to process footfall and crowd-flow queries. Since fully homomorphic encryption supports computations over ciphertexts [18], the server can perform set operations without learning the contents of the underlying Bloom filters. Given two encrypted Bloom filters representing the observation sets of cameras A and B, the server computes $Enc(t) = \sum_{i=1}^{m} Enc(BF_A[i]) \cdot Enc(BF_B[i])$, where multiplication emulates a bitwise AND operation and addition counts the number of set bits in the resulting intersection filter. The resulting ciphertext $Enc(t)$ represents the encrypted number of bits contained in the Bloom-filter intersection.

The server sends only $Enc(t)$ to the analyst, not the underlying encrypted Bloom filters. The analyst decrypts $Enc(t)$ using the corresponding secret key and estimates the intersection cardinality. Therefore, the server cannot perform membership tests because it cannot decrypt the Bloom filters, while the analyst cannot perform them because it never receives the Bloom filters.

4.8 Query Processing

When an analyst submits a footfall or crowd-flow query, the server retrieves the corresponding encrypted Bloom filters and performs the required computation homomorphically.

For a footfall query concerning camera c during epoch e, the server retrieves $(BF_{c,e})$ and homomorphically sums its encrypted bits. This produces $(t_{c,e})$, where $t_{c,e}$ is the number of set bits in the Bloom filter.

For a crowd-flow query involving a collection of (camera, epoch) pairs $\{(c_i, e_i)\}_{i=1}^{n}$, the server retrieves the corresponding encrypted Bloom filters and computes their encrypted intersection. It then homomorphically sums the bits of the resulting intersection filter.

In both cases, the server sends only the encrypted set-bit count to the analyst. It does not send the encrypted Bloom filters or any intermediate intersection representation. The analyst decrypts the returned value and applies the corresponding Bloom-filter cardinality estimator. For a footfall query, this estimates $|D_{c,e}|$; for a crowd-flow query, it estimates $\left|\bigcap_{i=1}^{n} D_{c_i,e_i}\right|$.

Consequently, the server performs the query without learning the Bloom-filter contents or the resulting count, while the analyst receives only the aggregate information required to estimate the requested footfall or crowd flow.

Algorithm 1: Sample-Then-Lock Fuzzy Extractor

function GEN(b, $\mathcal{I} = \{I_1, \ldots, I_V\}$)
  Sample a random 128-bit identifier R
  for $i = 1$ to V **do**
    Sample a 512-bit HMAC key h_i
    $x_i \leftarrow b_{I_i}$
    $z_i \leftarrow \mathrm{HMAC}(h_i, x_i)$
    $p_i \leftarrow (0^{128} \| R) \oplus z_i$
  end for
  $P \leftarrow \{(p_i, h_i)\}_{i=1}^{V}$
  return (R, P)
end function

function REP(b', P, $\mathcal{I}$)
  for $i = 1$ to V **do**
    $x'_i \leftarrow b'_{I_i}$
    $z'_i \leftarrow \mathrm{HMAC}(h_i, x'_i)$
    $y_i \leftarrow z'_i \oplus p_i$
    if $(y_i)_{1\ldots128} = 0^{128}$ **then**
      return $(y_i)_{129\ldots256}$
    end if
  end for
  return $\perp$
end function

Algorithm 2: Camera-Side Processing

Require: Public key pk, helper records $\mathcal{P}$, subsets $stats$
for each camera c **do**
  $D_{c,e} \leftarrow \emptyset$
  for each detected face **do**
    Extract embedding v
    Binarize v into b
    $R \leftarrow \perp$
    for each helper record $P \in \mathcal{P}$ **do**
      $R \leftarrow \mathrm{REP}(b, P, stats)$
      if $R \neq \perp$ **then**
        break
      end if
    end for
    if $R = \perp$ **then**
      $(R, P) \leftarrow \mathrm{GEN}(b, stats)$
      Share helper record P
    end if
    Insert R into $D_{c,e}$
  end for
  Remove duplicates from $D_{c,e}$
  Encode $D_{c,e}$ as $BF_{c,e}$
  $Enc(BF_{c,e}) \leftarrow Enc_{pk}(BF_{c,e})$
  Send $Enc(BF_{c,e})$ to server
  Discard temporary plaintext data
end for

Algorithm 3: Server-Side Query Processing

Require: Encrypted Bloom filters, query q
if q is a footfall query for (c, e) **then**
  Retrieve $Enc(BF_{c,e})$
  Compute encrypted bit count $Enc(t)$
else if q is a crowd-flow query over $\{(c_i, e_i)\}_{i=1}^{n}$ **then**
  Retrieve $\{Enc(BF_{c_i,e_i})\}_{i=1}^{n}$
  Compute encrypted intersection $Enc(I)$
  Compute encrypted bit count $Enc(t)$ over $Enc(I)$
end if
Send $Enc(t)$ to analyst

Algorithm 4: Analyst-Side Count Estimation

Require: Secret key sk, encrypted count $Enc(t)$, parameters m, k
$t \leftarrow Dec_{sk}(Enc(t))$
$\hat{c} \leftarrow -\frac{m}{k} \ln\left(1 - \frac{t}{m}\right)$
return $\hat{c}$

Figure 4: Overview of the proposed privacy-preserving crowd-monitoring workflow.

