

Article Secure Video Surveillance Framework in Smart City

Hao Li ¹, Tianhao Xiezhang ^{2,3}, Cheng Yang ^{2,3,*}, Lianbing Deng ¹ and Peng Yi ⁴

- ¹ Post-Doctoral Research Center of Zhuhai Da Hengqin Science and Technology Development Co. Ltd., Zhuhai 519000, China; cuclihao@cuc.edu.cn (H.L.); denglb@dhqtech.com (L.D.)
- ² State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China; xiezth@cuc.edu.cn
- ³ College of Information Engineering, Communication University of Chins, Beijing 100024, China
- ⁴ Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia;
 - peng.yi@unsw.edu.au Correspondence: cafeeyang@163.com

Abstract: In the construction process of smart cities, more and more video surveillance systems have been deployed for traffic, office buildings, shopping malls, and families. Thus, the security of video surveillance systems has attracted more attention. At present, many researchers focus on how to select the region of interest (RoI) accurately and then realize privacy protection in videos by selective encryption. However, relatively few researchers focus on building a security framework by analyzing the security of a video surveillance system from the system and data life cycle. By analyzing the surveillance video protection and the attack surface of a video surveillance system in a smart city, we constructed a secure surveillance framework in this manuscript. In the secure framework, a secure video surveillance model is proposed, and a secure authentication protocol that can resist man-in-the-middle attacks (MITM) and replay attacks is implemented. For the management of the video encryption key, we introduced the Chinese remainder theorem (CRT) on the basis of group key management to provide an efficient and secure key update. In addition, we built a decryption suite based on transparent encryption to ensure the security of the decryption environment. The security analysis proved that our system can guarantee the forward and backward security of the key update. In the experiment environment, the average decryption speed of our system can reach 91.47 Mb/s, which can meet the real-time requirement of practical applications.

Keywords: secure video surveillance; key management; decryption security; real-time performance

1. Introduction

With the continuous development of digital video cameras and network infrastructure, video surveillance is widely deployed in smart cities [1–3]. Moreover, the storage service for the public from cloud computing causes the cost of constructing video monitoring systems with different scales to become lower and lower. However, the accompanying security problems have become increasingly prominent [4,5].

Security and privacy issues exist widely in video surveillance systems deployed in smart cities. With the privacy leakage of Apple iCloud and the Verkada camera, the privacy of multimedia has attracted more and more attention. PatronuS [6] was proposed to find the region of interest (RoI) in videos quickly, and then the RoI was encrypted to achieve privacy protection in videos. Similarly, motion detection [7,8], skin detection [9], and compressed sensing [10] methods were proposed to find the proper RoIs for privacy protection. A lightweight encryption algorithm that was devised based on layered cellular automata (LCA) was proposed for fast RoI encryption [11].

However, privacy is a complex concept to define. It is even more difficult to define privacy areas in video, an unstructured file rich in various information [12]. It is not reasonable to simply regard the information of motion or skin as the RoI in practical application. For example, in traffic monitoring, privacy may be the license plate; privacy in



Citation: Li, H.; Xiezhang, T.; Yang, C.; Deng, L.; Yi, P. Secure Video Surveillance Framework in Smart City. *Sensors* **2021**, *21*, 4419. https:// doi.org/10.3390/s21134419

Academic Editors: Jorge Bernal Bernabe, Antonio Skarmeta, Fabio Massacci and Davy Preuveneers

Received: 29 April 2021 Accepted: 16 June 2021 Published: 28 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). hospitals may be electronic medical records (EMRs); and the privacy of station monitoring may be a passenger's ID card, etc. It is a better choice to disturb the whole image with an acceptable encryption calculation. At the same time, the risk of privacy leakage may appear in the process of video collection, transmission, storage, and analysis [13]. Therefore, to fully protect privacy in multimedia, we need to realize data security during its whole life cycle in a media system.

