smart cities

MDPI

Article

BlendSPS: A BLockchain-ENabled Decentralized Smart Public Safety System

Ronghua Xu, Seyed Yahya Nikouei, Deeraj Nagothu, Alem Fitwi and Yu Chen *

Dept. of Electrical and Computer Engineering, Binghamton University, SUNY, Binghamton, NY 13905, USA; rxu22@binghamton.edu (R.X.); snikoue1@binghamton.edu (S.Y.N.); dnagoth1@binghamton.edu (D.N.); afitwi1@binghamton.edu (A.F.)

* Correspondence: ychen@binghamton.edu; Tel.: +1-607-777-6133

Received: 23 July 2020; Accepted: 27 August 2020; Published: 1 September 2020

Abstract: Due to the recent advancements in the Internet of Things (IoT) and Edge-Fog-Cloud Computing technologies, the Smart Public Safety (SPS) system has become a more realistic solution for seamless public safety services that are enabled by integrating machine learning (ML) into heterogeneous edge computing networks. While SPS facilitates convenient exchanges of surveillance data streams among device owners and third-party applications, the existing monolithic service-oriented architecture (SOA) is unable to provide scalable and extensible services in a large-scale heterogeneous network environment. Moreover, traditional security solutions rely on a centralized trusted third-party authority, which not only can be a performance bottleneck or the single point of failure, but it also incurs privacy concerns on improperly use of private information. Inspired by blockchain and microservices technologies, this paper proposed a BLockchain-ENabled Decentralized Smart Public Safety (BlendSPS) system. Leveraging the hybrid blockchain fabric, a microservices based security mechanism is implemented to enable decentralized security architecture, and it supports immutability, auditability, and traceability for secure data sharing and operations among participants of the SPS system. An extensive experimental study verified the feasibility of the proposed BlendSPS that possesses security and privacy proprieties with limited overhead on IoT based edge networks.

Keywords: Smart Public Safety (SPS); microservices; blockchain; smart contract; security; Internet of Things (IoT); Proof-of-Work (PoW); Byzantine Fault Tolerant (BFT)

1. Introduction

Advancement in artificial intelligence (AI) and Internet of Things (IoT) technology makes the concept of Smart Cities become realistic, and these IoT based smart applications have greatly improved the citizen's quality of live and build a safe and sustainable urban environment. However, it brings multiple new concerns as the IoT is widely adopted in smart communities and smart cities. The resource constraint IoT devices need a lightweight application mechanism to perform service tasks, while distributed and heterogeneous network requires a scalable and flexible system infrastructure to support complicated and cooperative operations among participants in smart cities. To enhance the adoption of the IoT in smart communities and smart cities, researchers have been looking into lightweight IoT-based solutions to provide seamless services though integrating heterogeneous computing devices and different types of networks [1,2].

Being considered among the top concerns in the development of smart cities, smart public safety (SPS) facilitates the easy exchanges of surveillance data streams among data owners and third-party service providers. However, it also brings new challenges in architecture, performance, and security. The SPS relies on a distributed network environment consisting of a large number of IoT devices,

and all participants uses their domain independent platforms with high heterogeneity, dynamics and non-standard development technologies. Therefore, system architecture should be scalable, flexible, and efficient to support fast development and easy deployment among participants [3,4]. In addition, to meet the requirements of instant decision making with high accuracy, geographically scattered edge devices collect data at local and share data among service providers of different domains. However, conventional security and management solutions utilize a centralized architecture, which can be a performance bottleneck and is susceptible to a single point of failure. Additionally, SPS combines on-line video stream data from the cameras and with data from off-line sources to perform smart surveillance tasks. Thus, the data in use should be consistent, unaltered, and audible through the entire lifetime. Given a trustless distributed IoT network environment, an ideal SPS framework should be able to support decentralization, immutability, and auditability to ensure security and privacy-preserving data sharing and service operations.

Blockchain, which is a distributed ledger technology (DLT) evolved from Bitcoin [5], has been widely recognized for a great potential to revolutionize the fundamentals of information and communication technology. Blockchain is a natural candidate to enable the decentralized architecture for SPS, in which the data can be securely stored and distributively verified under a peer-to-peer (P2P) network without relying on a centralized trust authority. Such a decentralized architecture provides a prospective option to improve system performance and mitigate single point of failure issues existing in a centralized architecture. In addition, leveraging consensus protocols and public distributed ledgers, Blockchain provides a verifiable, traceable, and append-only chained data structure of transactions. Furthermore, integrating Blockchain into SPS framework not only establishes trust connections among participants, it also guarantees immutability and auditability to ensure data availability, correctness, and provenance.

In this paper, a BLockchain-ENabled Decentralized Smart Public Safety (BlendSPS) system is proposed, which is able to support decentralized, efficient, and secure information sharing and services operations in SPS scenarios. Leveraging the advanced features of the microservices architecture like fine-granularity and loose-coupling, BlendSPS decouples the functionality into multiple containerized microservices. Those computationally affordable microservices can be deployed and run on the resource-constrained IoT devices with limited overhead. A hybrid blockchain fabric is designed as a fundamental security infrastructure to enable a decentralized architecture, and support immutability, auditability, and traceability for data sharing given a trustless distributed IoT network environment.

The major contributions of this paper are as follows.

(1) A novel security-by-design system architecture named BlendSPS is proposed. With given threat models and security goals in SPS, a comprehensive description is presented and underlying rationales are explained.
(2) To address challenges resulted from dynamic and heterogeneity of IoT-based SPS networks, a microservices-enabled security framework is introduced and implemented in a edge-fog computing paradigm.
(3) A hybrid blockchain network architecture is introduced to address trade-offs by adopting blockchain in SPS system. Relying on a two-level consensus mechanism: intra-domain consensus and inter-domain consensus, such hybrid blockchain fabric aims to improve the scalability and efficiency as integrating blockchain with the hierarchical multidomain SPS system.
(4) A proof-of-concept security microservices prototype is implemented and tested on a physical private blockchain setup including Ethereum and Tendermint. The comprehensive experimental results demonstrate that it is practical to run the proposed BlendSPS on IoT-based networks with good performance and security proprieties.

The remainder of this paper is organized as follows. Section 2 reviews background knowledge of SPS systems, and related works in microservices and blockchain solutions. Given discussions on the threat model and security goals in Sections 3 and 4 presents the architecture of BlendSPS including

design rational, key components, and security features. Section 5 illustrates the blockchain-based security mechanism including microservices framework and hybrid blockchain fabric. The prototype implementation and evaluation are discussed in Section 6. Finally, Section 7 concludes this paper with ongoing efforts and future directions.

2. Background Knowledge and Related Work

2.1. Smart Surveillance Systems

With the constant increase in the number of cameras deployed for surveillance purposes, the surveillance community has noticed the demand for human resources to process video stream data to make decisions timely [6,7]. The conventional solutions rely on a cloud computing platform for the pervasive deployment of networked cameras, either static or mobile, which create a huge amount of surveillance data and atomize the video processing [8,9]. Object detection using machine learning (ML) [10] and statistical analysis [11] approaches are of main interest in recent years. Owning to the onerous computation requirement of big data and contextual ML tasks, smart safety surveillance applications are implemented at the powerful server side.

To minimize the role of the human agents, the second generation of surveillance solutions aims to implement various intelligent ML algorithms in decision-making tasks, like object detection [12] and abnormal behavior detection [13] at the centralized cloud. The centralized architecture that needs to merge raw frames from cameras back to the cloud brings a heavy burden on the communication network. To reduce the overhead on the communication channels, context information [14] or query languages [15] has been investigated to promote operators' awareness. Researchers have also proposed to improve the efficiency and throughput of the communication networks with better detection rates, such as reconfiguring the networked cameras [16], utilizing event-driven visualization [17], and mapping conventional real-time images to 3D camera images [18].

However, relying on a centralized architecture inevitably brings uncertain latency and scalability challenges. Decentralized surveillance systems are promising to address aforementioned challenges like limited network access, and more capable to handle mission-critical, delay-sensitive tasks. Advancements in edge-fog-cloud hierarchical architecture enables real-time surveillance [19]. To support the delay-sensitive, mission-critical tasks that often depend on efficient information fusion, quick decision-making, and situation awareness, an urban speeding traffic monitoring system using Fog Computing paradigm was proposed [12]. Merging raw surveillance data streams from drones on near-site fog computing devices can reduce the network traffic created by sending the video to a remote cloud.

To support object assessment method for a SPS system, an Instant Suspicious Activity identiFication at the Edge (I-SAFE) framework was designed leveraging the edge-fog-cloud hierarchy to detect loitering [20]. In I-SAFE, raw frames from surveillance cameras are feed to an edge device where low-level features are abstracted. The fog computing nodes collect features from edge side and perform intermediate-level tasks, including movements recognition, behavior understanding, and anomaly detection [21]. Finally, the cloud focuses on high level tasks of SPS, such as algorithm fine tuning, historical pattern analysis, and global statistical analysis.

2.2. Microservices in IoT

The traditional IoT based service-oriented architecture (SOA) utilizes a *monolithic architecture*, in which service and application software are developed as a single solution deployed on the cloud server. In monolithic application, the service features are developed as distinguishable functionalities. Those functions are module independent and interconnected by the same back-end with a dedicate/set of technology stack, like database. As a result, those earlier monolithic IoT-based applications are low reusable and scalable owing to the manners of tightly coupled dependence among functions and components. Therefore, adopting monolithic framework in a distributed IoT-based network inevitably

brings new challenges in terms of scalability, service extensibility, data privacy, and cross-platform interoperability [22].

Considering as an extension of SOA, *microservices architecture* only encapsulates a minimal functional software module as a fine-grained and independently executable unit, which is self-containment and loose-coupled from remaining system. Unlike monolithic architecture, in which communication between service units relies on inter process communication (IPC), those microservices units are geographically scattered across the network, so that they communicate with each others through a remote procedure call (RPC) manner, such as HTTP RESTful API. Finally, multiple distributed microservices nodes cooperate with each others to perform the complex functionalities of whole system. Microservices architecture achieves *fine granularity* by properly implementing a single dedicated function with minimal development resources. As those fine grained microservices units are independent of each others' developing technologies, microservices architecture is *loose coupling* and flexible to enable continuous development, efficient deployment, and easy maintenance.