5 EVALUATION

We evaluate the proposed system along three dimensions: counting accuracy, security, and computational and storage overhead. The accuracy evaluation covers the complete pipeline from face-embedding separability and binary local-identifier quality to Sample-Then-Lock reproduction and Bloom-filter intersection estimation.

The security analysis considers the helper records, global identifiers, encrypted Bloom filters, and query outputs. We examine whether helper records protect the underlying biometric representations and global identifiers, whether homomorphic encryption hides the Bloom-filter contents from the server, and whether analysts receive only aggregate query results.

The overhead evaluation measures Sample-Then-Lock generation and reproduction, Bloom-filter encryption, server-side homomorphic computation, and the storage required for helper records and encrypted Bloom filters. We do not include face-detection and embedding-extraction costs, as these operations rely on an off-the-shelf face-recognition pipeline and are outside the scope of the current prototype evaluation.

Experimental setup. We evaluate the pipeline on the SCface dataset [19], which contains 130 individuals, each captured in nine images under different camera distances, lighting conditions, and acquisition settings. This gives 1,170 face images in total. Each image is treated as one camera observation, corresponding to one detected face in one frame.

For each image, we extract a 512-dimensional face embedding using an AdaFace backbone with an IR-101 architecture. The resulting embedding is ℓ_2-normalized, producing one vector $v \in \mathbb{R}^{512}$ per image. We convert each embedding into a local identifier using the deterministic per-embedding mean-threshold binarization method described in Section 4.3. This produces a binary local identifier $b \in \{0, 1\}^{512}$. The same binarization function is used for all cameras and epochs, so local-identifier generation does not require camera-specific training or inter-camera coordination.

Sample-Then-Lock [16] uses $V = 250{,}000$ lockers, binary templates of dimension $d = 512$, and a 128-bit global identifier R. Each locker uses a subset of s bit positions from the binary local identifier. We vary the subset size from $s = 60$ to $s = 100$ to examine the central accuracy–security trade-off in Sample-Then-Lock. Smaller subsets increase the probability that two observations of the same person match on at least one locker, whereas larger subsets increase the difficulty of guessing the locker input. Shukla et al. [16] analyzed this trade-off for iris features; we evaluate it here for facial embeddings. For the encrypted Bloom-filter stage, we keep

the Bloom-filter parameters fixed throughout the evaluation, using capacity $n = 1000$ and false-positive rate $p = 0.01$, which gives $m = 9585$ bits and $k = 7$ hash functions. These parameters are fixed because Bloom-filter-based crowd-flow estimation has already been studied in prior work [5]; our evaluation therefore focuses on deriving stable global identifiers from noisy face observations before encrypted intersection-cardinality estimation.

5.1 Accuracy

The accuracy evaluation covers three stages of the pipeline. We first measure identity separation before and after binarization. We then evaluate whether Sample-Then-Lock reproduces the correct global identifier. Finally, we measure the accuracy of the complete crowd-flow estimation pipeline.

Embedding and local-identifier separability. We first evaluate whether binarization preserves the identity separation provided by the original face embeddings. Using all 1,170 images, we compute 4,680 intra-person comparisons and 679,185 inter-person comparisons.

Figure 5a shows the cosine-similarity distributions of the original embeddings. Intra-person comparisons have a mean similarity of 0.734, whereas inter-person comparisons have a mean similarity of 0.038. The embeddings therefore provide strong separation between same-person and different-person observations.

Figure 5b shows the corresponding normalized Hamming-distance distributions after binarization. Intra-person pairs have a mean distance of 0.233, whereas inter-person pairs have a mean distance of 0.488. Binary local identifiers belonging to the same person therefore remain substantially closer than identifiers belonging to different people. We quantify this separation using the area under the ROC curve. For cosine similarity, larger values indicate a stronger same-person match, so we compute $\mathrm{AUC}_{\cos} = \Pr[s_{\text{same}} > s_{\text{diff}}]$. For Hamming distance, smaller values indicate a stronger same-person match, so we compute $\mathrm{AUC}_{\mathrm{Ham}} = \Pr[d_{\text{same}} < d_{\text{diff}}]$. The cosine-similarity AUC is 0.9960, while the normalized Hamming-distance AUC is 0.9958. Mean-threshold binarization therefore preserves most of the identity separation present in the embedding space.