The main challenge of data security faced by video surveillance systems is how to ensure the video integrity, confidentiality, and availability during the data life cycle. Sufficient encryption for multimedia is necessary. However, how to implement terminal authentication and key management securely is equally essential. In addition, many monitors exist in different locations of video surveillance systems, which need to obtain the suitable key and decrypt sensitive videos in specified areas. Hence, ensuring the monitor obtains the appropriate permission and builds a secure decryption environment is also meaningful. Monitors with excessive authorization may infringe upon other people's privacy. Insufficient authorization of monitors may cause the video surveillance system to fail to work properly. For many small-scale scenes, such as homes and stores, a monitor is often replaced by a personal computer, which lacks the necessary security protection. It is obvious to ensure the security of the decryption environment. As a whole, in multi-level, multi-regional video surveillance systems for smart cities, the challenges mentioned above are unprecedented.

In our manuscript, we build a secure, convenient, and flexible video surveillance framework for use in a smart city. First, a trusted authority (TA) is built as a trust center of the city. Then, any terminal, whether a camera or a monitor, must be registered with the TA at the factory. When the terminal wants to access one video surveillance system, it should be authenticated to obtain the corresponding key legally. Specifically, the contributions of this manuscript include the following:

- (1) A secure model is proposed to realize unified protection of video surveillance systems in smart cities. The model provides terminal authentication, key management, and secure video decryption for video surveillance systems with different scales, such as shops, families, and transportation.
- (2) A secure authentication protocol is implemented for a secure video surveillance framework. Based on the Diffie–Hellman protocol, we add secret and timestamp elements that can effectively resist man-in-the-middle attacks (MITM) and replay attacks.
- (3) Combined with the actual needs of smart city video surveillance, we build a hierarchical key management architecture named the Key Management Scheme Based on Normal Tree (KMSNT), which is based on the group key distribution and the Chinese remainder theorem (CRT). The TA manages the key of all video surveillance systems in the city. Group keys maintained by the TA are sent to video surveillance systems with different scales, such as families, shops, and buildings. When a group member changes, KMSNT updates the group key efficiently to ensure the data security.
- (4) Considering that the decryption environment in a small-scale video surveillance system monitor may not be safe, we construct a video security decryption suite based on transparent encryption. It provides a secure environment for identity authentication, key acquisition, and data decryption of monitors. At the same time, it can prevent the terminal from illegally copying and modifying the video.

The remaining sections of this paper are organized as follows. The second part introduces related work. We prepare some relevant knowledge for our proposed scheme in Section 3. Section 4 proposes a secure video surveillance system model and describes the detailed algorithms for terminal authentication, key management, and secure video decryption. We analyze the security and real-time performance of the system in Section 5. Finally, Section 6 is the conclusion of our paper.

2. Related Work

We need to consider the security of video surveillance systems from two aspects. First, an adversary may access a video by illegal methods at a certain stage (video production, transmission, storage, etc.), leading to a privacy leakage. Second, an adversary may hijack the terminal to interfere with the normal work of the system. The related research can also be divided into two aspects: data protection and system security.

From the perspective of video data protection, digital video surveillance systems are mainly faced with three threats: confidentiality, integrity, and availability of the video. A secure video surveillance framework was proposed in [14] which used the AES symmetric encryption algorithm to encrypt the video stream, realized a symmetric key distribution through an RSA-based protocol, and employed hash-based message authentication message digest hashing (HMAC-MD5) to ensure the integrity of the video stream. Furthermore, hierarchical encryption and layered access control schemes were proposed in [15,16], in which an ordinary user can access the public video, while only authorized users are able to access the recovered private content. Generally speaking, encryption is still an important method to achieve privacy protection in video surveillance systems [17]. Naturally, it is increasingly urgent to build a secure and efficient key management scheme. Obviously, one-to-one key distribution protocols based on RSA or other asymmetric encryption algorithms are unsuitable for large-scale city monitoring systems because of the bottleneck in key management and distribution efficiency.