Thanks to the advanced properties, like scalability, re-usability, extensibility, easy maintenance, etc., the microservices architecture has been adopted by many smart application developments to improve the scalability and security requirements in distributed IoT-based system. From the perspective of architecture performance and security, the IoT-based applications are advancing from "things"-oriented and centralized ecosystem to a widely and finely distributed microservices-oriented-ecosystem [22]. To enable efficient and security video surveillance services at the edge network including large volumes of distributed IoT devices, a robust smart surveillance systems was proposed by integrating microservices architecture and blockchain technology [1,2,23]. The experimental results verified the feasibility of prototype design to provide a decentralized and fine-grained access control solution for public safety scenario. A similar design is also implemented by BlendSM-DDM [24] to provide a lightweight and decentralized security architecture in IoT-based data marketing systems.

2.3. Blockchain and Smart Contract

As a form of DLT, *Blockchain* initially was implemented as an enabling technology of Bitcoin [5]. Bitcoin aims to provide a cryptocurrency to record and verify commercial transactions among trustless entities without relying on any centralized third-party trust authority, such as financial institutes or government agencies. Blockchain relies on a decentralized architecture, where all participants use a Peer-to-Peer (P2P) network to distributively store and verify data on distributed ledger. To maintain integrity, consistence, and total order of data on the distributed ledger, a *consensus protocol* is executed among a large amount of distributed nodes, called miners or validators. The transactions are collected by miners who can record those valid transactions in a time-stamped block given the consensus algorithm. Finally, all transactions on a blockchain is organized as a verifiable, append-only chained structure of public ledger, in which a new block is identified by a cryptographic hash and chained to a preceding block in a chronological order. Thanks to the consensus protocol, participants can access and verify data on the public ledger that is distributively stored and maintained by "miner-accountants", as opposed to having to establish and maintain trust relationship with a transaction counter-party or a third-party intermediary [25].

Emerging from the intelligent property, *smart contract* (SC) introduces programmability into blockchain to support variety of customized transaction logic rather than simple cash transactions. A SC can be considered as a self-executing procedure stored on blockchain, so that users can achieve predefined agreements among trustless parties through a blockchain network. Leveraging cryptographic and security mechanisms, a SC combines protocols with user interfaces to formalize and secure relationships over computer networks [26]. Contract developer can use programming languages to encapsulate transaction logic and data types into a SC, which is saved at a specific address of a blockchain. Through exposing a set of public functions or application binary interfaces (ABIs), a SC can be invoked on receiving a contract-invoking request from users. Finally, data in SC is updated

given predefined transaction logic or contract agreements. Owing to proprieties like decentralization, autonomy, and self-sufficiency, SC is an ideal solution to provide decentralized application (DApp) in distributed IoT network.

To enabled decentralized security mechanism for distributed IoT-based applications, leveraging blockchain and smart contract has been a hot topic in academic and industrial communities. Some efforts have been reported recently, for example, smart surveillance system [2,23,27], social credit system [28,29], decentralized data marketing [24], space situation awareness [30], multidomain avionic system [31], biomedical imaging data processing [32], and access control strategy [33,34]. Given the aforementioned works, blockchain and smart contract are promising to enable a completely decentralized mechanism to secure data sharing and accessing for distributed IoT-based SPS systems.

3. Threat Model and Security Goals

In this section, we first discuss threats to SPS systems, and then provide security goals that BlendSPS aims to a tackle these threats. Figure 1 illustrates several potential threats to normal operations in an SPS. The players are defined as four roles: camera, edge device, fog server, and human user. The camera generates real-time video streams and transfers them to on-site/near-site edge devices. The edge devices extract lower level features from raw frames, and then send features to a more powerful fog layer for aggregation. The fog server uses collected features to perform higher level analytic tasks, like human behavior analysis and anomalous event detection. The user can query privacy-preserving information from surveillance visualization services based on his/her granted privileges. Note, an edge device can be a single board computer (SBC) that is mounted on the camera, which generates the video streams.

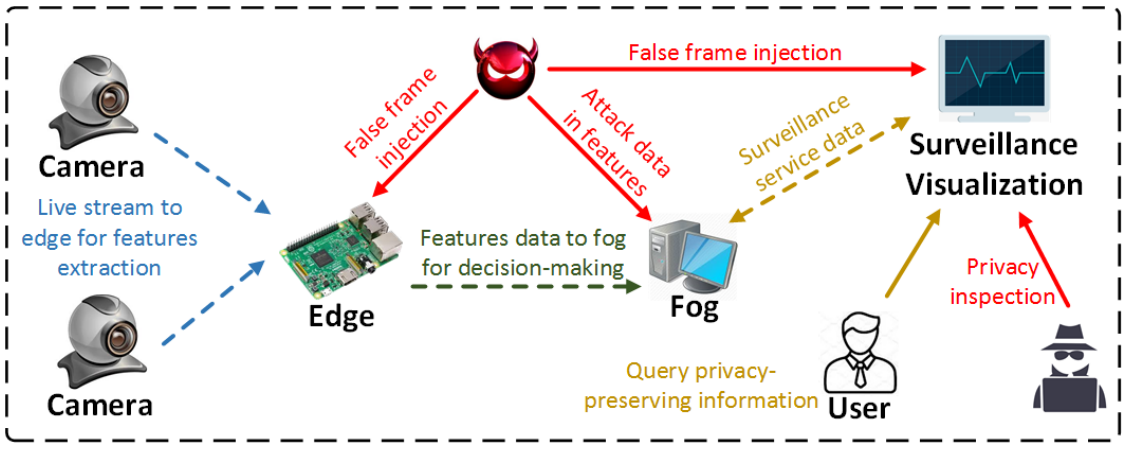

Figure 1. Threat model in smart public safety system.

Threat 1: False Frame Injection Attacks.

The smart surveillance relies on the authenticity of raw video streams from camera to fulfill features extraction and decision-making tasks. However, adversary can launch the visual layer attacks to pose a potential threat to safety and security of the infrastructure [35]. Through false frame injection, attacker can feed fake frames on edge to generate incorrect features. During the decision-making time, the attacker can also replace original frames with duplicate ones in decision-making process to reduce detection accuracy, as shown in Figure 1.

Security Goal 1: False frame detection and verification.

To design an online frame duplication attack detection for SPS system, an environmental fingerprint-based detection technique by using Electrical Network Frequency (ENF) was proposed [36]. ENF is the power supply frequency with a nominal frequency of 50/60 Hz depending on the

geographical location. The fluctuations in the ENF is caused by the power supply–demand and has similar throughout as an electrical grid. The ENF is embedded in both audio and video recordings that are generated by devices running on the power grid. As the similarity of two ENF signals can be measured using a correlation coefficient factor, and it could be used as a fingerprint to detect frame duplication attack.

This paper mainly focuses on false frame verification based on blockchain technology. During false frame detection process, edge nodes not only execute a online ENF-based false frame detection, it also claim ENF fingerprints of frames through transactions that will be recorded in the distributed ledger. Those ENF fingerprints are saved on an immutable distributed ledger that is available to participants in the blockchain network. Therefore, the fog nodes or surveillance visualization service providers can verify those checkpoint frames during decision-making process.

Threat 2: Extracted Features Data Tampering.

In distributed SPS settings, lower level video processing functions are deployed on edge devices, and only extracted features are sent back to the fog nodes for further processing for decision-making. By maliciously tampering with the feature data exchanged between edge and fog, an adversary can distort feature contextualization or change the behaviors in the anomalous event detection.

Security Goal 2: Immutability, Traceability, and Auditability for Data Sharing.

During the video processing, edge devices not only send back extracted features as computation results, but also claim correctness proofs of feature data through transactions that will be recorded in the distributed ledger. The consensus protocol guarantees the immutability and integrity of data in the ledger. The fog node will verify the received features before using them in contextualization and decision-making tasks.

Threat 3: Privacy Violation in Surveillance.

The surveillance visualization provides a spectrum of advanced services, like monitoring traffic flows or deterring crime, etc. However, it also causes people to grow more concerned about the invasion of their privacy as performing mass surveillance indiscriminately irrespective of the individual's private information [37]. The adversary can breach individual's privacy by unauthorized accessing videos or improperly data usage without permission.

Security Goal 3: Decentralized privacy-preserving mechanism.

To protect privacy-sensitive attributes that reveal a lot of information about individuals, like faces, a novel minor privacy protection was proposed by using a face object detector to process collected video streams in real-time at the edge [38,39]. The localized regions of frame will be reversibly scrambled though a lightweight scrambling algorithm. We design a decentralized privacy-preserving mechanism by integrating blockchain with existing sensitive privacy detection and scrambling solution. The privileges definition and access control rules are encapsulated into separate SCs, which are deployed on blockchain network. The surveillance service providers can grant service requests without relying on any third-party authority, and only authorized users are allowed to access the privacy-preserving information without violating the privacy of individuals.

4. Blendsps: Rationale and System Design

Leveraging containerized microservices framework and decentralized blockchain network architecture, our BlendSPS aims to enable efficient, privacy-preserving and secure data sharing, and operations in heterogeneous SPS system. Figure 2 illustrates the BlendSPS architecture that consists of (1) a *hierarchical SPS system framework* that relies on an edge-fog computing network to support a distributed smart surveillance as an edge service, (2) a *blockchain-enabled security service layer* that enables lightweight and decentralized security policies, and (3) a *hybrid blockchain fabric* that

ensures decentralization and security properties in SPS system. The rationale behind the design of the BlendSPS is described as follows.

- Hierarchical SPS framework is considered as the upper-level applications layer that provides SPS services on a heterogeneous network environment, like smart video surveillance, visual layer attack protection, privacy-preserving video stream accessing, etc.
- Blockchain-enabled security service layer functions as the intermediate level infrastructure that integrates containerized microservices with blockchain to support a scalable, flexible, and lightweight security mechanism. As a lightweight virtualization technology, containers have features, like platform independence, resource abstraction and OS-level isolation. Therefore, containerized microservices architecture is an ideal design for heterogeneous IoT-based SPS systems.
- Hybrid blockchain fabric provides a fundamental networking and security infrastructure to ensure decentralized security enforcement for the SPS system. Leveraging hybrid consensus mechanism and secure public distributed ledger, blockchain fabric brings security features, like immutability, auditability and traceability, to efficiently enhance the security issues of existing SPS systems.