Sample-Then-Lock reproducibility. We evaluate whether Sample-Then-Lock reproduces the correct global identifier from repeated face observations. Let f denote the number of face observations captured for each person during generation. Each observation produces one local identifier and one helper record containing V lockers. All f helper records protect the same global identifier R. Therefore, each person is represented by fV lockers in total.

A genuine reproduction attempt succeeds when a later observation of the same person opens at least one of the f helper records and reproduces R. The true-acceptance rate (TAR) is the fraction of same-person reproduction attempts that succeed.

A false acceptance occurs when an observation of a different person opens at least one helper record and reproduces the protected global identifier. The false-acceptance rate (FAR) is the fraction of different-person reproduction attempts that succeed.

Table 1 reports TAR for $f = 1$, $f = 2$, and $f = 3$. Smaller subsets are more likely to avoid the positions that differ between two observations of the same person and therefore achieve a higher TAR. Increasing s from 60 to 100 reduces TAR for $f = 3$ from 0.9077 to 0.1231.

Table 1: Effect of subset size on Sample-Then-Lock reproducibility. Here, f denotes the number of face observations captured per person during generation. Each later observation is tested against all f helper records, containing fV lockers in total. FAR was zero for all evaluated values of f.

Subset size s	TAR (# face observations f)			FAR
	1	2	3	
60	0.1538	0.6923	0.9077	0
65	0.1154	0.5846	0.7692	0
70	0.0615	0.4308	0.7000	0
75	0.0615	0.2923	0.5769	0
80	0.0308	0.2385	0.4462	0
85	0.0154	0.1077	0.3538	0
90	0.0077	0.1077	0.2692	0
95	0.0077	0.0538	0.1615	0
100	0.0000	0.0231	0.1231	0

We observe no false acceptances in the evaluated different-person trials for any tested value of f. This is an empirical result for the SCface dataset and does not imply that the population-wide FAR is exactly zero.

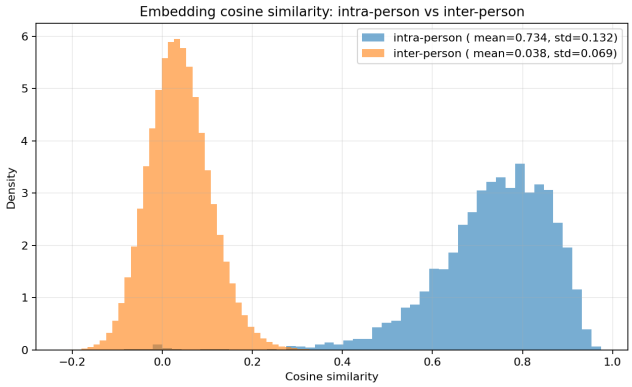
For the end-to-end evaluation, we select $s = 60$ and $f = 3$ because this configuration achieves the highest TAR among the tested settings. We evaluate the security of this parameter choice separately in Section 5.2.

End-to-end crowd-flow accuracy. Using the selected parameters $s = 60$ and $f = 3$, we evaluate the complete crowd-flow pipeline. For each target intersection size g, we construct two camera-epoch observation sets containing exactly g common individuals. The remaining individuals appear in only one of the two sets. Thus, g represents the ground-truth crowd flow between the two camera-epoch sets. We repeat this experiment 100 times using randomly selected individuals and face observations.

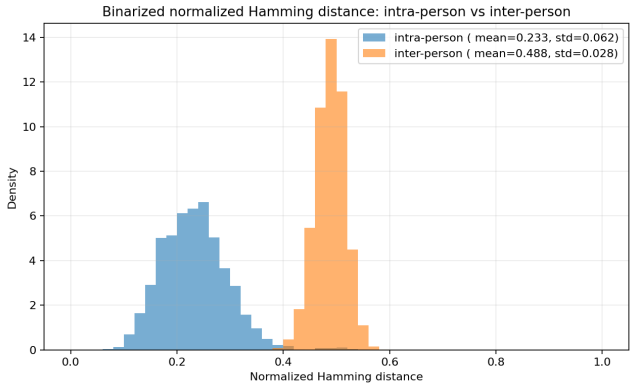
In the first observation set, the camera derives three local identifiers from the $f = 3$ face observations. The first local identifier is used to generate a global identifier R and a helper record. The remaining local identifiers are used to create additional helper records that protect the same R.

In the second observation set, the camera similarly derives three local identifiers from the $f = 3$ face observations. For an individual, the camera tests these local identifiers against all helper records created in the first set. If any reproduction attempt opens at least one locker, the camera reproduces the existing global identifier R and inserts it once into the second observation set. If no helper record opens, the camera generates a new global identifier and corresponding helper records.

