Centipede: Leveraging the Distributed Camera Crowd for Cooperative Video Data Storage

Jiaping Yu, Haiwen Chen, Kui Wu, *Senior Member, IEEE*, Tongqing Zhou, Zhiping Cai, *Member, IEEE*, and Fang Liu, *Member, IEEE*

***Abstract*—Surveillance cameras have been extensively used in smart cities and high security zones. However, with the exploding deployment of smart cameras, the rapid growth of cloud workloads from vision-based IoT applications are becoming a huge burden for all cloud service providers. Some researchers have proposed mechanisms, such as compression and deduplication to reduce the video traffic size, but these methods cannot offset the enormous growth of data volume. Most of the surveillance video data do not need to be proceeded in real time. By making use of the IoT camera's onboard resources to store the data, the cloud workloads can be fundamentally reduced. However, recent incidents have posed a new, powerful geo-range attack, where the attacker may compromise a group of surveillance cameras located within an area. Existing simple onboard solutions cannot offer secure defense against such geo-range attacks. To tackle the problem, we develop *Centipede*, a cooperative video data storage system that distributes video content across geographically dispersed surveillance cameras. It generates secure copies for the video content and enhances data security by judiciously distributing erasure-coded video blocks across optimally-chosen surveillance cameras. In this article, we implement *Centipede* and evaluate its performance. *Centipede* is the first solution that can fundamentally reduce the cloud workload and defend against geo-range attacks.**

***Index Terms*—Distributed storage, geo-range attacks, surveillance cameras.**

I. Introduction

With the continuous development of computer vision and semiconductor technology, vision-based IoT application has been widely adopted in the fields like smart cities [1], AI enabled transportation, and automated workerless factories. For instance, DiDi, the largest ride-hailing company in China, requires to install cameras in cars to monitor the trip in real time for the safety of drivers and customers [2]. In fact, the public transportation systems in many cities worldwide has deployed security cameras in buses or mass transit railway (MTR) [3].

The exploding deployment of smart cameras in different kinds of IoT applications leads to tremendously increased multimedia data over the Internet [4]. According to [5], by 2022, Internet video will represent 82% of all business Internet traffic, wherein Internet video surveillance traffic will increase seven fold. Intuitively, uploading the real-time captured video contents to a powerful cloud platform for computation-intensive data analytic, processing and centralized data storage is a natural choice, thus is supported and adopted by main IoT camera manufactures. As shown in Fig. 1(a), under such settings, the cloud would collect video segments from distributed cameras for archival purpose and responses to the viewing requests of users. However, the growing amount of standalone smart cameras will inevitably add to the burden of the cloud and degrade the perceived performance, significantly limiting the scalability of the above centralized solution [6].

In fact, what the users actually expect is the effective on-demand retrieval of the video data. Here, effectiveness means that users could retrieve the surveillance data even if the surveillance devices are under the security threats like natural disaster or sabotage. As a representative, for surveillance systems deployed in smart homes, owners usually launches a remote video viewing session when certain motions or voices are detected [7]. To this end, transmitting all the video data to the cloud in real time is, in most cases, not necessary. Considering the improved performance of IoT cameras' onboard hardware, it is promising to keep the generated data at the local storage or private data centers. Specifically, as shown in Fig. 1(b), an alternative, yet more efficient, solution only uploads some key frames or messages (e.g., movement detection in smart home) to the cloud, users then send out request according to these tips and the cloud helps to transfer the answers. Manufacturers like TPLink,[1] Lenovo[2], and Xiaomi[3] have already provided such simple onboard solutions. In this way, IoT applications can manage to serve a large amount of users to view their remote cameras.

Manuscript received December 30, 2020; revised March 22, 2021; accepted April 9, 2021. Date of publication April 22, 2021; date of current version November 5, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62072465; in part by the Key-Area Research and Development Program of Guang Dong Province under Grant 2019B010107001; in part by the National Key Research and Development Program of China under Grant 2018YFB0204301; and in part by the NUDT Research under Grant ZK19-38. *(Jiaping Yu and Haiwen Chen contributed equally to this work.) (Corresponding authors: Tongqing Zhou; Zhiping Cai.)*

Jiaping Yu, Haiwen Chen, Tongqing Zhou, and Zhiping Cai are with the College of Computer, National University of Defense Technology, Changsha 410005, China (e-mail: yujiaping19@nudt.edu.cn; chenhaiwen13@nudt.edu.cn; zhoutongqing@nudt.edu.cn; zpcai@nudt.edu.cn).

Kui Wu is with the Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: wkui@uvic.ca).

Fang Liu is with the School of Design, Hunan University, Changsha 410005, China (e-mail: fangl@hnu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3074823

[1]https://www.kasasmart.com/us/products/security-cameras
[2]https://manuals.plus/lenovo/lenovo-snowman-ip-camera-user-manual
[3]https://www.mi.com/us/mi-home-security-camera

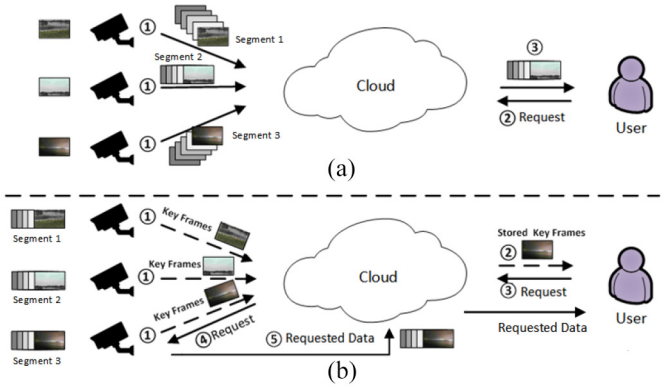


Fig. 1. (a) Cloud-based centralized solution. Devices upload all their recorded videos to the cloud. When user request, cloud send the requested data to the user. (b) Simple onboard solution. Devices upload the key frames to the cloud only when movement or other security issues detected. Users then send out request according to these tips and the cloud helps to transfer the answer from the origin camera devices.

Yet, we point out that keeping the complete video segments at their sources incurs threats on data confidentiality and availability. The underlying problem lays in that IoT cameras can be easily compromised due to the weak physical protection onsite, leading to single point offline and the onboard data inaccessible, or even damaged. From recent news reports, a bomb attack in Malibagh on May 26, 2019, which kills a female assistant inspector and two others, damaged all the nearby security cameras, making it impossible for the police to find the terrorists from video recordings. In recent non-peaceful protests in the U.S., it has been reported that the protesters have smashed many on-street surveillance cameras to avoid themselves to be visually captured. In this article, we denote such a type of attack as *geo-range* attacks, where an adversary is capable of compromising (via physically damaging or remote hacking) one or more cameras located in certain physical areas. We believe that *geo-range* attacks pose a new challenge in developing secure onboard data storage solution for surveillance and forensic needs.

To reconcile the tension between transmission efficiency and data storage security, in this research, we propose to develop a cooperative and distributed IoT video data storage system, named Centipede, by exploiting the assistance from one's peer crowd. Briefly, Centipede keeps the data at local storage to maintain scalability with respect to increasing amount of IoT devices and generates and stores the data replicates of each device in its peer devices. Instead of storing complete video data segments in one device, we split each segment into blocks with erasure coding. Each block from the same segment is stored in different peer devices. By using erasure codes (ECs) [8] and encryption techniques during data dispersing, it is expected to protect data availability and confidentiality when facing geo-range attack. We call our system *Centipede* to emphasize its security since we can successfully recover all video files even if a subset of cameras are compromised. As an analogy, a centipede can walk even if losing several legs.

In the system, we use the local storage of surveillance cameras and a virtualized *coordinator*. All the surveillance data are encoded and distributed to the local storage of surveillance cameras. The *coordinator* is a virtual component, which can be deployed on a cloud or edge server. It stores the hash values and calculates the video distribution schedules for surveillance cameras.