From the perspective of the attack surface of video surveillance systems, unauthorized video monitoring and stealing archived video footage are two common attack vectors used to attack video surveillance systems [18]. First of all, to ensure the safe running of a video surveillance system, any device connected to the system needs to be certified [19]. The dual-channel authentication approach using CCTV machine learning with radio frequency identification (RFID) was proposed [20]. The security of a video surveillance system depends on the secure storage of RFID and the accuracy of machine learning. However, an adversary attack proved that the machine learning algorithm can be misled [21]. Based on physically unclonable functions, authentication and key exchange protocols were proposed for Internet of Things (IoT) [22]. According to both [20] and [22], critical hardware is the foundation of security. Once the hardware information is leaked, the security of the system is no longer guaranteed. A role-based access control video surveillance mechanism was proposed in [23], which used a blockchain to guarantee the integrity of shared data and prevent indiscriminate access. However, a blockchain has the defects of heavy computing and a low efficiency. In practical applications, video surveillance systems with different scales need a set of secure, convenient, and flexible authentication and key distribution protocols to cooperate with video encryption algorithms of varying granularities and ultimately achieve the purpose of protecting the video data security. In addition, for smallscale scenes, such as homes and stores, the monitor is often a personal computer that lacks the necessary security protection. Therefore, how to ensure the security of the decryption environment in a monitor is also a very important problem.

3. Preliminaries

3.1. Group Key Distribution

When a large number of terminals need to achieve encryption or decryption, the one-to-one key distribution protocol exposes the bottleneck of a low update efficiency and poor real-time performance. Group key management aims to achieve an efficient key update [24]. A group key update scheme-based logical key hierarchy (LKH) is shown in Figure 1. In the LKH structure, users or terminals are represented as leaf nodes.

User join. When user 3 joins the group, the group key *GK* and the internal keys (*KM*, *KI*, *KB*) belonging to the path from the root to the user 3 node are updated in the following steps to prevent the new user from obtaining the old keys.



Figure 1. Group key update scheme-based LKH.

- (1) The new keys *GK'*, *KM'*, *KI'*, *KB'* are encrypted by K3 and K4 and unicast to user 3 and user 4;
- (2) *GK'*, *KM'*, *KI'* are encrypted by *KA* and multicast to users 1 and 2;
- (3) *GK'*, *KM'* are encrypted by *KJ* and multicast to users 5 to 8;
- (4) *GK'* is encrypted by *KN* and multicast to users 9 to 16.

User leave. When user 11 leaves the group, *GK*, *KN*, *KK*, *KF* need to be updated to ensure the forward security. The update steps are described below:

- (1) The new keys *GK*′, *KN*′, *KK*′, *KF*′ are encrypted by *K*12 and multicast to user 12;
- (2) *GK'*, *KN'*, *KK'* are encrypted by *KE* and multicast to users 9 and 10;
- (3) GK', KN' are encrypted by KL and multicast to users 13 to 16;
- (4) *GK'* is encrypted by *KM* and multicast to users 1 to 8.

The group key update scheme dramatically reduces the time consumption of the key update process. However, there is still room for optimization in the video surveillance system.

First, with the increase in users, the number of keys that users in the leaf node need to maintain increases heavily. When the number of users is N, the number of keys stored in the leaf node is $\log_k N + 1$. k is the branch number of the tree. For example, when the user number is 16, the leaf node needs to maintain five keys.

Second, there are still several events of encryption and multicasting during the key update process. For example, when one user is evicted from the group, the new keys need to be distributed (k - 1)d times. Here, *k* is the number of branches, and *d* is the depth of the tree.

3.2. Chinese Remainder Theorem (CRT)

Assuming that $m_1, m_2, ..., m_n$ are *n* pairwise relative prime numbers, then for any integer $a_1, a_2, ..., a_n$, the group of equations (Equation (1)) always has a unique integer solution modulo $N = m_1 \times m_2 \times ... \times m_n$.

$$\begin{cases}
x \equiv a_1 \pmod{m_1} \\
x \equiv a_2 \pmod{m_2} \\
\dots \\
x \equiv a_n \pmod{m_n}
\end{cases}$$
(1)

$$x \equiv \sum_{i=1}^{n} a_i \times \frac{N}{m_i} \times \left[\left(\frac{N}{m_i} \right)^{-1} \right]_{m_i} \pmod{N}$$
(2)

In Equation (2), there is $\frac{N}{m_i} \times \left[\left(\frac{N}{m_i} \right)^{-1} \right]_{m_i} \equiv 1 \pmod{m_i}$. $\left[\left(\frac{N}{m_i} \right)^{-1} \right]_{m_i}$ is the inverse of

$\frac{N}{m_i}$ modulo m_i .