After processing both observation sets, we insert their global identifiers into Bloom filters and estimate the intersection cardinality. This procedure allows us to distinguish the error introduced by Sample-Then-Lock reproduction from the additional approximation introduced by Bloom-filter estimation.

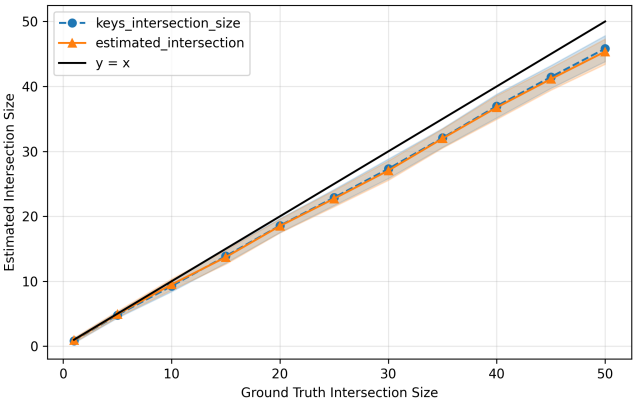


(a) Cosine-similarity distributions for intra-person and inter-person embedding comparisons.

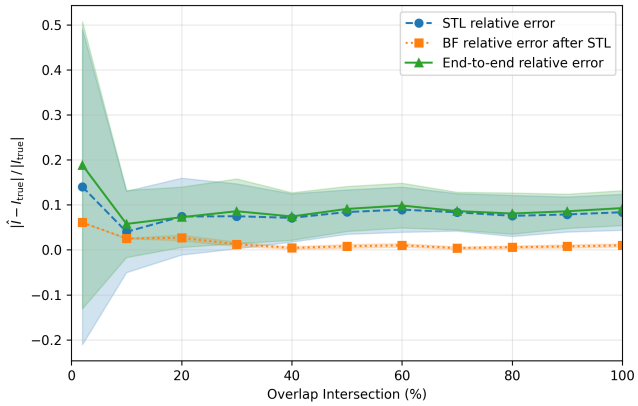


(b) Normalized Hamming-distance distributions for intra-person and inter-person binary local identifiers.

Figure 5: Separability of face observations before and after binarization. Mean-threshold binarization preserves most of the separation between same-person and different-person observations.



(a) Ground-truth intersection size, intersection obtained from Sample-Then-Lock global identifiers, and final Bloom-filter estimate.



(b) Relative error introduced by Sample-Then-Lock reproduction and Bloom-filter estimation.

Figure 6: End-to-end crowd-flow accuracy over 100 trials for each ground-truth intersection size. Lines show the mean, and shaded regions represent ±1 standard deviation. Most of the observed error originates from Sample-Then-Lock reproduction, while Bloom-filter estimation adds a smaller approximation error.

Figure 6a compares the ground-truth intersection size with two estimates: the intersection obtained from the Sample-Then-Lock-derived global identifiers and the final Bloom-filter estimate. Each line shows the mean over 100 trials, while the shaded regions represent ±1 standard deviation from the mean. Figure 6b separates the error introduced by each stage. Let g denote the ground-truth intersection size, g_{STL} the intersection size obtained from the Sample-Then-Lock-derived global identifiers, and $\hat{g}_{\mathrm{BF}}$ the final Bloom-filter estimate. For $g > 0$, we compute $e_{\mathrm{STL}} = |g_{\mathrm{STL}} - g|/g$, $e_{\mathrm{BF}} = |\hat{g}_{\mathrm{BF}} - g_{\mathrm{STL}}|/g$, and $e_{\mathrm{E2E}} = |\hat{g}_{\mathrm{BF}} - g|/g$.

The value e_{STL} measures the error caused by failures to reproduce global identifiers. The value e_{BF} measures the additional approximation introduced by Bloom-filter estimation, while e_{E2E} measures the final crowd-flow estimation error. The results show that Sample-Then-Lock reproduction is the main source of error under the selected parameters. The Bloom-filter stage introduces only a small additional deviation.

5.2 Security Analysis

We evaluate the irreversibility and unlinkability requirements defined in Section 3. The analysis considers camera-side processing, helper records exchanged among cameras, encrypted Bloom filters stored by the query server, and aggregate outputs returned to analysts.

Camera-side processing. Raw facial images, face embeddings, and binary local identifiers remain inside the camera. The camera uses these representations to derive global identifiers and discards

Table 2: Estimated subset entropy for different subset sizes. Average entropy describes a typical locker input, while minimum entropy describes the weakest sampled locker input.