Nevertheless, it is technically difficult to build *Centipede* to achieve desirable performance under practical constraints. To defend against geo-range attacks, each device should "push" their data to other backup devices as far as possible. Such intentions incur the cost on transmission delay and are limited by the storage capacities when scaling to large amount of devices. Hence, in particular, we need to jointly consider the constraints of local storage size and the network delay and carefully select one's backup devices during data dispersing.

To address the above challenge, we utilize an attacker's physical constraint in a geo-range attack, i.e., the attacker can only damage nearby cameras. This is reasonable in many real-world attacks when an attacker can only launch physical sabotage in a local area. We propose to find an evenly distributed camera set, which implies a series of locations that would not be attacked simultaneously, for each camera and send the data replicates to them. We call this technique *geo-aware erasure coding*.

Overall, we make the following contributions.

1) We raise the awareness of geo-range attacks and describe a quantitative model for them. Having show their influences and damage in real-world events, such attacks threaten the data availability of IoT cameras.
2) We design *Centipede*, a cooperative video data storage system that could backup the surveillance data locally instead of sending massive video uploading workloads to the cloud. With geo-aware erasure coding, *Centipede* can effectively distribute video content across geographically dispersed cameras to achieve data security under resource and cost constraints.
3) With the deployment of a prototype system as well as simulation study, we systematically evaluate the performance of *Centipede* with respect to availability in the presence of various levels of geo-range attacks.

The remainder of this article is organized as follows. Section II introduces the background and related work. Section III presents the attack model and design goals. The system architecture and the core algorithm in *Centipede* are introduced in Section IV. We implement and evaluate *Centipede* in Section V, Then we discussed the performance and limitation in Section VI and conclude this article in Section VII.

II. Related Work

Researchers have conducted studies to reduce the excessive workload from vision-based IoT devices to the cloud. Some of them focus on optimizing the cloud transmission strategy itself, while others are making efforts to analyze the distributed data storage systems and looking forward to find new solutions to reduce the traffic size.

A. Cloud Workflow Optimization

The optimization of video traffic size to the cloud has been thoroughly studied in the past few years. Related studies mainly consist of two aspects: one is the optimization of video contents with compression and deduplication, the other is the optimization of the cloud architecture.

Compression and deduplication are most common used methods. Liu *et al.* [9] presented an efficient video preprocessing strategy for wireless surveillance systems using light-weight AI and IoT collaboration. With their work, the redundant data in the surveillance video can be reduced during transmission. Hu *et al.* [10] proposed Starfish, a design that could better compress the video data with less data loss in the process of data transmission to the cloud. In [11], image processing framework proposed by Chamain *et al.* can achieve adequate performance with limited channel bandwidth to the cloud. Chen *et al.* [12] proposed a layered adaptive compression design for efficient data collection, which could achieve better performance than conventional clustered compression schemes. Aiming at better allocate the cloud bandwidth, Al-Dulaimy *et al.* [13] presented the concept of bandwidth slicing, which means distribute the limited cloud bandwidth to all virtual machines on the cloud based on their needs.

Another direction is to optimize cloud architecture, which includes deploying edge servers to share the pressure of the cloud or adopting multicloud architecture.

UniDrive [14] is an example multicloud storage system. It uses multiple consumer cloud storage (CCS), and stores the coded blocks to different cloud storage systems. UniDrive can enhance the performance of single cloud storage system, since it uses multiple clouds to transfer files in parallel. By following a server-less, client-centric design and by distributing the erasure file blocks to multiple cloud storage, this scheme can effectively reduce the bandwidth consumption for each single cloud service provider. Cai *et al.* [15] proposed a many-objective intelligent algorithm for efficient task scheduling in Internet of Things system based on multi cloud. Jinlong *et al.* [16] proposed an efficient multicloud storage system CoCloud based on Web APIs. By deploying proxies that can efficiently access the Web APIs for deduplication operation, CoCloud can achieve user-perceived performance for multicloud collaboration.

These optimization strategies are based on the features of cloud data transmission. Though to some extent, these methods could reduce the pressure of the cloud, the workload size can not be less than the information entropy. As for the multicloud architecture, although the pressure on a single cloud platform is reduced, without an additional compression algorithm, the overall bandwidth consumption of all the cloud storage platform in the network is still the same.

B. Distributed Data Storage and Analysis

To overcome the bottleneck of cloud data transmission, some work leverages distributed architecture for data analysis locally. Zeng *et al.* [17] proposed Distream, a distributed live video analytic system based on the smart camera-edge cluster architecture. Instead of uploading the videos to the cloud, Distream can analysis the video data on local edge clusters. Hung *et al.* [18] proposed a video query optimizer that can balance the resource demands and accuracy of outputs for video analysis. In the field of healthcare, Chen *et al.* [19] designed a distributed system that could connect individuals, community clinics (or family doctors), and hospitals to share the information.

Distributed storage is a solution to fundamentally reduce the large cloud traffic of video data [20]. However, because distributed storage system nodes are usually more fragile than cloud storage, storage devices are not resilient to physical attacks where an attacker can physically capture the storage device. The adversaries may modify the data stored in the local disk [19], or, to make matters worse, may steal or disrupt the storage device, compromising the system's confidentiality, integrity, and availability.

In order to enhance the security of data in distributed systems, researchers have designed a variety of encryption algorithms. Fadiheh *et al.* [21] developed a new approach to detecting the vulnerabilities of device hardware, but they also point out that hardware defects are more than we expected. According to [22], encryption products are not free of vulnerabilities. This is true in particular when the adversaries have physical access to the local storage and can replicate the surveillance data for offline analysis.

However, in the face of attacks aimed at destroying the system, encryption does not increase the probability of data survival. So other researchers conducted a series of studies on data recovery. With clustering [23] and erasure coding [24], a high fault tolerance storage system can be built with a distributed storage system. Li *et al.* [25] proposed a unified and configurable framework for readily deploying a variety of erasure coding solutions into existing distributed storage systems. While Liang *et al.* [26] combined the blockchain with a distributed data storage system and proposed a secure data storage and recovery scheme in the blockchain-based network.

In practice, large-scale attacks against video devices usually occur within a specific geographic location. Therefore, no matter what recovery strategy is adopted in actual deployment, it is necessary to ensure that the distance between storage devices cannot be too large to ensure that as many storage devices as possible survive the attack.

III. Preliminaries

This section describes the research problem by formally building a quantitative model of geo-range attacks. Then the design goals are presented by jointly considering the security concerns and data storage efficiency.

A. Quantitative Model for Geo-Range Attacks

We use a 2-D grid to cover the whole area under surveillance. The granularity of the grid is set based on specific surveillance requirements. Without loss of generality, we can assume that the dimension of grid is $W \times H$. We assume that the system may be subject to single-point geo-range attack where the sabotaged devices fall within one circular area, as well as

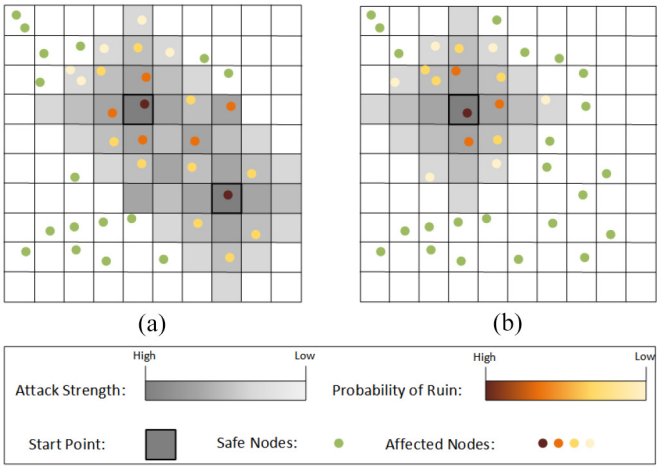


Fig. 2. Example of (a) single and (b) multiple point geo-range attacks.

multipoint geo-range attack where the sabotaged devices fall within several (overlapping) circular areas.