Combined with the actual situation of video surveillance systems in smart cities, the CRT is used to improve the efficiency of the group key distribution.

3.3. Transparent Encryption in Video

Transparent encryption aims to provide a secure encryption and decryption environment, which protects the decryption in personal computers used as a monitor. Three transparent encryption algorithms for H.264/AVC and H.264/SVC compressed videos have been proposed [25]. All three schemes use encryption-after-encoding schemes. Taking advantage of the inter-layer prediction technique used in H.264/SVC (scalable video coding), a block-based encryption scheme (BBES) for encrypting H.264/SVC enhancement layers has been proposed [26]. Transparent encryption is used to encrypt a small amount of content adaptive binary arithmetic coding (CABAC) parameters to reduce the video quality and dampen any appetite for pirated copies [27]. In these studies, decryption is only briefly introduced because it has the same process as encryption. However, in the practical scenario, decryption has to face more security risks than encryption. One of the risks is that the decryption environment may not be secure. The solution proposed in [28] protects the decryption environment by a secure display path provided by ARM processors. Its security comes from a trusted root based on hardware, such as TrustZone [29]. For most personal computers, there is no trusted root module based on hardware. One scheme that revises the method of conditional access from one that depends on hardware CAMs within STBs to a software-based method of distribution was provided in [30]. The scheme is not based on hardware security. However, the decryption module is coupled in the application software, which is still not safe.

Therefore, we proposed a multimedia decryption middleware scheme based on transparent encryption. In the scheme, decryption is conducted by the transparent encryption driver provided by the operating system. The scheme can ensure that the plaintext only exists in memory and can only be used by authorized applications and prevent illegal disk writing and network transmission. However, only encryption is not enough for the authorized multimedia distribution. Authority control is also important. In our paper, key management and file decryption are integrated together as one middleware and provide a security service to the upper multimedia applications. This not only eliminates the coupling between the complex decryption process and the complex decoding process, reducing the difficulty and security requirements of the upper software development, but also improves the decryption efficiency and reduces the delay. The middleware completes a series of complex operations such as authentication, key acquisition, and decryption without the user's awareness, which enhances the user experience.

4. Our Work

In this section, we propose a secure video surveillance system model including terminal registration, authentication, key management, and video encryption. Considering private protection in real life, a video surveillance based on cloud computing faces two security threats. (1) Traditional video streams are mainly transmitted and stored in a plaintext model, which leads to the risk of data leakage [31,32]. (2) Cloud-based video surveillance saves a lot of storage resources and provides a convenient and low-cost monitoring capability for small-scale scene applications such shops and families. However,

6 of 16

shops and families often lack security awareness and protection skills, which makes web cameras and monitors (most of the time, the monitor is just a normal personal computer) vulnerable [33,34].

4.1. System Model

In order to cover the above security risks, we built a typical smart city security video surveillance model (Figure 2). It is worth noting that we have not built this secure framework for a single video surveillance system, but for all video surveillance systems with different scales in smart cities. Taking a city as a unit, a unified secure video surveillance model was constructed. The model mainly consists of the following four subjects: trusted authority (TA), media cloud, monitor, and camera.



Figure 2. Secure video surveillance model in smart city.

The **trusted authority (TA)** is responsible for the registration, authentication, and key management (key generation, distribution, storage, and destruction) of terminal devices, including monitors and cameras. The TA is trustworthy, and its structure can be central or hierarchical.

The **media cloud** is used to store and process monitoring videos. We believe the media cloud is curious, that is, the media cloud honestly follows the protocol execution, but curiosity propels it to speculate over and analyze the data and searchable index tree available at the server.

The **camera** is the basic video capture unit in the system. Multiple cameras can form a group to monitor a specific area, such as traffic, a shop, a family, and a building. In our system, any camera needs to be registered with the TA in its factory settings. Before the camera joins the video surveillance system, it should pass the security authentication. The legal camera captures and encrypts the video and then sends the encrypted video to the media cloud and local temporary storage server.