Subset size s	Average entropy	Minimum entropy
60	54	53
65	58	57
70	63	61
75	67	64
80	71	70
85	75	74
90	79	78
95	83	82
100	86	84

them at the end of the epoch. It does not transmit them to the query server or analysts. Under the trusted-camera assumption, only helper records and encrypted Bloom filters leave the local biometric-processing stage.

Helper-record security. Section 4.4 describes the Sample-Then-Lock construction. Here, we estimate the security-relevant property of the resulting helper records without repeating the construction. Each locker depends on a subset value B_{I_i} containing s positions from the local identifier. Because an adversary can target the easiest locker, the relevant security measure is the minimum of entropies for included subsets, $\alpha = \min_{1\le i\le V} H_\infty(B_{I_i})$. This value measures the unpredictability of the weakest locker input. A larger α implies a lower probability of guessing an input that opens a locker. It therefore measures resistance to offline guessing rather than reproduction accuracy. All entropy tests are heuristic [20], we estimate α using the heuristic entropy test adopted by Shukla et al. [16, 21]. For each sampled subset I_i, we restrict all local identifiers to the positions in I_i and compute the normalized Hamming-distance distribution over different-person pairs. Let μ_i^{diff} and σ_i^{diff} denote the mean and standard deviation of this distribution. We estimate the effective degrees of freedom as $\mathrm{df}_i = \frac{\mu_i^{\text{diff}}(1-\mu_i^{\text{diff}})}{(\sigma_i^{\text{diff}})^2}$, and the subset min-entropy as $\widehat{H}_i = -\mathrm{df}_i \log_2\big(\max\{\mu_i^{\text{diff}}, 1-\mu_i^{\text{diff}}\}\big)$.

We report $\frac{1}{V}\sum_{i=1}^{V}\widehat{H}_i$ to describe a typical subset and $\widehat{\alpha} = \min_{1\le i\le V}\widehat{H}_i$ to describe the weakest subset. The minimum determines the estimated security of the complete helper record because an adversary can target the most predictable locker input.

For the selected setting $s = 60$, the weakest sampled subset has an estimated min-entropy of 53 bits. Under the adopted entropy model, this means that the best single guess for the weakest subset succeeds with probability approximately 2^{-53}. This value is a heuristic estimate derived from the available dataset rather than a formal population-wide security guarantee.

In the proposed crowd-monitoring scenario, helper records remain within the trusted camera domain rather than being intentionally published. We therefore consider the 53-bit estimate sufficient for evaluating the feasibility of the current prototype. This estimate is also higher than the 45 bits of security for facial fuzzy extraction reported by Zhang et al. [22].

Encrypted Bloom-filter protection. Bloom filters provide compact set representations but do not provide cryptographic confidentiality. Access to a plaintext Bloom filter would allow an adversary to test candidate global identifiers and potentially link observations across cameras or epochs. Each camera therefore encrypts its Bloom filter before sending it to the query server.

The query server stores only $\mathrm{Enc}(\mathrm{BF}_{c,e})$ and evaluates footfall and crowd-flow queries over ciphertexts. Because it does not possess the secret key, it cannot inspect Bloom-filter positions, test candidate global identifiers, or decrypt query results.

We use CKKS with polynomial-modulus degree $N_{\text{poly}} = 8192$, coefficient-modulus bit sizes $(40, 20, 40)$, total modulus size $\log Q = 100$, and scale $\Delta = 2^{20}$. Each 9585-bit Bloom filter is encrypted in chunks of 4096 values, resulting in three ciphertext vectors per camera and epoch. Microsoft SEAL parameter validation classifies this parameter set at the TC128 security level, corresponding to approximately 128 bits of classical RLWE security.

The Bloom-filter parameters $n = 1000$, $p = 0.01$, $m = 9585$, and $k = 7$ determine estimation accuracy and storage overhead. They do not provide cryptographic confidentiality, which instead follows from the homomorphic-encryption parameters.

Output disclosure. The query server returns only the encrypted set-bit count. The analyst decrypts this value and applies the cardinality estimator locally. The plaintext output is therefore limited to an aggregate footfall or crowd-flow estimate. The analyst does not receive plaintext Bloom filters or individual global identifiers.

5.3 Timing, Scalability, and Storage Overhead

We evaluate the runtime and storage overhead of the two main computational stages: Sample-Then-Lock processing at the cameras and encrypted Bloom-filter processing at the cameras and server. We perform all measurements on a MacBook Pro (2022) equipped with an Apple M2 processor and 24 GB of memory, running macOS Sonoma 14.1. The measurements do not include face detection or face-embedding extraction.

Sample-Then-Lock overhead. We implement Sample-Then-Lock in Rust using HMAC-SHA512. Each locker protects the 256-bit value $0^{128}\|R$, where R is a 128-bit global identifier. We therefore truncate each 512-bit HMAC-SHA512 output to 256 bits before constructing or opening a locker. More precisely, the implementation computes $z_i = \mathrm{Trunc}_{256}(\text{HMAC-SHA512}(h_i, b_{I_i}))$ and constructs the locker value as $p_i = (0^{128}\|R) \oplus z_i$.