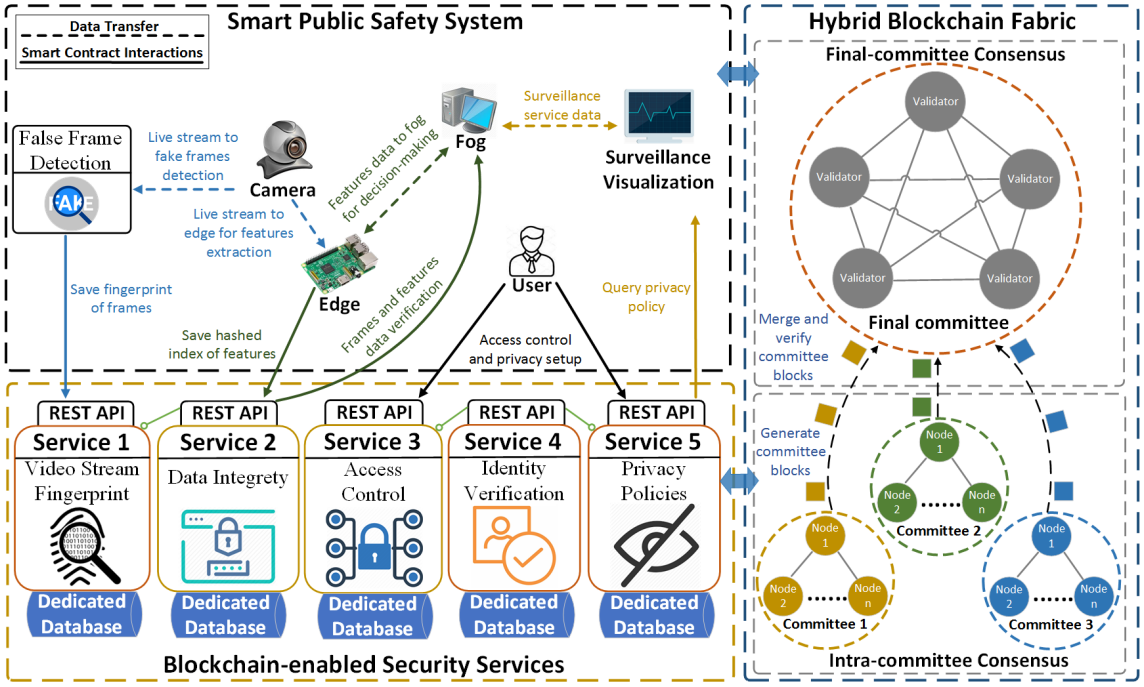

Figure 2. Architecture of BLockchain-ENabled Decentralized Smart Public Safety (BlendSPS).

4.1. Hierarchical Sps Framework

The left top part of Figure 2 demonstrates an user scenario of a SPS which includes three key elements: smart video surveillance, visual layer attack protection, and video stream privacy preservation. Video streams are collected by cameras and transmitted to microservices in real-time on edge devices for feature extraction. The lower level features are extracted by edge devices and are transferred to more powerful fog nodes where data aggregation and higher level analytic services, such as human behavior analysis and anomalous even detection, are conducted. To prevent visual layer attacks on raw video streams, ENF-based false frame detection microservices are responsible for authenticating on-line video stream. Moreover, extracted environmental fingerprint of frame is securely recorded into immutable distributed ledger for decentralized off-line verification. The privacy-conserving surveillance visualization ensures that only authorized user could access

sensitive information, and video frames containing user's privacy, such as faces or living areas, are marked and hidden from the public according to specified privacy policies of individuals.

4.2. Smart Video Surveillance

Unlike most conventional Deep Neural Networks (DNN) based smart video surveillance solutions that are implemented in a monolithic architecture, we break the whole surveillance process into multiple smaller sub-tasks that can be deployed and executed independent of the rest of the system. The classification of the human behavior is divided into two steps: extracting features for each individual from raw video streams, and then making a decision based on the handpicked features. Figure 3 shows the microservices based architecture adopted by our video surveillance on a edge-fog computation hierarchy model.

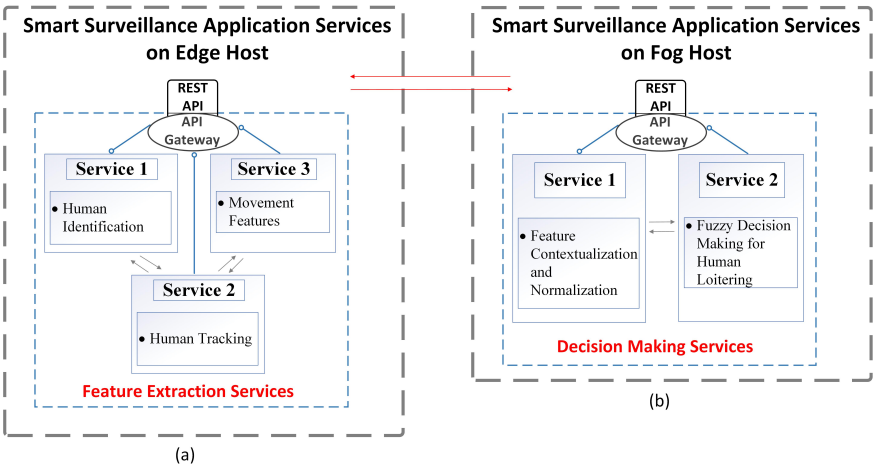

Figure 3. Smart Video Surveillance Microservices: (**a**) Services at Edge; (**b**) Services at Fog.

The lower level tasks, like video processing and object features detection, are performed on the edge side. Figure 3a shows the connection of the microservices implemented on the edge node, along with their connections to the fog device for video processing. The raw frames are fed to a microservice that detects the objects of interest and extracts the location of each of them. Then another tracking algorithm, optimized for accuracy and speed, is executed to track the aforementioned detected object in frames of video stream. Given individual's tracking data, the edge node extracts a set of features to identify a pattern of individual's movement based on the object position history. The extracted pattern features are then sent to the corresponding fog node for classification. To mitigate attacks on extracted features during prorogation and sharing process among edge and fog, the edge device also generates an authenticator of features and records it on the blockchain. The fog server can verify the received features by querying authenticator from blockchain, then use the valid ones in decision-making.

The decision-making microservices are deployed on the more powerful fog side, which is responsible for the feature contextualization and target behavior classification, as shown by Figure 3b. Contextualization helps to have better feature representation when two sets of features have similar values. For example, in a campus environment it is considered normal to detect people in the hallways, while it is highly suspicious to detect anyone staying in the same hallway for hours. Thus, time may be a very valuable indicator to help interpret the features. Training a classifier to detect human behavior requires huge amount of data, and the detection accuracy highly depends on the quality of the training set [20]. To reduce the need for a complete training data set, we use a fuzzy model to make a decision based on the walking pattern for each of the individuals' information sent by the edge node. Readers interested in details of study on Convolutional Neural Network (CNN)-based objects tracking and fuzzy decision-making for suspicious activity identification are referred to the works in [20,21].

4.3. Enf-Based False Frame Detection

To protect against visual layer attacks in video surveillance, an ENF-based false frame detection method is designed to detect false frame injection during online video stream generation time at camera side. The presence of ENF fluctuation traces in multimedia recordings comes from the source of ENF signal by power system. Thus, comparing the ENF signals between power and multimedia recordings can authenticate original digital video or audio sources. Figure 4 illustrates the difference of the ENF fluctuation traces from original video stream, power grid, and the attacked video recording.

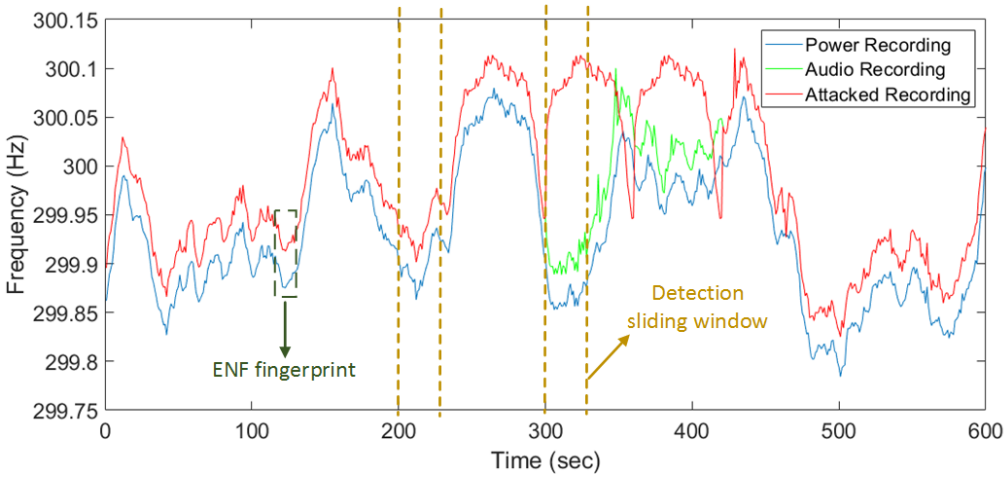

Figure 4. False frame detection by comparing Electrical Network Frequency (ENF) signals from power, original, and forged recording.

The estimated signal from both the power recordings and the audio recordings from the surveillance system are compared using correlation coefficient [40]. We adopt a sliding window-based mechanism to achieve an efficient online detection task. For each window, a 30 s recording is used for ENF estimation, and upon correlation comparison, a step of five seconds is used with an overlap of 25 s. The sliding window approach can reduce the delay in detection of the replay attack and consumes less computational power. As Figure 4 shows, the ENF of duplicated recordings is mismatched with the ENF of power recordings as sliding window comes, the correlation coefficient for duplicated recordings will drop and the false frame injection attack is detected. Readers interested in a detailed study of ENF application in digital media forensic analysis are referred to the work in [36].