We call the center of the circular area a *start point* of geo-range attack. Note that a multipoint geo-range attack may have multiple start points. For example, a start point may correspond to a bomb of terrorist attack. Since the granularity of the grid corresponds to the surveillance requirements, we only need to consider the circular area at the level of grid cells. To be specific, assume that we use a $W \times H$ boolean matrix B to denote the whole area. $B_{xy} = 0$ represents that the grid cell (x, y) is not a start point, and $B_{xy} = 1$ means that the grid cell (x, y) is a start point. Fig. 2 shows an example of single-point geo-range attack and an example of multipoint geo-range attack, respectively.

We need to assume some constraints on the geo-range attack, because no defense is possible if the attack can compromise all the surveillance devices. For this reason, we assume that the adversaries can only choose a limited number of start points. In practice, it is reasonable to assume that a surveillance device might survive the geo-range attack even if it falls within the attack range. For instance, an adversary might not find and damage all the nearby cameras. Therefore, we propose a probabilistic model, in which a camera is more likely to be compromised if it is more closer to a start point. We adopt a similar probabilistic model [27] that considers the distance between the target and the sensor

$$p_{((m,n),(i,j))} = \begin{cases} e^{-\alpha d((m,n),(i,j))}, & \text{if } d((m,n),(i,j)) \le r \\ 0, & \text{if } d((m,n),(i,j)) > r \end{cases} \quad (1)$$

where $p((m,n),(i,j))$ is the probability that a surveillance device in cell (m,n) is destructed by the geo-range attack with start point (i,j), r is the range of the attack, $\alpha \in (0,1)$ is a parameter representing the strength of the attack (a lower α value means a stronger attack), and $d((m,n),(i,j))$ denotes the distance between the surveillance device and the start point. In fact, the factor α plays a role of truncation for the influence physical distance in a geo-range attack. Namely, we usually have $(1/d_{\max}) \le \alpha \ge (1/d_{\min})$ with $d_{\max}$ ($d_{\min}$) stands for the maximum (minimum) effective threat range of a geo-range attack. Meanwhile, we highlight that in a multipoint geo-range attack, if a surveillance device falls within the circular areas of multiple start points as shown in Fig. 2 (a), the probability that the device is destructed is calculated as

$$p = 1 - \prod_{i=1}^{k}(1 - p_i) \quad (2)$$

where $(1 - p_i)$ is the probability that the device survives the attack from ith start point and k is the total number of start points.

In our attack model, we assume that the attackers are not powerful enough to break the *coordinator*. Nevertheless, the attackers can launch geo-range attacks.

1) The attackers may have physical access to the surveillance cameras and their local storage. They are able to destroy the surveillance devices and can read the data stored in the local storage.
2) The attackers, however, can only capture the surveillance devices within a certain area.

The above assumptions are made based on our observations of real-world geo-range attacks: 1) surveillance devices seem to be the only equipment that external adversaries (e.g., terrorists, nonpeaceful protesters, or malicious users) can make direct contact with; 2) the sabotage is usually limited within a relatively smaller area (e.g., a street rather than the whole city); and 3) the *coordinator* can be deployed remotely with high law enforcement such that physical contact for the attackers is impossible and cyber attacks are extremely difficult.

Under the above attack model, all the surveillance devices could be untrustworthy. The adversaries with user privileges may also try to read/erase all the evidence on local storage. In the following, we present a quantitative model that not only facilitates our analysis on geo-range attacks but also can guide our design of video block distribution strategy.

B. Design Objectives

As stated above, our main goal is to develop a feasible cooperative and distributed IoT video data storage system. More specifically, we need to accomplish the following objectives.

1) *Geo-Range Attack Resistance:* As stated above, geo-range attack is an existential threat to the surveillance cameras. Once it happened, all the surveillance cameras within a certain range will be influenced and the data on them possibly damaged.
   Goal 1: Centipede need special design against Geo-range attack. Geo-range attack may harm both the data availability and confidentiality, thus, we have the following sub goals.
2) *Availability:* One of the most important reason users upload their surveillance data to the cloud instead of applying simple onboard solution is the cloud could provide better availability. Local storage of surveillance cameras is not resistant to physical attacks, and once the storage devices are damaged, it would be impossible to recover the data.
   Goal 2: Centipede should provide better availability than simple onboard solution.

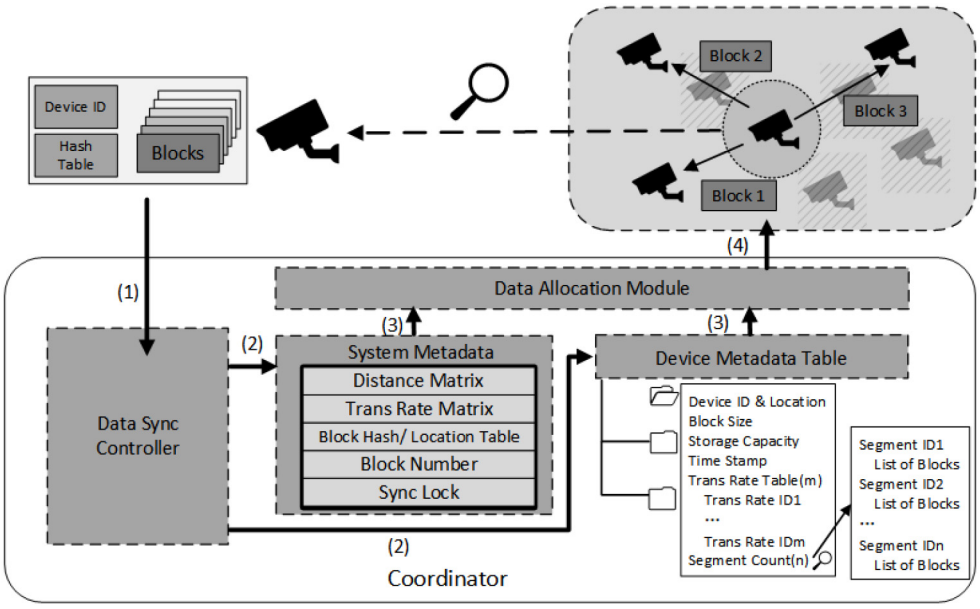


Fig. 3. Workflow and information in the *coordinator* and surveillance devices. (1): Surveillance devices send meta information to the Data Sync Controller. (2): Data sync controller stores the system metadata and device metadata, respectively. (3): Data allocation module calculates the data distribution schedule and sends it back to surveillance devices. (4): Surveillance devices encode local data and distribute the encoded blocks to other peer devices.

3) *Confidentiality:* In geo-range attacks, the adversaries can have physical access to the cameras. And thus it is likely for them to capture the storage devices and read the data. This becomes a huge threat to the data confidentiality. *Goal 3: Centipede* should provide better confidentiality than simple onboard solution.

IV. Our Centipede System

In this section, we introduce our Centipede system from two aspects: 1) system structure and 2) two key algorithm. We first present an overview of the structure and workflow of *Centipede*, then introduce two key algorithm: 1) geo-aware erasure coding and 2) data encryption mechanism in detail.

A. System Overview

As introduced in Section I, evidence have proved that local storage of surveillance cameras may have security risks when facing geo-range attacks. In this article, we hope Centipede could resist the damage of geo-range attacks.

Centipede uses distributed P2P storage with erasure coding. EC is a technology of data protection and recovery in which data segments are broken into smaller blocks and are coded with redundant codes [28]. The key property of EC is that k blocks of data are expanded into n ($n > k$) blocks of encoded data, such that *any* subset of k encoded blocks suffices to reconstruct the original data. Such a code is named as (n, k) code and allows us to recover from up to $n-k$ losses in a group of n encoded blocks. One example is Reed-Solomon code, whose details can be found in [29]. We omit the encoding/decoding details since they are out of the focus of this article.

The detailed data stored in the *coordinator* and the surveillance devices are shown in Fig. 3. The surveillance cameras store the distributed data blocks and the hash table of each block stored in them. The function and data in *coordinator* consists of four parts: 1) data sync controller; 2) system metadata; 3) device metadata table; and 4) the data allocation module. The Data Sync Controller controls the sync lock to assure the data consistency. The System Metadata is calculated based on the received device metadata. It consists of Distance Matrix and Transaction Rate Matrix among all the devices, the corresponding table of block hash and location, the block number in the system, and a sync lock. The Device Metadata Table contains all the devices' metadata, which includes the device ID, location, the storage capacity, time stamp of last data distribution, the transaction rate with other devices, and the list of distributed segments with a list of correspondence blocks for each segment. The Data Allocation Module calculates the data distribution schedule based on the system and device metadata table.