Each helper record contains $V = 250{,}000$ lockers. Loading the pre-generated locker subsets takes approximately 6.0 s. This setup cost is incurred only once when the process starts.

Generating the global identifier and helper records for one person using $f = 3$ face observations takes 265.9 ms on average. The first observation generates R and a helper record, while the remaining observations produce additional helper records that protect the same R.

We measure reproduction as one atomic call to $\textsc{Rep}(b', P)$, in which one local identifier b' is tested against one helper record P. The implementation scans the lockers sequentially and stops when a locker opens. In a worst-case no-match attempt, it evaluates all $V = 250{,}000$ lockers. One full scan takes 442.8 ms on average.

Each locker stores a 512-bit HMAC key h_i and a 256-bit masked value p_i, requiring 96 bytes. A helper record containing $V = 250{,}000$ lockers therefore requires approximately 24 MB of raw storage, equivalent to approximately 22.9 MiB. Since each of the f face observations contributes one helper record, the selected setting $f = 3$ requires approximately 72 MB per person.

Encrypted Bloom-filter overhead. We evaluate the encrypted Bloom-filter stage using the Bloom-filter and CKKS parameters reported in Section 5.2. Encrypting one Bloom filter takes 18.7 ms and produces a serialized encrypted Bloom filter of approximately 462 KiB per camera epoch.

On the server, homomorphically counting the set bits in an encrypted Bloom filter takes 109.9 ms. A pairwise crowd-flow query additionally requires a homomorphic AND operation between two encrypted Bloom filters, which takes 7.6 ms. Table 3 summarizes the measured runtime and storage overhead.

Scalability. The main scalability bottleneck is searching the helper-record pool. Let M denote the number of candidate identities whose helper records are available to a camera. Each candidate identity has f helper records, and the camera derives f local identifiers from the face observations of a newly observed person. In the worst case, every local identifier must be tested against every helper record. Matching one observed person against one candidate identity therefore requires at most f^2 reproduction attempts, corresponding to f^2V locker evaluations.

For $f = 3$, matching one observed person against one candidate identity may require up to nine full helper-record scans. Based on the measured atomic reproduction time, this corresponds to approximately $9 \times 442.8\,\text{ms} = 3.99\,\text{s}$ in the worst-case no-match setting. Successful reproduction may terminate earlier when a locker opens.

Matching one observed person against all M candidate identities has worst-case complexity $O(Mf^2V)$. Processing N observed individuals therefore requires $O(NMf^2V)$ locker evaluations. When the number of stored candidate identities grows proportionally to the number of observed individuals, the reproduction cost becomes quadratic in the population size.

Helper-record storage grows linearly with the number of identities. A pool of M identities requires approximately $24fM$ MB at each camera. With $f = 3$, this becomes $72M$ MB. For example, storing helper records for 1,000 identities requires approximately 72 GB per camera.

Encrypted Bloom-filter storage grows with the number of cameras and epochs. Each camera produces one encrypted Bloom filter of approximately 462 KiB per epoch. For $|C|$ cameras and $|E|$ epochs, the server stores approximately $462|C||E|$ KiB. These results show that helper-record storage and exhaustive Sample-Then-Lock reproduction are the main scalability limitations of the current prototype.

Table 3: Timing and storage overhead of the encrypted crowd-monitoring pipeline.

Stage	Operation	Mean	Std.
Fuzzy Extractor	$\textsc{Gen}$ — one person, $f = 3$ templates	265.9 ms	14.5 ms
	$\textsc{Rep}(b', P)$ — atomic, sequential full V	442.8 ms	3.2 ms
	Helper record size	24 MB	–
FHE — Camera	Encrypt $\text{Enc}(\text{BF}_{c,e})$ per epoch	18.7 ms	0.2 ms
	Serialized encrypted BF size, $\lvert\text{Enc}(\text{BF}_{c,e})\rvert$	462 KiB	0.6 KiB
FHE — Server	Homomorphic bit-count	109.9 ms	0.7 ms
	Crowd-flow AND	7.6 ms	0.0 ms

6 DISCUSSION AND FUTURE WORK

The experimental results presented in Section 5 show that our framework can derive stable identifiers from noisy facial observations and use them for encrypted crowd-flow estimation. These findings indicate that the combination of similarity-preserving local identifiers, Sample-Then-Lock fuzzy extraction, and encrypted Bloom-filter cardinality estimation is a promising baseline for privacy-preserving crowd monitoring without stable device identifiers. Rather than assuming that individuals can be mapped to exact identifiers, our system provides an end-to-end path from noisy camera-derived observations to reusable random identifiers and encrypted aggregate queries.