Because ENF is considered as an environmental fingerprint, the estimated ENF signals from multimedia recordings can also be used as an authenticator for offline verification on surveillance data, like video or audio streams. In case of the false frame detection, a section of recordings can be used as checkpoints, as Figure 4 shows. Then ENF signals extracted from the checkpoint are saved on the blockchain. Other users, like fog server or surveillance visualization, can query ENF fingerprints from blockchain, then verify them before using raw video streams.

4.4. Privacy Preserving for Video Stream

To enable surveillance visualization without violating the privacy of individual person captured in the videos, a privacy preserving mechanism is designed by integrating lightweight sensitive privacy-attributes detection methods, scrambling technique and blockchain technology. The non-compute intensive object detection algorithm is responsible for classifying and localizing those objects on video frames which are deemed sensitive. Then, the scrambling technique reversibly masks those sensitive objects or regions detected by the object detection scheme. To illustrate how privacy preserving mechanism works, we use a test picture from the images of groups dataset [41]. Figure 5 presents an example of children privacy protection. Figure 5a shows that children's faces are accurately detected and bounded through a face-detection algorithm. In Figure 5b, the faces of the two

children are enciphered following the scrambling process to hide their identity in case of unwarranted disclose. Readers interested in a detailed study of face identification and scrambling technologies for privacy protection are referred to the work in [39].

To access video or analytic results from surveillance visualization, users must interact with blockchain-enabled security services to verify privacy rules and gain proofs that they have the right approbation, as Figure 2 shows. The data owners or service providers deploy SCs that define privileges and privacy preserving rules on blockchain. These decentralized security services ensure that only authorized users can successfully access privacy-preserving information given their access right and privacy specification.

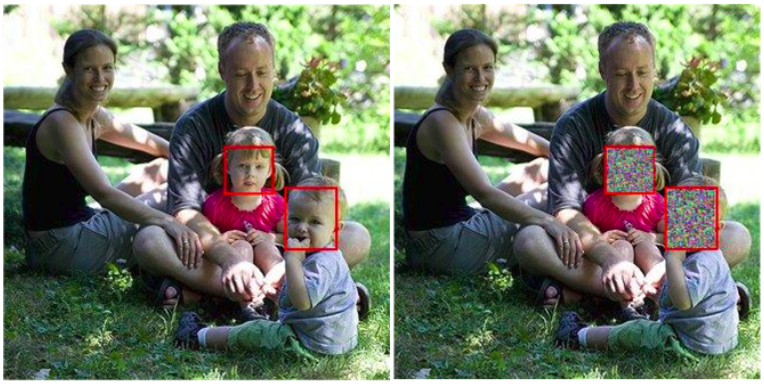

Figure 5. Sensitive attributes detection and privacy protection.

5. Blockchain-Based Security Mechanism

The blockchain-based security mechanism leverages lightweight microservices architecture and hybrid blockchain fabric to ensure decentralization, security, and privacy proprieties for the BlendSPS system. This section provides detail explanations on two parts: upper layered microservices-enabled security service architecture and the underlying hybrid blockchain fabric.

5.1. Microservices-Enabled Security Service

The microservices-enabled security services layer functions as a fundamental microservices oriented service architecture to support the decentralized security and privacy proprieties in the BlendSPS system, as shown by the left bottom part of Figure 2. Each containerized microservices node exposes a set of web-service APIs to upper application layer and uses the local RPC interfaces to trigger self-executing procedures defined by SC. The key design ideas and operations are described below.

5.1.1. Video Stream Fingerprint

The ENF-based false frame detection uses a sliding window-based method to obtain and estimate the ENF fingerprint from video records online. This real-time detection mechanism ensures that the raw data from the camera is authenticated. However, insecure communication channels or modification by other service providers also jeopardize the integrity of the original data. Based on the blockchain network, a decentralized video stream data auditing strategy is design to protect against false frame injection attacks in the BlendSPS system. By recording the extracted ENF fingerprint data into a distributed ledger, an intra-committee that includes a small number of validators executes a lightweight consensus protocol to ensure the sanctity of the data stored on the ledger. Therefore, any participants from intra-committee can verify the immutable ENF data without relying on a centralized third-party trust authority.

The video stream fingerprint audition procedures are presented in Algorithm 1. As two functions used in real-time false frame detection, the *extract_ENF_signal()* at line 2 is responsible for extracting

ENF signal given input of *frames* data, and the *correlation_ENF_signa()* at line 8 outputs the similarity *coef_ENF_fingerprint* by using a correlation coefficient between the two sampled signals. The ENF fingerprint recording and verification procedures use a set of RPC interfaces exposed by validators to interact with distributed ledger. In *record_ENF_fingerprint(frames)*, video stream owner first calls *extract_ENF_signal()* to get ENF signal by feeding checkpoint frames. Then, he/she launches a transaction (tx) request by calling *transaction_commit()* RPC to record *ENF_fingerprint_tx* into distributed ledger. Finally, tx_result will be returned to notify the feature owner as long as $ENF\_fingerprint\_tx$ has been recorded and confirmed in a new block.

Algorithm 1 The video stream fingerprint audition procedures

```
1:  procedure: record_ENF_fingerprint(frames)
2:      ENF_fingerprint_tx ← extract_ENF_signal(frames)
3:      tx_result ← transaction_commit(ENF_fingerprint_tx)
4:      return tx_result
5:  procedure: verify_ENF_fingerprint(frames)
6:      ENF_fingerprint_tx ← extract_ENF_signal(frames)
7:      query_ENF_fingerprint_tx ← transaction_query(ENF_fingerprint_tx['id'])
8:      coef_ENF_fingerprint ← correlation_ENF_signal(ENF_fingerprint_tx, query_ENF_fingerprint_tx)
9:      if coef_ENF_fingerprint < coef_threshold then
10:         verify_ENF ← True
11:     else
12:         verify_ENF ← False
13:     end if
14:     return verify_ENF
```

The users who utilize video stream in their task will execute *verify_ENF_fingerprint(frames)* to verify these checkpoint *frames*. Simply by calling *transaction_query()*, the user can fetch recorded *query_ENF_fingerprint_tx* from the distributed ledger for further verification process. Given comparison between *coef_threshold* and *coef_ENF_fingerprint*, which is evaluated by calling *correlation_ENF_signal* function, the user can verify whether or not the checkpoint frames are generated by original owner. The false frame injection attacks can be detected.

5.1.2. Data Integrity

As feature data extracted at edge devices are transferred to fog nodes, it is necessary to ensure data integrity for decision-making. Relying on the blockchain network, a data integrity scheme based on hashed features authentication is designed to enable a decentralized and secured features sharing among participants in the BlendSPS system. The distributed ledger is ideal to enable an immutable and traceable storage. However, directly putting a huge amount of features data into a transaction brings more communication and computation cost by transaction propagation and verification. In addition, larger transaction size means fewer committed transactions per block, it also reduces transactions rate given fixed block size.

To ensure efficient data recording, access, and verification, only a fixed length string of hashed features is saved on the distributed ledger instead of the raw data. For a set of features F_i, the string of hashed features is calculated as $hash\_F_i = \mathcal{H}(F_i)$, where $\mathcal{H}(\cdot)$ is a predefined collision-resistant hash function that outputs a binary hash string $h \in \{0,1\}^\lambda$ with the length λ. The $hash\_F_i$ will be put into a transaction that is recorded on the distributed ledger. Any participants can query $hash\_F_i$ from the distributed ledger as an authenticator for verification process.

The hashed features authentication procedures are presented by Algorithm 2. The $hash\_feature()$ function is responsible for computing hash string given the input of $features\_set$. First, all parameter vectors of each feature line are converted to the string format and combined as a $string\_features$, as shown from line 3 to line 6. Then, line 7 converts a $string\_features$ to a binary string $bytes\_features$. Finally, a cryptographic one-way hash function outputs a fix length of the hash string $feature\_hash$ given input of $bytes\_features$, and a dictionary $\{feature\_id : feature\_hash\}$ will be returned.

Algorithm 2 The hashed features authentication procedures

```
function: hash_feature(features_set)
    string_features ← [empty_string]
    for feature_vector in features_set.items do
        feature_string ← Convert_to_string(feature_vector)
        string_features ← (string_features feature_string)
    end for
    bytes_features ← Convert_to_bytes(string_features)
    feature_hash ← Convert_to_hash(bytes_features)
    feature_id ← features_set.name
    return {feature_id:feature_hash}
procedure: record_hashed_feature(features_set)
    feature_tx ← hash_feature(features_set)
    tx_result ← transaction_commit(feature_tx)
    return tx_result
procedure: verify_hashed_feature(features_set)
    feature_tx ← hash_feature(features_set)
    query_feature_tx ← transaction_query(feature_tx)
    if query_feature_tx == feature_tx then
        verify_hash ← True
    else
        verify_hash ← False
    end if
    return verify_hash
```

The hashed feature authentication procedures interact with as set of RPC interfaces of the blockchain client to record and query data on the distributed ledger. In the $record\_hashed\_feature()$ procedure, the feature owner first computes a hashed feature dictionary $feature\_tx$ as a transaction data by calling $hash\_feature()$ function. Then, $record\_hashed\_feature()$ RPC is called to record $feature\_tx$ into the distributed ledger. As $feature\_tx$ has been recorded and confirmed on the distributed ledger, tx_result will be returned to notify the feature owner.

The $verify\_hashed\_model()$ procedure is performed by entities who utilize these features data, like fog layer service in the contextualization and decision-making processes. Through executing $hash\_feature(features\_set)$, $feature\_tx$ of currently verifying $features\_set$ is calculated as the key index for querying data in the distribute ledger. The user simply calls $transaction\_query()$ RPC to fetch the recorded $query\_tx$ as proof in verification. Given a comparison between $query\_feature\_tx$ and $feature\_tx$, the user can verify whether or not the $features\_set$ is authentic.

5.1.3. Identity Verification and Access Control

As each blockchain account is uniquely indexed by its address, which is derived from the public key, the blockchain account address is ideal for identity authentication needed by other security microservices, like data integrity and access control. Identity authentication relies on a virtual trust zone that is ensured by a blockchain network, and each entity records its account address into the blockchain as a virtual identity (VID) for identity verification.