The high-level workflow of *Centipede* consists of four steps, as illustrated in Fig. 3. In the first step, the surveillance devices send the metadata to the *coordinator*. Then the *coordinator* stores the device metadata and generates the latest system metadata based on the uploaded device metadata. Then it calculates the data distribution schedule based on the latest metadata (details in Section IV-B) and sends the schedule to surveillance devices. At the last step, the surveillance devices first split the video segment into blocks and encode data with EC, then distribute the encoded blocks to other peer devices following the schedule.

In this article, we assume an underlying network that supports the communication between the cameras and the communication between the cameras and the *coordinator* [30]. The robustness and security of this network can be supported with existing techniques [31]–[34]. In this article, we applied an encryption method inspired by [33]. According to their

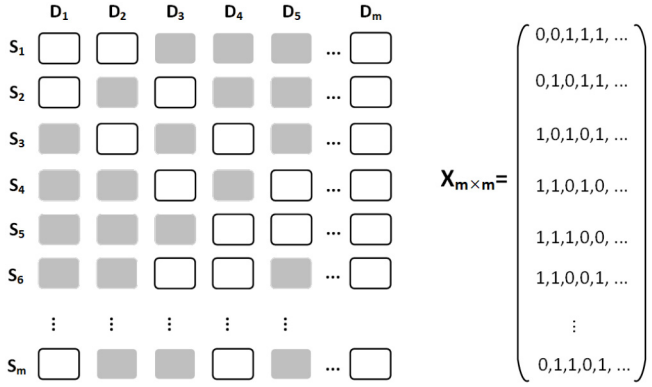


Fig. 4. Data distribution schedule.

works, this encryption mechanism can not only ensure data security, but also facilitate data compression. In addition, we assume the security of the *coordinator*.

B. Geo-Aware Erasure Coding

To defend against geo-range attacks, the *coordinator* needs to carefully calculate the data distribution schedule, such that all the data can be recovered even if the geo-range attack can compromise a group of surveillance devices in a given area.

We consider a system with one coordinator and m surveillance devices, denoted by a set $N = \{1, 2, \ldots, m\}$. The coordinator can collect the devices' meta information, such as the storage space, the transmission rate, and the size of backup surveillance data. Based on the meta information, the coordinator needs to calculate, for every source device, the optimal destination devices that are used to store the encoded data blocks of the source device.

The data distribution schedule is represented by an $m \times m$ Boolean matrix X, as shown in Fig. 4, where S_i denotes source device i and D_j denotes the destination device j. $X_{ij} = 1$ means that surveillance device j stores one data block of surveillance device i. For instance, in this example, the three encoded data blocks of device 1 will be stored at devices 3, 4, and 5, respectively. After receiving the data distribution schedule from the coordinator, a destination surveillance device can pull the data blocks from the source devices instructed by the data distribution schedule.

Centipede needs to maximize the data security in the presence of geo-range attacks. To be specific, it needs to solve the following problem.

(Problem) Secure Data Backup: Assume the encoded k data blocks of source device i are stored at destination devices $i_1 \ldots, i_k$, respectively. Denote the distance of two destination device i_1 and i_2 as $d(i_1, i_2)$. Secure data backup in the context of geo-range attacks is to maximize $d(i_1, i_2)$ over all (i_1, i_2) pairs.

We also need to consider the system constraints when solving the secure data backup problem. In particular, we consider the storage capacity and the network delay.

Storage Constraint: We need to assure that the size of all data blocks stored in a device does not exceed its storage space. Denote the storage capacity of device j as C_j, then we have

$$\sum_{i=1}^{m} B_i \times X_{ij} \leq C_j, \ \forall j \in \{1, 2, \ldots, m\} \tag{3}$$

where B_i represents the data block size from surveillance device i (in bits).

Network Delay Constraint: Transmission rate means the amount of data that can be transmitted per unit time. We use an $m \times m$ matrix $R = [R_{ij}]$ to record the transmission rates, where R_{ij} denotes the transmission rate from device i to device j. We set the transmission delay constraint T_i for device i as

$$T_i = \sum_{j=0}^{m} \frac{B_i}{R_{ij}} \times X_{ij}. \tag{4}$$

The transmission delay constraints in the system can be represented as

$$T_i \leq T_{\max} \ \forall i \in \{1, 2, \ldots, m\} \tag{5}$$

where $T_{\max}$ is the longest delay the system could tolerate, which is defined by the user.

Overall, the secure data backup problem can be formally formulated as

$$\max_{\mathbf{X}} \ d(i_1, i_2) \tag{6a}$$

$$\text{s.t.} \ X_{ii_1} = 1, X_{ii_2} = 1 \ \forall i \in \{1, 2, \ldots, m\} \tag{6b}$$

$$\sum_{i=1}^{m} B_i \times X_{ij} \leq C_j, \ \forall j \in \{1, 2, \ldots, m\} \tag{6c}$$

$$T_i \leq T_{\max} \ \forall i \in \{1, 2, \ldots, m\}. \tag{6d}$$

This is a typical p-dispersion problem, which has been proved to be NP-complete [35]. Therefore, we tackle the problem with heuristics. We utilize the special geometric feature underlying the problem, i.e., the optimal solution must be the one that for each individual node, it maximizes the smallest distance between this node and its backup nodes under the given constraints. This fact can be easily proved by contradiction. This is because if the above condition is not true, we can improve the solution with a higher value of the objective function, and thus the solution cannot be the optimal one. This property provides us with a good heuristic to solve the problem. Due to the geometric feature we call the algorithm geo-aware erasure coding (GEC).

The main idea of GEC is as follows. For any given device, we treat it as the *starting point* of an (imaginary) geo-range attack of unknown radius, and the goal of GEC is to distribute the encoded video blocks of this device to other backup devices outside the radius of the geo-range attack and push the radius as large as possible under the constraints (3)–(5). The pseudo code of GEC is shown in Algorithm 1.

Using GEC as the basic building block, we have different ways to solve the problem. The difference lays in the location selection order, namely, the order of calling GEC, of each device. Obviously, different orders can lead to different node assignment results. As such, we propose three strategies for determining the order.

Algorithm 1 Geo-Aware Erasure Coding

Require: current device G whose data need to distribute;
Ensure: a list of nodes *PlacedNodes* to store G's data blocks;

```
Initialize a new Node object CNode as Chosen Node, a new Level object CLevel as Chosen Level;
NPB = 0; // NPB = # of placed blocks
Level = 0;
while NPB < BlockNum do
    for all grid cells do
        ProceedingQueue.push(G);
    end for
    while !ProceedingQueue.empty() do
        Count =ProcecedingQueue.size();
        for i = 0 to Count do
            temp=ProceedingQueue.front();
            ProceedingQueue.pop();
            if unsearched grid UG near temp exists
            then
                ProceedingQueue.push(UG);
                if PNode in UG meets (3)~ (5) and
                ((PNode.delay <CNode.delay and
                CLevel == Level) or CLevel <Level)
                 then
                    CNode=PNode;
                    CLevel=Level;
                end if
            end if
        end for
        Level++;
    end while
    PlacedNodes.add(CNode);
    NPB++;
end while
Return PlacedNodes;
```

1) *Distributed:* This strategy uses a random permutation to determine the order of devices that call GEC.
2) *Storage First:* This strategy first ranks the devices based on the size of storage space occupied by the data blocks. Devices with larger file blocks will call GEC first. Since the storage space of each device is limited, this strategy tries to avoid the situation that the system has no enough devices to host large data blocks.
3) *Density First:* This strategy is based on the density of surveillance devices in a certain area. To determine the density, we use the clustering algorithm proposed in [36]. Devices in the area with higher density will call GEC first. Within each cluster, we follow the density first strategy, i.e., devices with larger file block sizes will call GEC first.