At the same time, the current implementation should be viewed as a research baseline rather than a complete deployment-ready system. We therefore identify several directions for future work.

Real-world multi-camera deployment. Our evaluation uses the SCface dataset as a controlled proxy for cross-camera variation, since it contains multiple observations of the same individuals under different camera distances, lighting conditions, and acquisition settings. This allows us to isolate the reliability of the identifier-generation stage. However, a real deployment would introduce additional factors such as face detection failures, motion blur, partial occlusion, repeated observations within the same epoch, and changing environmental conditions. Future work should therefore evaluate the framework on live multi-camera video streams. Such a deployment would also allow studying how temporal aggregation over multiple frames can improve identifier reproducibility.

Scalability of global identifier generation. The current prototype demonstrates that Sample-Then-Lock can be used to obtain consistent random identifiers from noisy facial observations, but the helper record layer remains the main scalability bottleneck. In our selected configuration, helper records are large and reproduction requires scanning a large number of lockers. This cost is acceptable for establishing a baseline, but larger deployments with many cameras, epochs, and observed individuals will require more efficient helper-record management. Future work should investigate indexing, pruning, batching, and approximate pre-filtering techniques so that cameras do not need to test every helper record for every new observation. Another promising direction is to optimize subset selection, reducing the number of lockers needed to achieve the same reproducibility and min-entropy guarantees.

Enhancing the output privacy. The proposed framework protects raw images, embeddings, local identifiers, and Bloom-filter contents from the server. Nevertheless, the final footfall and crowd-flow counts are released to authorized analysts. Aggregate outputs may still reveal sensitive information when queries concern sparsely populated regions, or cover narrow time spans. In our previous work, we addressed this problem through controlled uncertainty

using detection sampling [11] and differentially private computation over encrypted Bloom filters [12].

This framework provides a first end-to-end baseline for privacy-preserving crowd-flow estimation from noisy facial observations. It shows that encrypted crowd-flow analytics need not rely on stable device identifiers such as MAC addresses, and that fuzzy extraction can bridge the gap between noisy biometric observations and privacy-preserving set-cardinality estimation. This makes the system a foundation for future work on scalable and deployable crowd-monitoring protocols.

7 CONCLUSION

We extend existing privacy-preserving crowd-flow systems that assume stable identifiers to a setting where observations are derived from noisy biometric data, such as camera-based face detections. Our method converts facial embeddings into local identifiers, uses Sample-Then-Lock fuzzy extraction to reproduce global identifiers from sufficiently similar observations, and encodes the resulting identifiers in encrypted Bloom filters for aggregate footfall and crowd-flow estimation.

Experiments on the SCface dataset show that this pipeline is feasible for privacy preserving crowd monitoring estimation. With the selected parameters, Sample-Then-Lock achieves a true-acceptance rate of 0.9077 with three samples per identity, while maintaining a minimum estimated subset min-entropy of 53 bits and no observed false acceptances in the tested trials. These results provide an initial baseline for privacy-preserving crowd-flow estimation from noisy facial observations without relying on stable device identifiers.

8 ACKNOWLEDGEMENTS

This research was funded by the Dutch Research Council (NWO) through the PERSPECTIEF programme P21-08, “XCARCITY.” We gratefully acknowledge their financial support. We also sincerely thank Amey Shukla and Benjamin Fuller for their valuable guidance and assistance, and for sharing their implementation of Sample-Then-Lock (STL) [16] with us.

REFERENCES

[1] Michael Southworth. Designing the walkable city. *Journal of Urban Planning and Development*, 131(4):246–257, 2005. doi: 10.1061/(ASCE)0733-9488(2005)131:4(246). URL https://doi.org/10.1061/(ASCE)0733-9488(2005)131:4(246).

[2] Yuan Lai and Constantine E. Kontokosta. Quantifying place: Analyzing the drivers of pedestrian activity in dense urban environments. *Landscape and Urban Planning*, 180:166–178, 2018. ISSN 0169-2046. doi: https://doi.org/10.1016/j.landurbplan.2018.08.018. URL https://doi.org/10.1016/j.landurbplan.2018.08.018.

[3] C. Martella, J. Li, C. Conrado, and A. Vermeeren. On current crowd management practices and the need for increased situation awareness, prediction, and intervention. *Safety Science*, 91:381–393, 2017. ISSN 0925-7535. doi: https://doi.org/10.1016/j.ssci.2016.09.006. URL https://doi.org/10.1016/j.ssci.2016.09.006.