The identity verification procedure is presented in Algorithm 3. The identity verification is triggered as a host receives a service request from the client, like access control or privacy policies services. The host calls RPC function *get_VNodeInfo()* to query the recorded Virtual Node (VNode) information from the SC. Then, it checks if client has the same VZoneID as the host does and returns the identity verification results *verify_ID*. Readers interested in a detailed study of VID based identity authentication are referred to the work in [30].

Algorithm 3 The identity and access control verification procedures

```
1:  procedure: identity_verification(client_ID)
2:      host_ID ← get_Account()
3:      json_VNode_host ← get_VNodeInfo(host_ID)
4:      json_VNode_client ← get_VNodeInfo(client_ID)
5:      if json_VNode_host['VZoneID'] == json_VNode_client['VZoneID'] then
6:          verify_ID ← True
7:      else
8:          verify_ID ← False
9:      end if
10:     return verify_ID
11: procedure: access_control_verification(client_ID)
12:     json_access_data ← get_AccessToken(client_ID)
13:     if json_access_data['isValid'] != True then
14:         return False
15:     end if
16:     if (json_access_data['issuedate'] > current_time) or (json_access_data['expireddate'] < current_time) then
17:         return False
18:     end if
19:     for access_right in json_access_data['authorization'] do
20:         if is_access_valid(access_right) != True then
21:             return False
22:         end if
23:     end for
24:     access to service or data is permitted
25:     return True
```

To enable a decentralized access authorization and verification, a decentralized capability-based access control mechanism is integrated [34]. Data owners can implement access control models and policies as a SC based access control (AC) microservices entity. In initial, an user must send an access authorization request to the AC microservices to get a capability token before requesting services or resources at SPS service providers. Given predefined access authorization policies, AC microservices put authorized access right into capability token which is saved in the SC.

The access control verification procedure is explained from line 11 to 24 in Algorithm 3. Once a service request is received from a user, the service provider calls *get_AccessToken()* to fetch the user's access token, then checks if the AC token *json_access_data* satisfies the valid conditions. If the AC token is valid, the service provider verifies whether user's access request is permitted through comparing every access right item saved in the AC token. Otherwise, verification process aborts and the service request is denied. If the service request is permitted by AC token, *access_control_verification()* outputs *True*, and then the service provider grants the user's access to data or service. Otherwise, service request is denied.

5.1.4. Privacy Policies

The privacy-preserving microservices is applied mostly for privacy-sensitive data management. Therefore, the privacy-sensitive data is not accessible or even not visible to unauthorized entities. Data integrity service ensures that only hash strings of sensitive data are recorded on the blockchain for authenticity checking, while raw date is encrypted and stored off the chain. Hence, data privacy is protected during the transmission and storage time. In addition, AC service programs access control rules as SCs, and therefore the access authorization and verification can be executed automatically. The decentralized AC service can effectively prevent unauthorized access to sensitive data. Furthermore, the privacy policies can be securely stored on the blockchain, according to which a data or service requester is aware of his/her privileges to access the sensitive data.

With above mechanism, the data owners are allowed to adjust their access control and privacy policies flexibly. Only authorized users are assigned access to surveillance services without violating

the privacy of individuals. The privacy policies service relies on existing security services, like AC and identity verification, to enabled a decentralized privacy-preserving surveillance visualization. As Figure 2 shows, the surveillance visualization firstly interacts with privacy policies microservices, which could query privacy rules based on user's identity. Then, the user can fetch the surveillance service data, like video streams or detection results, according to user's permissions defined by AC services. Finally, given user's privacy policies, the scrambling contents and objects in video steams are visualized to users without revealing information pertinent to individuals' privacy.

5.2. Hybrid Blockchain Network Architecture

The BlendSPS utilizes a hierarchical edge-fog computing paradigm, in which each layer has different performance, security and privacy requirements. The cameras and edge devices are deployed on the synchronous and permissioned edge network that is managed by a domain administrator. Therefore, lightweight design and high throughput become key matrices as running the consensus protocol. Meanwhile, decision-making tasks and surveillance services are deployed on the fog computing layer, which requires data sharing and operations across domain boundaries and relies on an asynchronous network environment. Thus, scalability and security are the key matrices as choosing consensus algorithm. It is hard to optimize the trade-offs among performance, scalability, and security by integrating a single consensus mechanism into the BlendSPS system.

To handle the aforementioned issues as performing consensus algorithms in a distributed BlendSPS network that is highly heterogeneous and dynamical, the hybrid blockchain fabric adopts a two-level consensus mechanism: intra-domain consensus and inter-domain consensus, as the right part of Figure 2 shows. Considering a local domain that includes a small number of cameras and edge devices, a lightweight but efficient intra-committee consensus protocol is executed among specified committee members to validate transactions within the domain and maintain a local distributed ledger. For multidomain operations, like recording hashed features and updating access token, a scalable and security inter-domain consensus protocol is responsible to finalize those inter-domain transactions on a global distributed ledger. The design rationale and workflows are explained as follows.

5.2.1. Permissioned Network

The SPS system is deployed on a permissioned network, where every entity must register its unique identity information to the system administrator; thus, only authorized nodes can join the network. As permissioned network provides basic security primitives, like public key infrastructure (PKI), digital signature, etc., it enhances security proprieties of blockchain from network infrastructure prospective. For a local domain, domain owner chooses a subset of the nodes as an intra-domain committee, and only validators from the committee can execute the intra-domain consensus protocol, launch transactions and maintain the shared local ledger in the private blockchain network.

To securely record the cross-domain transactions in the SPS system, a consortium blockchain network is used by participants from different domains. Given the computation capacity of devices and the security policy, the SPS system administrator specifies all participants as miner or node (non-miner). Unlike the private blockchain network adopted by local domains, both miners and nodes in the SPS system can send transactions and access data on the global ledger. However, only authorized participants can work as miners to execute the inter-domain consensus protocol.

5.2.2. Intra-Domain Consensus

Given a private blockchain network managed by a local domain owner, a lightweight Byzantine Fault Tolerant (BFT) based consensus protocol is executed by validators of the intra-domain committee. The BFT consensus comes from classical Byzantine General Problem [42], and it aims to achieve a single value agreement among n geographically distributed and inter-connect participants given failures of partner or conflicting information. Considering a Byzantine failure scenario, there are f dishonest nodes who attempt to break the consensus agreement by sending contradicting values to other nodes.

If a super-majority of participants ($n - f > 2f$) are honest, they can still agree on the consistent actions. Thus, BFT consensus ensures the ultimate goal of agreement if a network includes $n \geq 3f + 1$ total nodes and only f are Byzantine ones.

The BFT consensus protocol can achieve high throughput in a small-scale consensus network and it introduces a low computation overhead as executing the consensus algorithm on hosts. Therefore, the BFT-based consensus protocol is suitable for the intra-domain committee at a distributed edge network. For recording data on the local ledger, a user sends data transactions to a validator within the intra-domain committee. Then, the validator verifies received transactions and broadcasts valid ones to other validators. Each validator collects valid transactions and records them in a new block given a block generation algorithm. If the proposed block is verified and confirmed by no less than 2/3 of validators, the consensus agreement is achieved by finalizing recorded data in block on the distributed ledger. As intra-domain consensus protocol is only executed among a small-scale of committee members, communication cost incurred by messages propagation is reduced.

5.2.3. Inter-Domain Consensus

Inter-domain operations rely on a consortium blockchain network and it inevitably runs into critical issues in open-access and asynchronous network environments. The inter-domain consensus protocol adopts a scalable Proof-of-Work (PoW) mechanism to enable a probabilistic finality on inter-domain transactions. In PoW consensus, each miner must exhaustively query a cryptographic hash function to gain a hash code as a work proof for new block generations. The PoW mining process can be formally defined by the following equation:

$$hash\_block = \mathcal{H}(block\_data nonce) \leq D(h), \tag{1}$$

where *nonce* is a random number used to calculate the candidate *hash_block*, $D(h) = 2^{L-h}$ is a difficulty condition specified by a certain length of bits h as parameter, and $\mathcal{H}(\cdot)$ is a predefined collision-resistant cryptographic hash function that outputs a fixed λ length of hash string $L \in \{0,1\}^{\lambda}$.

If the *hash_block* of a candidate block satisfies a pre-defined difficult condition defined in Equation (1), the miner broadcasts the candidate block to peers and appends it on the local chain. Every node follows a message gossiping rule to multicast valid transactions and blocks to peers rather than the whole network. All honest nodes only accept valid blocks and always extend blocks on the longest chain that they have ever synchronized from the network. Such a longest chain rule can effectively mitigate fork issues in an asynchronous network and ensure that all honest miners are working on a common main chain. The security of the inter-domain consensus is ensured if majority (51%) of the miners are honest and correctly execute the consensus protocol.

6. Experimental Results

To verify the feasibility of the BlendSPS scheme, a proof-of-concept prototype is built and tested in a real physical network environment. The Docker is adopted to develop microservices framework, and those containerized microservices units can be deployed both on the edge (Raspberry Pi) devices and fog (desktop) server. The security services are implemented in Python with Flask [43] as web-service framework. For the blockchain network, we use Ethereum [44] to build inter-domain blockchain network, and use Tendermint [45] to develop intra-domain blockchain network. The Solidity [46], which is a contract-oriented and high-level language, is used for developing SCs. We use RSA for asymmetry cryptography, like digital signature, and SHA-256 for hash function, which are developed using standard python lib: cryptography [47]. All documents and source code are available on the BlendSPS project repository [48].

6.1. Experimental Setup

As prototype implementation and test cases design are mainly to verify performance and security proprieties provided by BlendSPS, the experimental set-up focus on security functions related configuration. For private Ethereum network, six miners are deployed on six separate desktops, and all nodes use Go-Ethereum [49] as the client application to interact with Ethereum network. The Tendermint are running on a 20-validators test network, where each validator is hosted on a Raspberry Pi (RPi) device. All desktops and RPi devices are connected through a local area network (LAN). Table 1 shows configurations of devices used for the experimental study.