No matter which strategy is used, for each block, GEC needs to iterate over all cells, which requires $O(W \times H)$ times. Hence, the time complexity of GEC is $O(n \times W \times H)$, where n is the number of data blocks in a device. The time complexity of solving the problem is thus $O(m \times n \times W \times H)$, where m is the total number of devices.

Algorithm 2 Key Matrix Generation

Require: *Seed* of key generation and period length u;
Ensure: The key matrix K

```
for all i = 1 to u × u do
    w_i = RC4(Seed);
end for
for all i = 1 to u do
    for all j = 1 to u do
        W_ij =w_{(i−1)×m+j};
    end for
end for
K = QR(W_{u×u});
return K;
```

C. Encryption Process

The surveillance data need to be protected and secured since it may contain privacy information. In Centipede, to assure the security of the surveillance data during transmission, we apply an encryption process before the data distribution.

The encryption scheme is inspired by [33]. To encrypt the surveillance data, it requires a random one-time use 2-D key matrix. Since the surveillance data can also be transformed into a matrix, the encryption process is to multiply the data matrix with the key matrix. To do this, we need to divide the surveillance data into v period, and the length of each period is u. Then the surveillance data S is transformed into an $u \times v$ matrix, and the key matrix K is an $u \times u$ matrix. The encrypted data can be represented as matrix W, which is the result of $S \times K$.

The theoretical foundation of this encryption process can be found in [33]. To generate the key matrix, a stream cipher need to be applied. Here, we use the RC4 as the key generator. the pseudo code of key generation is shown in Algorithm 2.

Wherein, *QR* represents the householder *QR* decomposition.

V. Evaluation

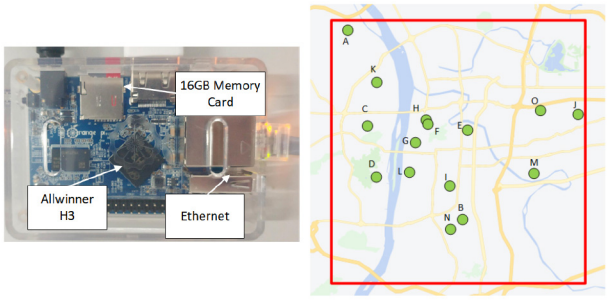
In this section, we build a test bed for *Centipede* deployment and evaluate its performance.

A. System Implementation

We build a test bed for *Centipede* deployment. The device side is implemented with camera and Orange Pi One,[4] an open-source single-board computer. Fig. 5(a) shows a sample Orange Pi One used in our prototype, which has Allwinner H3 with Quad-core Cortex-A7 CPU, 512-MB memory, 16-GB storage space, and a 10/100 Mb/s Ethernet card. The *coordinator* is implemented with a remote server. It is an g6 series server, which has Intel Xeon Platinum 8269CY CPU and 8-GB RAM. All of the devices and the remote server are connected to the Internet through a wired network.

We currently deployed more than 30 devices in Hunan province, China and plan to deploy more than a hundred devices in the future. Fig. 5(b) shows an example of device

[4]http://www.orangepi.org/orangepione/



(a) (b)

Fig. 5. (a) Sample orange Pi one used in our *Centipede* prototype. (b) Map of deployed surveillance cameras.

deployment scenario in Changsha city. Considering the number of deployed devices, in this deployment scenario, we choose $RS(5, 3)$ as our EC. It could generate two erasure blocks from three data blocks, so that the surveillance data could be recovered unless more than two devices storing the blocks are destroyed.

With the device location, remaining storage space and network delay, the coordinator can calculate the data block distribution schedule for each device. Table I shows detailed locations of the devices and the result of data block distribution with the density first strategy. For instance, the data blocks of device *A* are distributed to devices *D*, *H*, *J*, *M*, *N*, and device *A* stores seven data blocks for others. From the table, we can see that the data blocks of a local device are distributed to distant devices. We can also see that several devices, such as the devices *E*, *F*, *G*, *H*, are concentrated in a small area, as shown on the map. All these devices tend to distribute their data blocks to some common devices (e.g., *A*, *M*, *N*), which puts a lot of pressure on the storage capacity of these devices. This problem is due to the devices' deployment locations. We have found the same phenomenon when we use different strategies to run GEC.

To test the security of *Centipede* and the effectiveness of GEC, we simulated the geo-range attack in this scenario and compare the possibility of data loss among GEC and two state-of-practice strategies.

1) *K-Means:* This strategy first distribute the cameras into $n+1$ clusters through the K-means clustering algorithm, then each device randomly choose one device in each cluster (except the cluster itself in) as the data storage devices.
2) *Random:* This strategy randomly chooses n devices as data storage devices.

We assume that the attacker starts a geo-range attack roughly from the deployment center with different attack strength [i.e., different α values in (1)]. With each given α value, we gradually expand the range of the attack until all devices are covered. We consider an high threat attack shall have an attack strength $1/e$, otherwise, a local replicate is sufficient to prevent data breach. Meanwhile, here we set the radius of an effective attack area to be at least 20 unit and at most 100 unit. According to the formulation in 1, this can be attained by choosing the α value from 0.01 to 0.05. The experiment results are shown in Fig. 6.

TABLE I
DATA DISTRIBUTION RESULT FROM FIELD EXPERIMENTS

Device	Coordinate	Data Block Distribution	Number of Stored Data Blocks
A	(112.919E, 28.287N)	(D, H, J, M, N)	7
B	(113.017E, 28.141N)	(A, D, H, J, M)	6
C	(112.941E, 28.213N)	(A, E, J, M, N)	2
D	(112.943E, 28.172N)	(A, E, J, M, N)	7
E	(113.024E, 28.214N)	(A, D, J, M, N)	7
F	(112.991E, 28.215N)	(A, D, J, M, N)	1
G	(112.981E, 28.200N)	(A, J, K, M, N)	0
H	(112.990E, 28.219N)	(A, D, K, N, O)	7
I	(113.008E, 28.171N)	(B, D, F, K, O)	0
J	(113.114E, 28.221N)	(B, D, E, H, K)	7
K	(112.949E, 28.242N)	(B, E, H, L, O)	7
L	(112.979E, 28.177N)	(B, E, H, K, O)	4
M	(112.981E, 28.203N)	(B, E, K, L, O)	7
N	(113.010E, 28.133N)	(C, E, H, L, O)	7
O	(112.979E, 28.182N)	(B, C, H, K, L)	6

Fig. 6 shows the number of failed devices with the growth of device numbers covered in the attack range. Note that in this context, a failed device means the remaining block generated by this device is not enough to recover the data generated by it. From Fig. 6, when the attack strength is relatively weak ($\alpha \geq 0.015$), GEC guarantees that all data can be recovered when there are 7 out of 15 devices are covered in the attack range. Even under the strongest attack($\alpha = 0.01$), GEC can still recover all the data when 6 devices are affected. While the K-Means and random strategy both have data loss when only 4 devices are covered. As the number of damaged devices continues to increase, more data cannot be recovered. This is because, in the real world deployment, a large portion of devices are deployed in a close area (like the downtown and the high security area). The data from those devices need to "push" to other distant devices, but a device is limited with storage space. Intuitively, for any given deployment and given attack strength, there should be a threshold on the number of damaged devices, beyond which the system reliability deteriorates quickly.

In Fig. 7, for GEC, we compare the average number of failed devices (i.e., their data cannot be recovered) when different schedule strategies, *distributed*, *storage first*, and *density first*, are used. Note that the exact meaning of these schedule strategies is explained in Section IV-B. From Fig. 7, we can see that the number of failed devices increases as the attack range increases in all three strategies. When the attack range is less than 35, the density first strategy has the best performance. When the attack range reaches 50, the number of failed devices is nearly the same among the three strategies.