The **monitor** can view the videos captured by the cameras under its jurisdiction. In the actual scene, monitors and cameras have an obvious hierarchical structure. Simply, a monitor in a small shop can only view the video of the cameras belonging to the shop. A video surveillance system for an office building that has many independent companies has a more obvious hierarchy. One company's monitor can only view the videos covering its office area but cannot view the videos beyond that area. However, the property company of the building has the right to view the surveillance video of the whole public area.

Figure 2 shows the secure video surveillance model used in a smart city. In the model, any terminal (monitor and camera) needs to register with the TA before it leaves the factory.

The terminal sends the authentication request to the TA using its registration information when it wants to join the video surveillance system. After the authentication is finished, the TA classifies the terminal into the corresponding group for unified key management. For a legal camera, when the group key is received, the collected video is encrypted and sent to the media cloud and local temporary storage server. For a legal monitor, it has the right to access the video produced by the camera on its leaf nodes. Considering that a monitor may be a personal computer without any professional security protection in small-scale scenarios, we adopted transparent encryption to ensure the plaintext and decryption key do not exist in the computer memory. In particular, the model can provide a specific interface for the public police to resist crime.

Our model's hierarchical key management architecture has three advantages for a secure video surveillance model in smart cities. (1) Cameras often work as a cluster, and hierarchical key management is more suitable for group key management for each cluster. (2) The hierarchical key management architecture has better visualization to represent the monitors and cameras in different regions and geographical locations. (3) The architecture can be easily extended, and the TA can be transformed from the central model to the distributed model, which is necessary for large-scale video surveillance in smart cities. Taking the central model architecture as an example (Figure 3), we define the TA to be on the root node, cameras to be on the leaf nodes, and monitors to be located on internal nodes.



Figure 3. Hierarchical key management architecture.

4.2. Registration

Monitors and cameras need to register with the TA before they leave the factory. The registration process is initiated by the terminal *i* through the secure channel established between the TA and the factory. During this process, the TA records terminal information, including the manufacturer, registration time, device name, and type. Then, the TA assigns a unique public ID named ID_i and a secret parameter k_i , where ID_i is used as the identity of the terminal, and k_i is used for future authentication.

4.3. Authentication

Before the device *i* joins the system, the TA and the device should be mutually authenticated. Firstly, the device *i* sends the authentication request message (containing ID_i , the parent node of *i* denoted as ID_{parent_i} , a random unique prime number *p*, and a random positive integer *g*) to the TA. Then, the TA starts the authentication phase and sends a message *auth_i* to the terminal. The message contains the device ID ID_i , the unique identity of the TA ID_{TA} , the message authentication code (*MAC_i*), and the timestamp *T*.

The message $auth_i$ is shown in Equation (3). The message authentication code MAC_i is obtained by hashing the ID_i , terminal secret key k_i , $g^a \pmod{p}$, and T. In MAC_i , a is a positive integer randomly chosen by the TA:

$$auth_i = ID_i \parallel ID_{TA} \parallel MAC_i \parallel g^a (mod \ p) \parallel T$$
(3)

$$MAC_{i} = hash(ID_{i} \parallel ID_{parent_{i}} \parallel k_{i} \parallel g^{a}(mod \ p) \parallel T)$$

$$\tag{4}$$

When the terminal *i* receives the message $auth_i$, ID_i , ID_{TA} , MAC_i , and *T* are separated from $auth_i$. Then, the terminal *i* calculates its own $MAC = hash(ID_i || ID_{parent_i} || k_i || g^a (mod p) || T)$, where k_i is stored in the terminal *i* in the register processing. The *MAC* is compared with the received MAC_i . If the comparison is false, the authentication of the server fails. In addition, checking the timestamp *T* can resist replay attacks.

After passing the server authentication, the terminal *i* generates a random integer *b* and calculates $g^b \pmod{p}$. Then, the terminal *i* sends the response message $resp_i$ to the TA. $resp_i$ is shown in Equation (5), where $MAC'_i = hash(ID_i || k_i || g^{ab} \pmod{p} || (T+1))$.

$$resp_i = ID_i \parallel ID_{TA} \parallel MAC'_i \parallel g^b (mod \ p) \parallel (T+1)$$
(5)

When the TA receives the response message $resp_i$ from $resp_i$ and the reserved information (*a*, K_i , ID_i , ID_{TA} , and T), MAC'_i is easy to check. If it is equal, the authentication of the terminal *i* succeeds. After mutual authentication, the TA builds a secure channel with the terminal *i* and shares the secret $K_i = g^{ab} (mod \ p)$ for key management.