Table 1. Configuration of experimental devices.

Device	**Redbarn HPC**	**Dell Optiplex 760 Desktop**	**Raspberry Pi 3 Model B+**
CPU	3.4GHz, Intel(R) Core(TM) i7-2600K (8 cores)	3GHz, Intel(R) Core(TM) E8400 (2 cores)	1.4GHz, Broadcom ARM Cortex-A53 (ARMv8)
Memory	16GB DDR3	4GB DDR3	1GB SDRAM
Storage	500GB HHD	250G HHD	32GB (microSD card)
OS	Ubuntu 18.04	Ubuntu 16.04	Raspbian GNU/Linux (Jessie)

In security service simulation test, Redbarn HPC acts as a system oracle that provides security basics, like PKI and registration for network management. The oracle can only manage permissioned network by adding and removing participants. The validators can only update and verify data and transactions on the distributed ledger rather than changing the permissioned network configuration. All desktops can work as fog computing nodes, and RPi devices run as edge computing nodes. The security microservices are deployed both on edge and fog layers for experimental test.

6.2. Performance Evaluation

To evaluate the performance of operating microservices-based security schemes, a set of experiments is conducted on our prototype blockchain private networks by simulating service transactions, like access right token generation, identity verification, etc. The cost of message encryption and decryption are not considered during the test.

6.2.1. Microservices Overhead: Computation Overhead and Network Latency

A service request experiment, which includes five RPi devices and four desktops, is designed to evaluate the overhead incurred by the security microservices on the host machines. Four types of microservices are deployed on four RPi devices and four desktops separately, and each machine only runs a single microservices node. One RPi device functions as a client that sends service request to these security service providers. One-hundred test runs have been done in total based on the proposed test scenario, where a client initiates the connection by sending a request for service to the server side for an access permission.

Figure 6 shows the computation overhead for hosting individual microservices node on edge and fog platforms. The results reveal that computation overhead increases as the complexity of the tasks grows. As video stream fingerprint relies on a lightweight Tendermint to record and verify the ENF fingerprint data, it needs less processing time than other security services , which require more computational resource by SC operations. Unlike data integrity, identity verification and AC microservices need more cryptographic computations and authentication operations. Therefore, they require higher processing time both on the RPi device and the desktop. Due to multiple SC interactivity, identity verification microservice takes largest processing time for querying the data in blockchain.

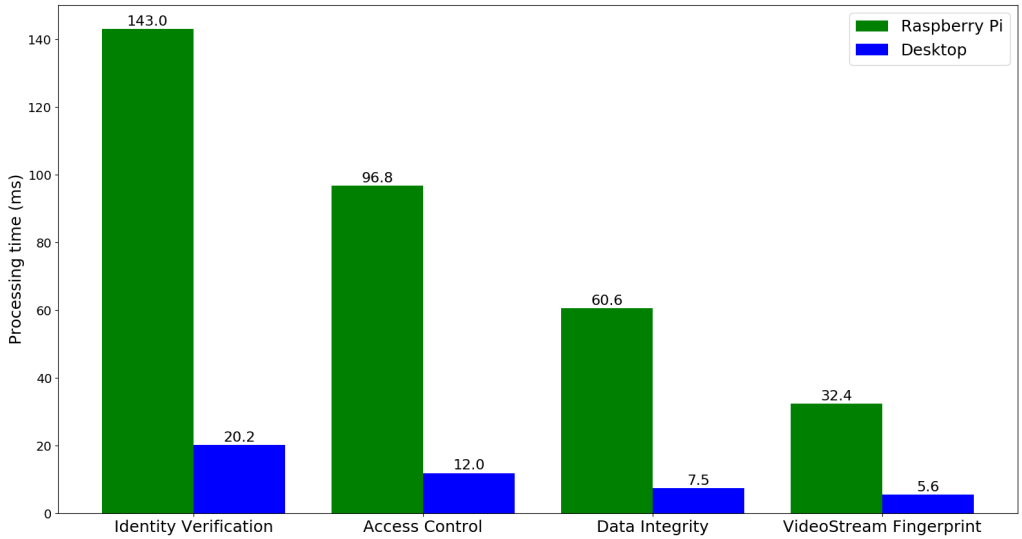

Figure 6. Processing time of security microservices on different host platform.

The end-to-end delay is evaluated based on the test case that a client sends multiple service transactions per second (TPS) and waits until that all responses are received. Figure 7 shows network latency of running security microservices as send transaction rate varies from 1 to 100 TPS. In terms of the bandwidth of network and capacity of the server side, the time latency of sending transactions and receiving all acknowledgments is almost linear scale to the transaction rate. Considering the same networking environment and transaction data size, the influence of communication cost is almost negligible. Therefore, the computation cost on the server side becomes dominant as scaling multiple transactions during single microservices node scenario.

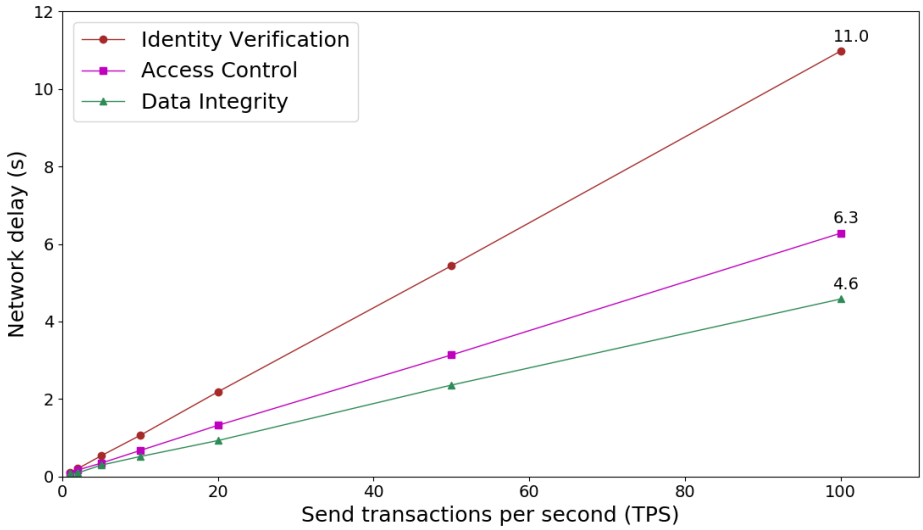

Figure 7. Network latency of accessing security microservices with different transaction rate.

6.2.2. Throughput Evaluation: Microservices vs. Monolithic Framework

To evaluate the network delay influence between microservices and monolithic framework as scaling multiple transactions, a set of comparative experiment is conducted on two service demo applications: Micro_App and Mono_App. Micro_App uses microservices framework in which five containerized security microservices are deployed on five RPi devices separately, while Mono_App relies on a monolithic framework by encapsulating all security functions into one container that is

deployed on a RPi device. A RPi device works as a client to send service transactions to Micro_App and Mono_App service providers that are deployed on a desktop.

Figure 8 shows network latency of launching multiple identity verification requests on microservices and monolithic frameworks. On receiving transactions from client, Micro_App service provider can even distribute service workload into subgroups that are assigned to microservices nodes in the network. Therefore, total network delay is reduced to improve the quality of service (QoS) in terms of response time. As the bottom line in Figure 8 shows, the Micro_App with full microservices capacity achieves lower network delay than other scenarios, and it is amount to 23% of that Mono_App does when TPS is 100. Compared to Mono_App, certain fraction of microservices node dropout does not disturb the service access. However, the network delay increases significantly as fraction of dropout increases, as Figure 8 shows.

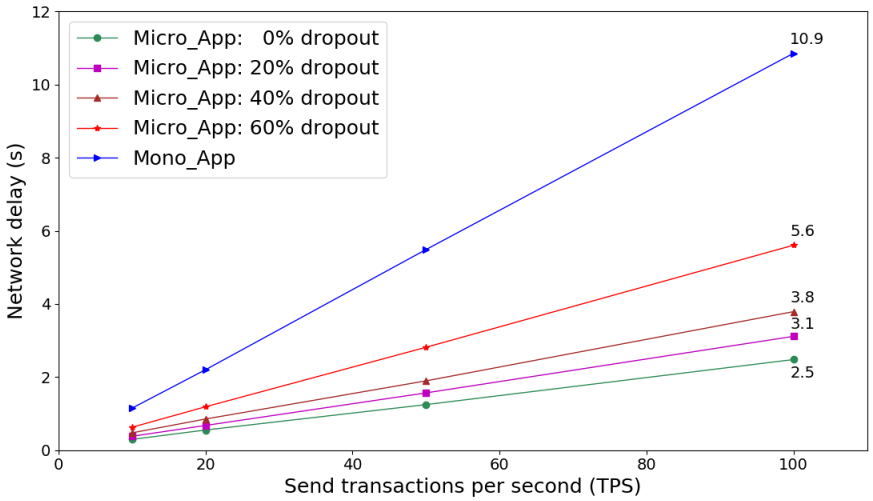

Figure 8. Network latency of service requests on microservices and monolithic frameworks.

Figure 9 presents the transaction throughput that Micro_App and Mono_App can achieve as TPS increases. The transaction throughput is greatly influenced by network communication capacity and service processing capability that a security microservices host machine can provide. As Mono_App uses a single monolithic application node to handle all security service transactions, transaction throughput is easier to become saturate than Micro_App as TPS increases. Figure 9 shows that transaction throughput ascent of Micro_App with 0% dropout becomes flat when TPS is about 60, however, Mono_App cannot dramatically increase transaction throughput as TPS is larger than 20.