B. Performance Evaluation of Data Retrieval

With the geo-aware erasure coding, *Centipede* could assure the effective data retrieval even if one or more surveillance devices are disabled. In this part, we further evaluate the data retrieval performance in a *Centipede* system with 15 devices. We use a personal computer to represent the client end. It has Intel Core i5-6200U CPU, with 8-GB RAM and a Dell Wireless 1820A 802.11ac wireless network adapter. We first

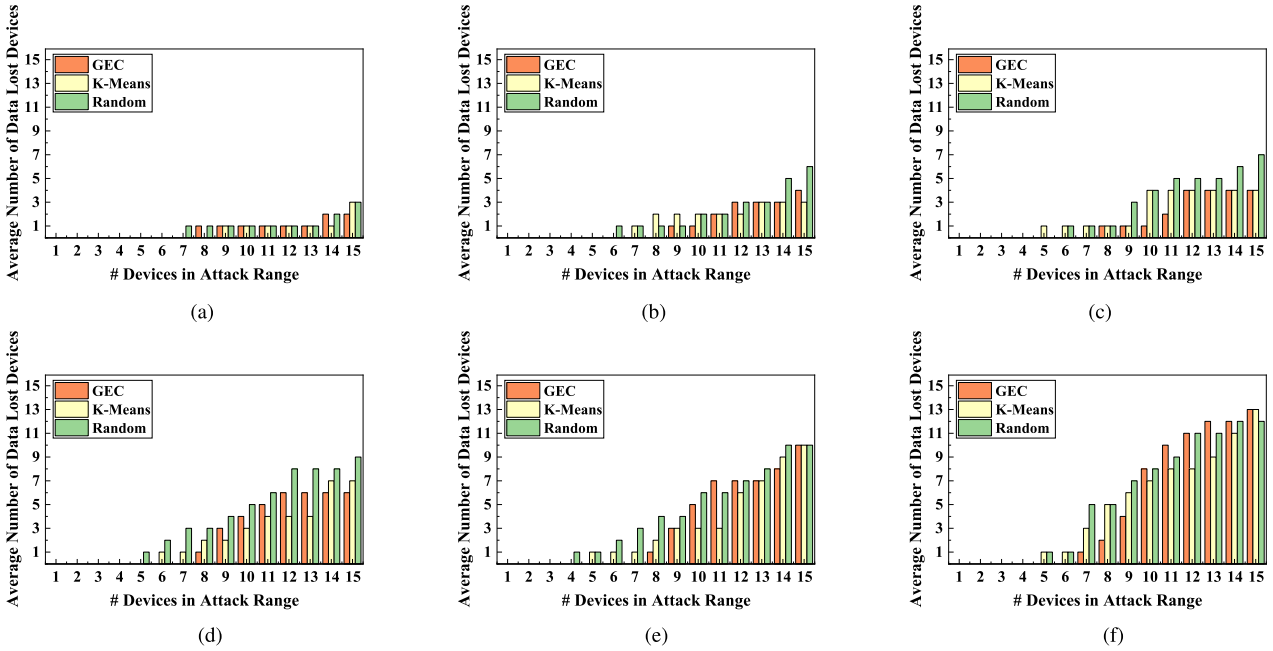


Fig. 6. Comparison of GEC, *K-Means* and *Random* distribution strategy. Note that the attack strength value $\alpha = 0.01$ implies a strong attack [refer to (1)]. (a) Attack strength $\alpha = 0.05$. (b) Attack strength $\alpha = 0.03$. (c) Attack strength $\alpha = 0.025$. (d) Attack strength $\alpha = 0.02$. (e) Attack strength $\alpha = 0.015$. (f) Attack strength $\alpha = 0.01$.

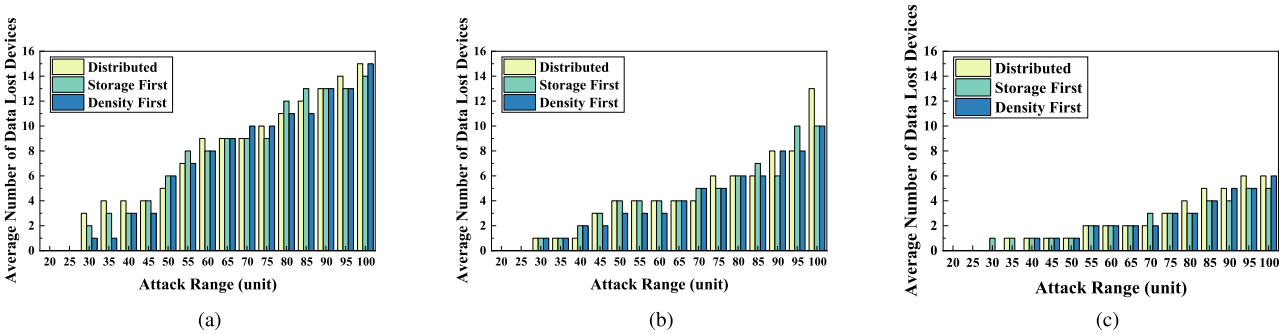


Fig. 7. Comparison of three distribution order (*distributed*, *storage first*, and *density first*) in the field test. Note that the attack strength value $\alpha = 0.01$ implies a strong attack [refer to (1)]. (a) Attack strength $\alpha = 0.01$. (b) Attack strength $\alpha = 0.02$. (c) Attack strength $\alpha = 0.03$.

TABLE II
DATA RETRIEVAL PERFORMANCE

	Time (sec)			Mean Speed (MB/s)
	Segment 1 Size: 677.64MB	Segment 2 Size: 417.89MB	Segmment 3 Size: 623MB	
Onedrive	463.29	286.24	367.04	1.539
WPS Netdisk	488.32	431.21	463.26	1.242
Centipede	340.86	632.73	945.73	0.895

retrieve three pieces of data segments generated by three different devices, then compare the retrieval time of Centipede with two CCS: 1) Onedrive and 2) WPS Netdisk. The results are shown in Table II.

We can see that the retrieval speed of Centipede is on average slightly slower than the tested CCSs. This is because our main intuition is to protect the recording contents from the geo-range attacks, thus would sacrifice some efficiency for availability and confidentiality. Specifically, since the involvement of erasure coding stores data in blocks in distributed camera nodes, the retrieval latency then depends on the nodes with the poorest network situation. We highlight that, compared with the CCS products, our prototype can still provide acceptable efficiency with an average speed of 0.895 MB/s. We deter the optimization of transmission efficiency to the future work.

C. Simulations

With simulation, we can perform a more comprehensive evaluation since we can easily test different scenarios by "deploying" more devices to random places. In our simulations, we set the grid dimension to 100×100. We randomly deploy 100 devices in the area. For each device, the number of data blocks that need to be backup is set to 12.

We use RS code *RS*(12:8) as our erasure coding algorithm. It applies Reed Solomon code to 8 data blocks to generate 4 redundant blocks, so that each data block becomes unavailable only if five or more data blocks are lost. Note that *RS*(12:8)

has been used in various storage systems, e.g., Baidu Atlas cloud storage system [37].

To evaluate *Centipede*, we assume single-point geo-range attacks as well as multipoint geo-range attacks. For the former, the starting point of the attack is randomly chosen in the grid; for the latter, we choose two random starting points in the grid. In addition, we vary the attack strength α and the range of attacks r [refer to (1)]. For each scenario, we run the simulation 100 times and calculate the average number of destroyed devices. In addition, we call that a device has survived an attack if all the data of this device can be recovered. For each scenario, we calculate the average number of survived devices over the simulation runs. Note that a destroyed device can be a survived device if its data can be recovered.

1) Single-Point Geo-Range Attack: Fig. 8(a)–(c) shows the evaluation results in the presence of single-point geo-range attacks with $\alpha = 0.01$,0.015 and 0.02, respectively. We investigate the performance by gradually increasing the range of the attacks. By comparing the Fig. 8(a)–(c), we can tell that as the value of α increases, the number of survived devices goes up. This is because according to (1), a higher α value means a weaker attack. No matter what the attack strength is, the performance of GEC is at least no worse than the other two strategies. *Centipede* with GEC has more than 25% of device survival rate when α is no less than 0.015, while *K-Means* and *Random* strategies can only assure 20% of survival rate. When facing a high strength attack with a larger range (e.g., $\alpha = 0.01$), the GEC only has a very narrow margin. That is because in this scenario most of the devices are covered in the attack range, and the powerful attack can destroy most of the devices within the attack range. In this situation, distribution strategies cannot make many differences. But in practice, this is not likely to happen, because it will take a lot of effort to find and destroy all the devices within a certain range.