4.4. Key Management

For privacy protection in video surveillance, we propose the Key Management Scheme Based on Normal Tree (KMSNT). KMSNT realizes the group key distribution and provides secure methods for member join and leave to ensure backward and forward security. Aiming to solve slow and complex group key updates, we introduce the CRT to achieve simple and fast group key renewal.

There are three main functions in KMSNT: (1) group formation and group key distribution; (2) group key retrieval; (3) group member change. Additionally, the participants in KMSNT are the TA, monitor, and camera.

4.4.1. Group Formation and Group Key Distribution

According to the devices' authentication information, the TA maintains a large tree that denotes the grouping status of each monitor and its cameras. No matter the monitor or camera, the device *i* obtains the following data after authentication: $\langle ID_i, g_i, b_i, p_i, K_i = g_i^{a_i b_i} (mod p_i) \rangle$. We finish the group formation and group key distribution in this phase.

For each monitor *j*, the TA generates a group key gk_j and a set of prime numbers $\{P_i | 0 < i \le n\}$ randomly. The value *n* is the number of cameras in the group. Then, the TA prepares a message fragment M_i for each camera. The message for the *i*th camera consists of the group key gk_j , the prime number P_i , a timestamp *T*, a message authentication code MAC_i , and the camera ID ID_i .

$$MAC_{i} = hash(ID_{i} \parallel gk_{i} \parallel T \parallel P_{i} \parallel K_{i})$$

$$\tag{6}$$

$$M_i = (ID_i \parallel gk_i \parallel P_i \parallel T \parallel MAC_i)$$
(7)

The final group key distribution message for the monitor $j C_j$ is encrypted by K_i and sent to the group member.

$$C_j = \left(\left\{ ID_i, \ E_{K_i}(M_i) \right\}_{i=1}^n \right) \tag{8}$$

where *n* is the number of cameras belonging to the monitor *j*. $E_k(x)$ denotes that the information *x* is encrypted using the key *k*.

4.4.2. Group Key Retrieval

When the camera *m* receives the ciphertext C_j sent from the TA, the corresponding ciphertext fragment is intercepted according to the camera ID ID_m . Then, M_m is released after the decryption. The camera validates the integrity of MAC_m and checks the timestamp *T*. When the validation succeeds, the camera obtains the group key gk_j and the prime number P_m correctly.

Before the monitor j' accesses the encrypted data produced by its cameras, the monitor j' should be verified following the steps in Section 4.3. Then, the TA sends the group key message to the monitor j'. The group key is released following the same steps as those with the camera.

4.4.3. Group Member Change

In order to ensure the security of the group key, the group key should be updated when the group members are changed. Member join and leave are two basic operations. Complex group changes, such as group merging and deletion, are made up of these two operations. In general, a group member change for video surveillance systems mainly occurs in the cameras because only the cameras are located on the leaf node.

Member Join. When a new camera wants to join the group, the TA should update the group key to ensure the new member cannot access the old key. This guarantees the backward security. Using the CRT, we realize a convenient key update.

Firstly, the TA calculates the value $N = \prod_{i=1}^{n+1} P_i$ where P_i is the secret prime number of the camera *i* in the group. Additionally, *n* is the member number of the original group, and P_{n+1} is the secret prime number of the new camera.

Secondly, the TA generates a new random key gk'_j for the group *j*. According to the CRT, we can obtain the following equations:

 $\begin{cases} x \equiv \left(K_1 \bigoplus gk'_j\right) \pmod{P_1} \\ x \equiv \left(K_2 \bigoplus gk'_j\right) \pmod{P_2} \\ \dots \\ x \equiv \left(K_i \bigoplus gk'_j\right) \pmod{P_i} \\ \dots \\ x \equiv \left(K_n \bigoplus gk'_j\right) \pmod{P_n} \\ x \equiv \left(K_{n+1} \bigoplus gk'_j\right) \pmod{P_{n+1}} \end{cases}$ (9)