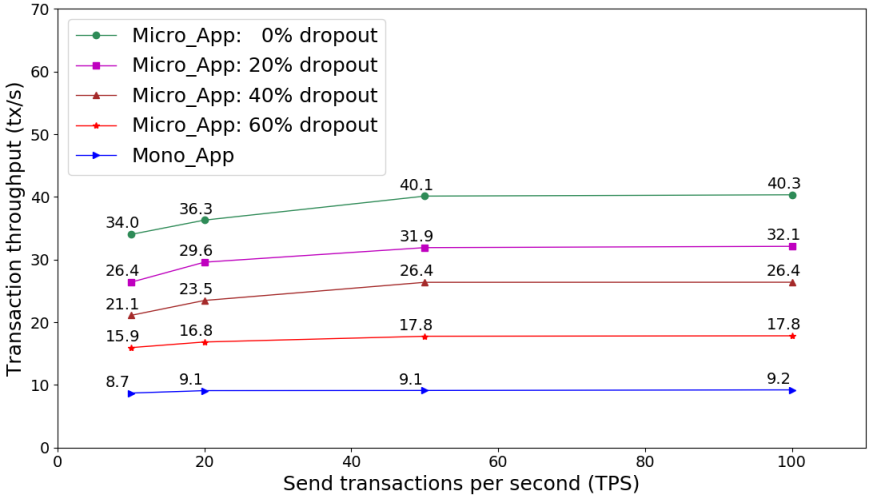

Figure 9. Transaction throughput of service requests on microservices and monolithic frameworks.

Figure 9 also indicates that transaction throughput of Micro_App is greatly impacted by microservices dropout rate. As each microservices node has limited service processing capacity, so that service access overload to dropout nodes is transferred to other working nodes. As a result, the transaction throughput of Micro_App declines owning to the decreased system capacity by dropout nodes. Micro_App can tolerant certain fraction of microservices nodes dropout, and it is more robust than Mono_App, which is vulnerable to performance bottleneck. Moreover, through properly deploying security microservices nodes, Micro_App is more scalable and flexible than Mono_App on dynamic and heterogeneous IoT-based networks.

6.2.3. Blockchain Fabric Performance: Tendermint vs. Ethereum

The security microservices utilizes a set of *transaction_commit*() RPC functions to save data to the distributed ledger, and consensus protocol is responsible to guarantee the security of recorded data on the distributed ledger. Thus, executing consensus protocol and recording data into distributed ledger inevitably introduce extra delays besides normal service operation. One-hundred testing runs have been carried out based on the proposed test scenario, in which a video stream fingerprint microservices node saves ENF fingerprint into Tendermint and a AC microservices node updates access token SC on Ethereum.

Figure 10 shows the time delay, that a node launches a blockchain transaction (*bc_tx*) and waits until it has been committed on the distributed ledger. The *bc_tx* committed time is closely related to the block confirmation time that is defined by the consensus algorithm. Tendermint utilizes a BFT consensus protocol to achieve a deterministic finality on a new block for each voting round, so that *bc_tx* committed time is almost stable (~2.9 s), as showb by Figure 10. However, Ethereum relies on a random block generation mechanism defined by the PoW consensus protocol, and it achieves a probabilistic finality on committed data on the distributed ledger. Therefore, the *bc_tx* committed time of Ethereum is greatly varying owing to variable block confirmation time as illustrated by Figure 10.

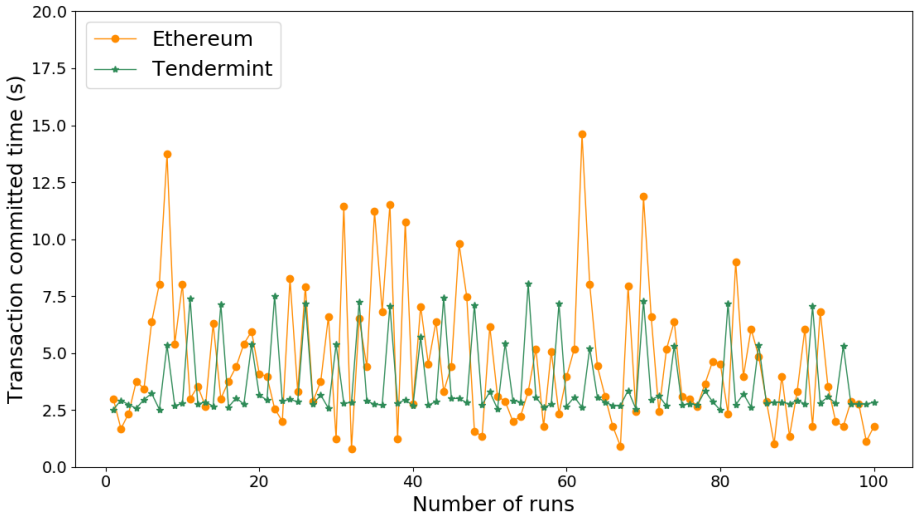

Figure 10. Network latency for committing data transactions in blockchain.

Table 2 provides a comprehensive performance of running intra-domain (Tendermint) and inter-domain consensus (Ethereum) protocols regarding several key performance matrices. The *bc_tx* committed time is calculated by averaging 100 test results in Figure 10. Ethereum achieves a 4.6 s latency by committing a transaction on distributed ledger, which is 28% longer than Tendermint does. The SC-based security services are generally used by either non-time-sensitive operations, like verify access token, or offline tasks, like checking integrity of contextual features. Thus, 4.6 s latency for updating a SC data meets service requirements in SPS applications. The ENF-based false frame detection relies on a minimal 5 s sliding window to obtain a constant correlation coefficient for

dissimilar ENF signal estimations. Thus, 3.6 s *bc_tx* committed time is enough for finalizing an ENF fingerprint data on the distributed ledger within one detection cycle.

Table 2. Comparative evaluation on blockchain fabric.

	Ethereum		**Tendermint**
	Miner	**Node**	**Validator**
bc_tx committed time (s)	4.6		3.6
CPU usage (%)	103	5	27.5
Memory usage (MB)	1232	45	64
Gas/bc_tx (Ether)	0.001		×

Considering resource consumption in term of CPU and memory usage, Tendermint demonstrates advantages over Ethereum. Ethereum uses a computation intensive PoW consensus algorithm, and mining process almost occupies the full CPU capacity and consumes about 1.2 GB memory. Therefore, it is not feasible to deploy miners on resource constrained IoT devices. However, Ethereum can be deployed as a light node on RPi devices, which only synchronizes and validates blockchain data instead of mining new blocks. Table 2 shows that a Ethereum node only needs 5% CPU capacity and 45 MB memory to support data recording and querying on SC. Tendermint uses a lightweight BFT consensus algorithm to achieve efficiency in CPU and memory usage, so that it is suitable for deploying validators on resource-constraint IoT devices.

In an Ethereum network, transaction commitment requires gas that is used to pay for miners. The average gas fee for each transaction is 0.001 Ether, which amounts to $2.3 given the Ether price in the public Ethereum market ($236.23/Ether at 20 July 2020). Compared to Ethereum, Tendermint does not require transaction fee. Therefore, it is more suitable for inter-domain scenario, which requires large volume of data transactions without introducing additional financial cost.

6.3. Discussions

Our experimental results verified the feasibility of the proposed BlendSPS solution. It has the potential to enable a practical IoT-based SPS system featured as a decentralized, secure, and privacy preserving service. Compared to centralized security solutions implemented by monolithic framework, the BlendSPS has the following advantages.

(1) Decentralized network architecture: The BlendSPS system leverages the blockchain and smart contract technology to provide decentralized security services. Therefore, geographical scattered data owners and service providers maintain control on their own resources and securely share information without relying on a centralized third authority to ensure a trust relationship. It is promising to improve system performance and reduce the risk of single point of failure.

(2) Flexible and fine-grained SOA framework: BlendSPS uses fine-grained and loose-coupling microservices to enable flexible and robust service architecture. As the whole system can be decoupled into multiple fine-grained microservices units, each microservices unit is only responsible for a dedicated task according to domain related performance and security requirements. Moreover, running microservices units is independent of remaining parts of system. Therefore, user and service providers can increase or decrease serving microservices nodes to achieve expected QoS without disturbing system functionality.

(3) Security proprieties: Given a partial synchronous network environment of SPS settings, persistence and termination are two security proprieties provided by the hybrid blockchain fabric to enable a robust distributed ledger. The persistence ensures that those finalized hashed model strings are immutable and traceable, and can be audited by other participants. Termination ensures aliveness

so that all valid hashed model transactions by honest nodes are finalized in distributed ledger after a sufficient amount of time.

7. Conclusions

This paper introduces BlendSPS, a blockchain-enabled decentralized smart public safety system, to enhance security and privacy-preserving proprieties in distributed SPS network. Leveraging lightweight microservices-based SOA framework and hybrid blockchain fabric, BlendSPS supports a decentralized, secure and efficient data sharing and multidomain operations in SPS settings. Moreover, BlendSPS brings low computation and communication cost on edge network, making it ideal for IoT-based SPS applications

While the experimental results on the prototype are encouraging, there are still open questions to answer before deploying a practical solution on real-world SPS scenarios. By integrating blockchain with a heterogeneous IoT-based SPS network, a hybrid blockchain fabric is promising to address trade-offs caused by consensus mechanisms, like scalability, efficiency, and security. However, it also brings new performance and security concerns during cross-chain transactions. In addition, the transparency of the distributed ledger may exacerbate the privacy problem when users record sensitive data on blockchain. Furthermore, each node needs a full replica of the public ledger to join the blockchain network, hence, it inevitably increases bootstrapping time for new participant and incurs huge storage space consumption on host machine. Part of our ongoing effort is focused on further exploration of the efficient and privacy-preserving blockchain protocol, which can be executed at the edge networks with lower communication, computation, and storage overhead.

Author Contributions: Conceptualization, R.X. and Y.C.; Methodology, R.X., S.Y.N., D.N., and A.F.; Software, R.X., S.Y.N., D.N., and A.F.; Validation, R.X., D.N., and Y.C.; Formal analysis, R.X. and Y.C.; Investigation, R.X.; Resources, R.X., S.Y.N., D.N., and A.F.; Data Curation, R.X.; Writing—Original Draft Preparation, R.X., S.Y.N., D.N., and A.F.; Writing—Review and Editing, R.X. and Y.C.; Visualization, R.X.; Supervision, Y.C.; Project Administration, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript.