[4] European Parliament and Council of the European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016. Official Journal of the European Union, L 119, 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[5] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. Privacy-preserving crowd-monitoring using bloom filters and homomorphic encryption. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '21, page 37–42, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382915. doi: 10.1145/3434770.3459735. URL https://doi.org/10.1145/3434770.3459735.

[6] Antoni B. Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2008. doi: 10.1109/CVPR.2008.4587569.

[7] Justas Brazauskas, Chris Jensen, Matthew Danish, Ian Lewis, and Richard Mortier. Cerberus: Privacy-preserving crowd counting and localisation using face detection in edge devices. In *Proceedings of the 7th International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '24, page 25–30, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705397. doi: 10.1145/3642968.3654817. URL https://doi.org/10.1145/3642968.3654817.

[8] Chen Zhang, Jing-an Cheng, Qiang Zhou, Wenzhe Zhai, and Mingliang Gao. Privacy-preserving crowd counting via quantum-enhanced federated learning. *Expert Systems*, 42(9):e70098, 2025. doi: https://doi.org/10.1111/exsy.70098. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.70098. e70098 EXSY-May-25-1921.R1.

[9] Bo Tian, Bowen Zhao, Yang Xiao, Yang Liu, Qingqi Pei, and Yulong Shen. RAPOO: An Efficient Privacy-Preserving Facial Expression Recognition via Mobile Crowdsensing . *IEEE Transactions on Mobile Computing*, 24(11):11568–11581, 2025. ISSN 1558-0660. doi: 10.1109/TMC.2025.3581687. URL https://doi.ieeecomputersociety.org/10.1109/TMC.2025.3581687.

[10] Manish Bhat, Samuel Paul, Umesh Kumar Sahu, and Umesh Kumar Yadav. Revolutionizing crowd surveillance through voice-driven face recognition empowering rapid identification: towards development of sustainable smart cities. *Engineering Research Express*, 6(2):025219, 2024. doi: 10.1088/2631-8695/ad4ae9. URL https://doi.org/10.1088/2631-8695/ad4ae9.

[11] Fatemeh Marzani, Thijs van Ede, Geert Heijenk, and Maarten van Steen. Stop watching me! moving from data protection to privacy preservation in crowd monitoring. In Mila Dalla Preda, Sebastian Schrittwieser, Vincent Naessens, and Bjorn De Sutter, editors, *Availability, Reliability and Security*, pages 46–67, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-032-00624-0.

[12] Fatemeh Marzani, Thijs van Ede, Geert Heijenk, and Maarten van Steen. Differentially private crowd monitoring over encrypted bloom filters. In *Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing*, SAC '26, page 1785–1792, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722943. doi: 10.1145/3748522.3779813. URL https://doi.org/10.1145/3748522.3779813.

[13] Minchul Kim, Anil K. Jain, and Xiaoming Liu. Adaface: Quality adaptive margin for face recognition. In *In Proceeding of IEEE Computer Vision and Pattern Recognition*, New Orleans, LA, June 2022.

[14] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam D. Smith. Reusable fuzzy extractors for low-entropy distributions. *Journal of Cryptology*, 34(1):1–33, 2021. doi: 10.1007/s00145-020-09367-8.

[15] Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? *IEEE Transactions on Information Theory*, 66(8):5282–5298, 2020. doi: 10.1109/TIT.2020.2984751.

[16] Amey Shukla, Luke Demarest, Benjamin Fuller, Sohaib Ahmad, Caleb Manicke, Alexander Russell, and Sixia Chen. Fuzzy extractors are practical: Cryptographic strength key derivation from the iris. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, pages 3605–3619. ACM, 2025. doi: 10.1145/3719027.3765098.

[17] S Joshua Swamidass and Pierre Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling*, 47(3):952–964, 2007.

[18] Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978. URL https://api.semanticscholar.org/CorpusID:6905087.

[19] Mislav Grgic, Kresimir Delac, and Sonja Grgic. Scface – surveillance cameras face database. *Multimedia Tools and Applications*, 51(3):863–879, 2011. ISSN 1573-7721. doi: 10.1007/s11042-009-0417-2. URL http://dx.doi.org/10.1007/s11042-009-0417-2.

[20] Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log(n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 685–694, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306911. doi: 10.1145/1993636.1993727. URL https://doi.org/10.1145/1993636.1993727.

[21] J. Daugman. How iris recognition works. *IEEE Trans. Cir. and Sys. for Video Technol.*, 14(1):21–30, January 2004. ISSN 1051-8215. doi: 10.1109/TCSVT.2003.818350. URL https://doi.org/10.1109/TCSVT.2003.818350.

[22] Kaiyi Zhang, Hongrui Cui, and Yu Yu. Facial template protection via lattice-based fuzzy extractors. Cryptology ePrint Archive, Paper 2021/1559, 2021. URL https://eprint.iacr.org/2021/1559.