By comparing the survival possibility under different attack ranges, we can see that with the highest attack strength ($\alpha = 0.01$), when the attack range is under 35, for all of the three attack strengths, *Centipede* is completely secure with GEC strategy. When the attack range reaches 60, GEC can protect around half of the devices. This is because in this case most storage nodes fall within the attack range and may be destroyed with high probability. With the decrement of attack strength, the device survival possibility is increasing. When the attack strength $\alpha = 0.02$, almost half of the devices can survive. The *K-means* and *Random* strategies have fared less well. In the single-point attack scenario, both of their falling trends appear earlier than GEC. When the attack range is 30, the survived device number drops in both distribution strategies. And they both reach the lowest point of survived device number in a smaller attack range.

2) Multipoint Geo-Range Attack: Fig. 8(d)–(f) shows the evaluation results in the presence of multipoint geo-range attacks with $\alpha = 0.01$, 0.015 and 0.02, respectively. Compared with single point attack, we can see that the number of survived devices in multipoint attacks drops faster than that in single-point attacks. Nevertheless, when the ranges of the attacks are small (e.g., 30), GEC can ensure 100% device survival rate. When the attack range is no more than 45, GEC can protect about 50% of the devices. This is much better than *K-means* and *Random*. But when the attack range rises to 75, there is no significant difference among the three strategies. In this scenario, *K-Means* strategy has better performance than *Random* strategy. With *Random* strategy, there are devices fail when the attack range is 10, while *K-means* can assure all the devices survive when the attack range is under 20 with all attack strength.

VI. Discussion

Based on our evaluation, we prove the following result.

First, *Centipede* can provide strong data protection in the presence of geo-range attacks. For an attack to be effective, the adversaries need to launch a strong attack (i.e., a low value of α) and ensure a sufficient attack range. Simply increasing the attack range or reducing the value of α cannot cause a serious problem to *Centipede*.

The second result is, *Centipede* can provide higher availability and confidentiality than simple onboard solution. With erasure coding and encryption, *Centipede* can recover the surveillance data even if one or more cameras are damaged and prevent unauthorized access to the original data even if the adversaries can get full control of several devices in the network.

Besides, *Centipede* can reduce the video traffic to the cloud. By baking up the surveillance video among the cameras instead of the cloud, *Centipede* keeps most of the surveillance data among the devices locally. With a high-efficiency D2D transmission technique, *Centipede* can fundamentally reduce the transmission pressure of cloud.

However, there are still some open problems.

The first problem is the overall transmission overhead. In this article, our target is to reduce the burden to the cloud and keep most of the video workflow in the device level. The overall transmission overhead is out of the scope of this article. Actually, with $RS(n, k)$ erasure coding, the total size of the data transmitted in the system is n/k times of the original data. But most of these workloads are transmitted among local storage devices but not to the cloud. With the D2D transmission technique, these data can be transmitted effectively among devices without additional burden to the cloud and Internet.

The second problem is the limitation of erasure coding. Though GEC could maximize the security of *Centipede*, if most of the devices fall in the attack range with a high probability of being destroyed, erasure coding cannot guarantee the recovery of original data. The only way to avoid the problem is to back up the data remotely. This is beyond the scope of this article.

The third problem is the *Coordinator* could be a vulnerability of the system. As a virtual component, in this article, we assume it is deployed on the cloud or remotely with high law enforcement. And its security issue is not within the scope of this article.

The fourth problem is the retrieval performance of Centipede. With Geo-aware Erasure Coding, Centipede assures the effectiveness of data retrieval. Though we take the network delay into account when back up the surveillance data, the

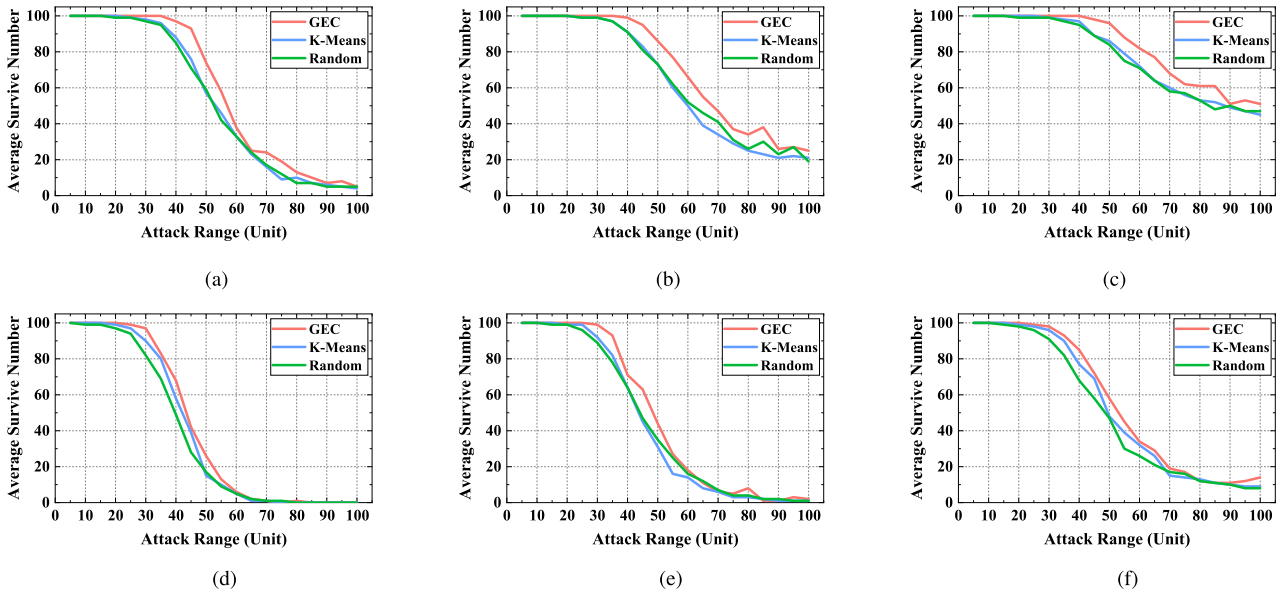


Fig. 8. Average survival number of devices with different strategies under single and multipoint attacks. (a) Single point attack, $\alpha = 0.01$. (b) Single point attack, $\alpha = 0.015$. (c) Single point attack, $\alpha = 0.02$. (d) Multi point attack, $\alpha = 0.01$. (e) Multi point attack, $\alpha = 0.015$. (f) Multi point attack, $\alpha = 0.02$.

retrieval performance could be further improved with better arranged underlying networks.

VII. Conclusion

By designing, implementing, and evaluating *Centipede*, we present a cooperative video data storage system for surveillance cameras to reduce the pressure on cloud server. Compared with simple onboard data storage solution, *Centipede* can protect the surveillance system against geo-range attack. The risk of geo-range attacks is prominent in recent terrorist attacks or nonpeaceful protests. With *Centipede*, we can recover a camera's surveillance data from distributed backup even if the camera is physically damaged by the attacker. With a novel idea called *geo-aware erasure coding*, *Centipede* generates erasure coded data blocks and distributes the encoded blocks to geographically dispersed surveillance cameras. In this way, *Centipede* can enhance the security of the whole system in the presence of single-point and multipoint geo-range attacks. With encryption and erasure coding, it can also prevent unauthorized access to the original data even if the adversaries can get full control of several devices in the network. In the future, we plan to remove the coordinator introduced in *Centipede* to avoid single point failure and develop a fully distributed solution against geo-range attacks.

References

[1] H. Wu, J. Zhang, Z. Cai, F. Liu, Y. Li, and A. Liu, "Toward energy-aware caching for intelligent connected vehicles," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8157–8166, Sep. 2020.