ABI	Application Binary Interfaces
AC	Access Control
BFT	Byzantine Fault Tolerant
CNN	Convolutional Neural Networks
DLT	Distributed Ledger Technology
DNN	Deep Neural Networks
ENF	Electrical Network Frequency
IoT	Internet of Things
IPC	Inter Process Communication
ML	Machine Learning
PKI	Public Key Infrastructure
PoW	Proof-of-Work
QoS	Quality of Service
RPC	Remote Procedure Call
SC	Smart Contract
SOA	Service-oriented Architecture
SPS	Smart Public Safety
VID	Virtual Identity

References

1. Nikouei, S.Y.; Xu, R.; Chen, Y.; Aved, A.; Blasch, E. Decentralized smart surveillance through microservices platform. In *Sensors and Systems for Space Applications XII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2019; Volume 11017, p. 110170K.
2. Xu, R.; Nikouei, S.Y.; Chen, Y.; Blasch, E.; Aved, A. Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 564–571.
3. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.Y.; Faughnan, T. Smart surveillance as an edge network service: From harr-cascade, svm to a lightweight cnn. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 256–265.
4. Wu, R.; Liu, B.; Chen, Y.; Blasch, E.; Ling, H.; Chen, G. A container-based elastic cloud architecture for pseudo real-time exploitation of wide area motion imagery (wami) stream. *J. Signal Process. Syst.* **2017**, *88*, 219–231. [CrossRef]
5. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; Technical Report; Manubot, 2019. Available online: https://git.dhimmel.com/bitcoin-whitepaper/ (accessed on 23 July 2020).
6. Xu, T.; Botelho, L.M.; Lin, F.X. Vstore: A data store for analytics on large videos. In Proceedings of the Fourteenth EuroSys Conference 2019, Dresden, Germany, 25–28 March 2019; ACM: New York, NY, USA, 2019; p. 16.
7. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.Y.; Faughnan, T.R. Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN. In Proceedings of the IEEE International Conference on Edge Computing, San Francisco, CA, USA, 2–7 July 2018.
8. Blasch, E.P.; Liu, K.; Liu, B.; Shen, D.; Chen, G. Cloud Based Video Detection and Tracking System. US Patent 9,373,174, 21 June 2016.
9. Ma, J.; Dai, Y.; Hirota, K. A Survey of Video-Based Crowd Anomaly Detection in Dense Scenes. *J. Adv. Comput. Intell. Intell. Inform.* **2017**, *21*, 235–246. [CrossRef]
10. Ribeiro, M.; Lazzaretti, A.E.; Lopes, H.S. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognit. Lett.* **2017**, *105*, 13–22. [CrossRef]
11. Fuse, T.; Kamiya, K. Statistical Anomaly Detection in Human Dynamics Monitoring Using a Hierarchical Dirichlet Process Hidden Markov Model. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3083–3092. [CrossRef]
12. Chen, N.; Chen, Y.; You, Y.; Ling, H.; Liang, P.; Zimmermann, R. Dynamic urban surveillance video stream processing using fog computing. In Proceedings of the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), Taipei, Taiwan, 20–22 April 2016; pp. 105–112.
13. Wang, X. Intelligent multi-camera video surveillance: A review. *Pattern Recognit. Lett.* **2013**, *34*, 3–19. [CrossRef]
14. Blasch, E.; Nagy, J.; Aved, A.; Jones, E.K.; Pottenger, W.M.; Basharat, A.; Hoogs, A.; Schneider, M.; Hammoud, R.; Chen, G.; et al. Context aided video-to-text information fusion. In Proceedings of the 17th International Conference on Information Fusion (FUSION), Salamanca, Spain, 7–10 July 2014; pp. 1–8.
15. Aved, A.J.; Blasch, E.P. Multi-int query language for dddas designs. *Procedia Comput. Sci.* **2015**, *51*, 2518–2532. [CrossRef]
16. Piciarelli, C.; Esterle, L.; Khan, A.; Rinner, B.; Foresti, G.L. Dynamic Reconfiguration in Camera Networks: A short survey. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 965–977. [CrossRef]
17. Fan, C.T.; Wang, Y.K.; Huang, C.R. Heterogeneous information fusion and visualization for a large-scale intelligent video surveillance system. *IEEE Trans. Syst. Man. Cybern. Syst.* **2017**, *47*, 593–604. [CrossRef]
18. Wu, J. Mobility-enhanced public safety surveillance system using 3d cameras and high speed broadband networks. *GENI NICE Evening Demos* 2015. Available online: https://cis.temple.edu/~jiewu/research/research_projects/Safety_Surveillance.html (accessed on 23 July 2020).
19. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 103–130.
20. Nikouei, S.Y.; Chen, Y.; Aved, A.; Blasch, E.; Faughnan, T.R. I-safe: Instant suspicious activity identification at the edge using fuzzy decision making. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, Arlington, VA, USA, 7–9 November 2019; pp. 101–112.

21. Nikouei, S.Y.; Chen, Y.; Song, S.; Faughnan, T.R. Kerman: A hybrid lightweight tracking algorithm to enable smart surveillance as an edge service. In Proceedings of the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 11–14 January 2019; pp. 1–6.
22. Datta, S.K.; Bonnet, C. Next-generation, data centric and end-to-end iot architecture based on microservices. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Jeju, Korea, 24–26 June 2018; pp. 206–212.
23. Nagothu, D.; Xu, R.; Nikouei, S.Y.; Chen, Y. A microservice-enabled architecture for smart surveillance using blockchain technology. In Proceedings of the 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, 16–19 September 2018; pp. 1–4.
24. Xu, R.; Ramachandran, G.S.; Chen, Y.; Krishnamachari, B. Blendsm-ddm: Blockchain-enabled secure microservices for decentralized data marketplaces. In Proceedings of the 2019 IEEE International Smart Cities Conference (ISC2), Casablanca, Morocco, 14–17 October 2019; pp. 14–17.
25. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Newton, MA, USA, 2015.
26. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*. [CrossRef]
27. Nikouei, S.Y.; Xu, R.; Nagothu, D.; Chen, Y.; Aved, A.; Blasch, E. Real-time index authentication for event-oriented surveillance video query using blockchain. In Proceedings of the 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, 16–19 September 2018; pp. 1–8.
28. Xu, R.; Lin, X.; Dong, Q.; Chen, Y. Constructing Trustworthy and Safe Communities on a Blockchain-Enabled Social Credits System. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, New York, NY, USA, 5–7 November 2018; ACM: New York, NY, USA, 2018; pp. 449–453.
29. Lin, X.; Xu, R.; Chen, Y.; Lum, J.K. A blockchain-enabled decentralized time banking for a new social value system. In Proceedings of the 2019 IEEE Conference on Communications and Network Security (CNS), Washington, DC, USA, 10–12 June 2019; pp. 1–5.
30. Xu, R.; Chen, Y.; Blasch, E.; Chen, G. Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness. *Opt. Eng.* **2019**, *58*, 41609. [CrossRef]
31. Xu, R.; Chen, Y.; Blasch, E.; Aved, A.; Chen, G.; Shen, D. Hybrid blockchain-enabled secure microservices fabric for decentralized multi-domain avionics systems. In *Sensors and Systems for Space Applications XIII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2020; Volume 11422, p. 114220J.
32. Xu, R.; Chen, S.; Yang, L.; Chen, Y.; Chen, G. Decentralized autonomous imaging data processing using blockchain. In *Multimodal Biomedical Imaging XIV*; International Society for Optics and Photonics: Bellingham, WA, USA, 2019; Volume 10871, p. 108710U.
33. Xu, R.; Chen, Y.; Blasch, E.; Chen, G. BlendCAC: A BLockchain-ENabled Decentralized Capability-based Access Control for IoTs. In Proceedings of the 2018 IEEE International Conference on Blockchain (Blockchain-2018), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1–8.
34. Xu., R.; Chen, Y.; Blasch, E.; Chen, G. Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the iot. *Computers* **2018**, *7*, 39. [CrossRef]
35. Nagothu, D.; Schwell, J.; Chen, Y.; Blasch, E.; Zhu, S. A study on smart online frame forging attacks against video surveillance system. In *Sensors and Systems for Space Applications XII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2019; Volume 11017, p. 110170L.
36. Nagothu, D.; Chen, Y.; Blasch, E.; Aved, A.; Zhu, S. Detecting Malicious False Frame Injection Attacks on Surveillance Systems at the Edge Using Electrical Network Frequency Signals. *Sensors* **2019**, *19*, 2424. [CrossRef] [PubMed]
37. Fitwi, A.; Chen, Y.; Zhu, S. No peeking through my windows: Conserving privacy in personal drones. In Proceedings of the 2019 IEEE International Smart Cities Conference (ISC2), Casablanca, Morocco, 14–17 October 2019; pp. 199–204.
38. Fitwi, A.; Chen, Y.; Zhu, S. A lightweight blockchain-based privacy protection for smart surveillance at the edge. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 552–555.
39. Fitwi, A.; Yuan, M.; Nikouei, S.Y.; Chen, Y. Minor privacy protection by real-time children identification and face scrambling at the edge. *EAI Endorsed Trans. Secur. Saf.* **2020**, *18*. [CrossRef]
40. Hajj-Ahmad, A.; Garg, R.; Wu, M. Spectrum combining for ENF signal estimation. *IEEE Signal Process. Lett.* **2013**, *20*, 885–888. [CrossRef]

41. The Images of Groups Dataset. Available online: http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html (accessed on 23 July 2020).
42. Lamport, L.; Shostak, R.; Pease, M. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]
43. Flask: A Pyhon Microframework. Available online: http://flask.pocoo.org/ (accessed on 23 July 2020).
44. Ethereum Homestead Documentation. Available online: http://www.ethdocs.org/en/latest/index.html (accessed on 23 July 2020).
45. Kwon, J. Tendermint: Consensus without mining. **2014**, *1*, 11. Available online: https://tendermint.com/docs/tendermint.pdf (accessed on 23 July 2020).
46. Solidity. Available online: http://solidity.readthedocs.io/en/latest/ (accessed on 23 July 2020).
47. Pyca/Cryptography Documentation. Available online: http://pyca/cryptography (accessed on 23 July 2020).
48. BlendSPS Project. Available online: https://github.com/samuelxu999/Research/tree/master/Security/py_dev/BlendSPS/ (accessed on 23 July 2020).
49. Go-ethereum. Available online: https://ethereum.github.io/go-ethereum/ (accessed on 23 July 2020).