[2] S. Roy. (2020). *China's Didi Uses AI-Powered Facial Recognition Feature to Boost Safety*. [Online]. Available: https://techwireasia.com/2020/01/chinas-didi-uses-ai-powered-facial-recognition-feature-to-boost-safety/

[3] R. Freeman. (2020). *Cameras to Keep a Watchful Eye on Bus Lanes*. [Online]. Available: https://www.punchline-gloucester.com/articles/aanews/cameras-to-keep-a-watchful-eye-on-bus-lanes

[4] M. Hamzei and N. J. Navimipour, "Toward efficient service composition techniques in the Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3774–3787, Oct. 2018.

[5] G. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," San Jose, CA, USA, Cisco, White Paper, 2020.

[6] T. Zhou, B. Xiao, Z. Cai, and M. Xu, "A utility model for photo selection in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 48–62, Jan. 2021.

[7] J. Li, Z. Li, G. Tyson, and G. Xie, "Your privilege gives your privacy away: An analysis of a home security camera service," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 387–396.

[8] S. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, "Erasure coding for distributed storage: An overview," *Sci. China Inf. Sci.*, vol. 61, no. 10, 2018, Art. no. 100301.

[9] Y. Liu, L. Kong, G. Chen, F. Xu, and Z. Wang, "Light-weight AI and IoT collaboration for surveillance video pre-processing," *J. Syst. Archit.*, vol. 114, Mar. 2021, Art. no. 101934.

[10] P. Hu, J. Im, Z. Asgar, and S. Katti, "Starfish: Resilient image compression for AIoT cameras," in *Proc. 18th Conf. Embedded Netw. Sens. Syst.*, 2020, pp. 395–408.

[11] L. D. Chamain, S. S. Cheung, and Z. Ding, "Quannet: Joint image compression and classification over channels with limited bandwidth," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, 2019, pp. 338–343.

[12] S. Chen, S. Zhang, X. Zheng, and X. Ruan, "Layered adaptive compression design for efficient data collection in industrial wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 129, pp. 37–45, Mar. 2019.

[13] A. Al-Dulaimy, W. Itani, J. Taheri, and M. Shamseddine, "bwSlicer: A bandwidth slicing framework for cloud data centers," *Future Gener. Comput. Syst.*, vol. 112, pp. 767–784, Nov. 2020.

[14] H. Tang, F. Liu, G. Shen, Y. Jin, and C. Guo, "UniDrive: Synergize multiple consumer cloud storage services," in *Proc. 16th Annu. Middleware Conf. (Middleware)*, 2015, pp. 137–148.

[15] X. Cai, S. Geng, D. Wu, J. Cai, and J. Chen, "A multi-cloud model based many-objective intelligent algorithm for efficient task scheduling in Internet of Things," *IEEE Internet Things J.*, early access, Nov. 24, 2020, doi: 10.1109/JIOT.2020.3040019.

[16] E. Jinlong, Y. Cui, P. Wang, Z. Li, and C. Zhang, "CoCloud: Enabling efficient cross-cloud file collaboration based on inefficient Web APIs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 56–69, Jan. 2018.

[17] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: Scaling live video analytics with workload-adaptive distributed edge intelligence," in *Proc. 18th Conf. Embedded Netw. Sens. Syst.*, 2020, pp. 409–421.

[18] C. Hung *et al.*, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2018, pp. 115–131.

[19] H. Chen, J. Yu, F. Liu, Z. Cai, and J. Xia, "Archipelago: A medical distributed storage system for interconnected health," *IEEE Internet Comput.*, vol. 24, no. 2, pp. 28–38, Mar./Apr. 2020.

[20] Z. Wei, B. Zhao, and J. Su, "PDA: A novel privacy-preserving robust data aggregation scheme in people-centric sensing system," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 11, pp. 1–9, 2013.

[21] M. R. Fadiheh, D. Stoffel, C. Barrett, S. Mitra, and W. Kunz, "Processor hardware security vulnerabilities and their detection by unique program execution checking," in *Proc. Design Autom. Test Europe Conf. Exhibition (DATE)*, 2019, pp. 994–999.

[22] B. Schneier, K. Seidel, and S. Vijayakumar, *A Worldwide Survey of Encryption Products*. Cambridge, MA, USA: Berkman Center Res. Publ., 2016.

[23] S. Sahraei and M. Gastpar, "Increasing availability in distributed storage systems via clustering," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1705–1709.

[24] X. Xie *et al.*, "AZ-Code: An efficient availability zone level erasure code to provide high fault tolerance in cloud storage systems," in *Proc. 35th Symp. Mass Storage Syst. Technol. (MSST)*, 2019, pp. 230–243.

[25] X. Li, R. Li, P. P. C. Lee, and Y. Hu, "OpenEC: Toward unified and configurable erasure coding management in distributed storage systems," in *Proc. 17th USENIX Conf. File Storage Technol.*, 2019, pp. 331–344.

[26] W. Liang, Y. Fan, K.-C. Li, D. Zhang, and J.-L. Gaudiot, "Secure data storage and recovery in industrial blockchain network environments," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6543–6552, Oct. 2020.

[27] Y. Zou and K. Chakrabarty, "Uncertainty-aware and coverage-oriented deployment for sensor networks," *J. Parallel Distrib. Comput.*, vol. 64, no. 7, pp. 788–798, 2004.

[28] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2809–2816, 2006.

[29] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: Wiley, 1999.

[30] C. Huang, G. Huang, W. Liu, R. Wang, and M. Xie, "A parallel joint optimized relay selection protocol for wake-up radio enabled WSNs," *Phys. Commun.*, vol. 47, Aug. 2021, Art. no. 101320. [Online]. Available: https://doi.org/10.1016/j.phycom.2021.101320

[31] S. Atapattu, N. Ross, Y. Jing, Y. He, and J. S. Evans, "Physical-layer security in full-duplex multi-hop multi-user wireless network with relay selection," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1216–1232, Feb. 2019.

[32] Y. Zhong, T. Han, Q. Li, and X. Ge, "Delay and physical layer security tradeoff in large wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–7.

[33] T. Y. Liu, K. J. Lin, and H. C. Wu, "ECG data encryption then compression using singular value decomposition," *IEEE J. Biomed. Health Informat.*, vol. 22, no. 3, pp. 707–713, May 2018.

[34] A. Al-Ani, M. Anbar, R. Abdullah, A. K. Al-Ani, and S. Al-Mashhadi, "Propose a new approach for securing DHCPv6 server in IPv6 link-local network," in *Intelligent and Interactive Computing*. Singapore: Springer, 2019, pp. 365–376.

[35] E. Erkut, "The discrete p-dispersion problem," *Eur. J. Oper. Res.*, vol. 46, no. 1, pp. 48–60, 1990.

[36] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[37] C. Lai *et al.*, "Atlas: Baidu's key-value storage system for cloud data," in *Proc. 31st Symp. Mass Storage Syst. Technol. (MSST)*, 2015, pp. 1–14.

Jiaping Yu received the M.S. degree in information science and technology from Temple University, Philadelphia, PA, USA, in 2017. He is currently pursuing the Ph.D. degree with the College of Computer, National University of Defense Technology, Changsha, China.

His current research interests include distributed systems, blockchain, and edge computing.

Haiwen Chen received the M.S. degree in computer science and technology from the National University of Defense Technology, Changsha, China, in 2017, where he is currently pursuing the Ph.D. degree with the College of Computer.

His current research interests include distributed systems, edge computing, and machine learning.

Kui Wu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada, in 2002.

In 2002, he joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, where he is currently a Professor. His current research interests include network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.

Tongqing Zhou received the bachelor's, master's, and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 2012, 2014, and 2018, respectively.

He is currently a Postdoctoral Fellow with the College of Computer, NUDT. His main research interests include ubiquitous computing, mobile sensing, and data privacy.

Zhiping Cai (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1996, 2002, and 2005, respectively.

He is currently a Professor with the College of Computer, NUDT. His current research interests include network security and big data.

Prof. Cai is a Senior Member of China Computer Federation.

Fang Liu (Member, IEEE) received the B.S. and Ph.D. degrees in computer science from the College of Computer, National University of Defense Technology, Changsha, China, in 1999 and 2005, respectively.

She is currently a Professor with the School of Design, Hunan University, Changsha. Her current research interests include computer architecture, edge computing, and storage systems.