 $K_i = g_i^{a_i b_i} \pmod{p_i}$ is the secret shared between the TA and the camera *i*. The value P_i is the secret prime number produced by the authentication process. Since P_1, P_2, \ldots , P_n, P_{n+1} are a set of prime numbers selected randomly, they are pairwise relative prime integers and $K_i \bigoplus gk'_j \in \mathbb{Z}$ ($i \in [1, n+1]$). Therefore, the equations group has a unique solution modulo *N*.

$$x \equiv \sum_{i=1}^{n+1} \left(K_i \bigoplus gk'_j \right) \times \frac{N}{P_i} \times \left[\left(\frac{N}{P_i} \right)^{-1} \right]_{P_i} (\text{mod } N)$$
(10)

For each *i*, there is $\frac{N}{P_i} \times \left[\left(\frac{N}{P_i} \right)^{-1} \right]_{P_i} \equiv 1 \pmod{P_i}$. $\left[\left(\frac{N}{P_i} \right)^{-1} \right]_{P_i}$ is the inverse of $\frac{N}{P_i}$ modulo P_i . Then, *x* is sent to the whole new group members. When the *i*th camera receives *x*, the new group key gk'_i is renewed by the following equation:

$$gk'_{i} = (x \bmod P_{i}) \bigoplus g_{i}^{a_{i}b_{i}}$$

$$\tag{11}$$

Member Leave. When the old member leaves the current group, the TA should update the group key to maintain the forward security. The leaving member can no longer access the group key. For the traditional group key distribution scheme, the group key update finishes after (k - 1)d rounds of information exchange. The value *k* is the number of branches of the tree, and *d* is the depth of the tree. However, with the CRT, we only need one round of message exchange.

The steps of the group key update in member leave are similar to those in the member join process. To transmit the new group key gk'_j secretly, the reserved member in the group can make up the following equations:

$$x \equiv (K_1 \oplus gk'_j) \pmod{P_1}$$

$$x \equiv (K_2 \oplus gk'_j) \pmod{P_2}$$

$$\dots$$

$$x \equiv (K_i \oplus gk'_j) \pmod{P_2}$$

$$\dots$$

$$x \equiv (K_{n-1} \oplus gk'_j) \pmod{P_{n-1}}$$
(12)

Then, *x* is sent to the reserved group members. When the *i*th camera receives *x*, the new group key gk'_i is built by the following equation:

$$gk'_{i} = (x \bmod P_{i}) \bigoplus g_{i}^{a_{i}b_{i}}$$
(13)

4.5. Secure Video Decryption

Video encryption has been fully discussed in many papers. We do not provide any new video encryption algorithm in this manuscript. However, we attempt to solve a new problem which is exposed in video surveillance systems. In some realistic video surveillance scenes, such as shops and families, the monitor is often a personal computer without any professional protection. This leads to the decryption environment of the monitor being complex and vulnerable. Based on transparent encryption, we designed a secure decryption scheme to resist vulnerable environments. The proposed scheme for monitors consists of a user module and a kernel module. The former is used for monitor authority control to communicate with the TA to obtain the decryption key. The kernel module finishes the video integrity validation and executes secure decryption to protect the plaintext from modification.

4.5.1. Authority Control in User Module

Authority control is realized for monitors to obtain the decryption key legally. We assume that the monitor has registered in the TA earlier. Firstly, the monitor and TA verify each other according to the steps described in Section 4.3. If the validation fails, the video decryption process is finished. The monitor with a failed verification cannot access the encrypted video. Otherwise, the monitor requests the decryption key from the TA following the steps in the Group Key Retrieval function (Section 4.4).

The monitor *m* sends ID_m to the TA and begins the decryption key request process. The TA receives the request and prepares a message M_m for the monitor. The message consists of the corresponding group key gk_m , the prime number P_m , a timestamp *T*, a message authentication code MAC_m , and ID_m .

$$MAC_m = hash(ID_m \parallel gk_m \parallel T \parallel P_m \parallel K_m)$$
(14)

$$M_m = (ID_m \parallel gk_m \parallel P_m \parallel T \parallel MAC_m)$$
⁽¹⁵⁾

The final group key distribution message C_m for the monitor *m* is encrypted by K_m and sent to the monitor *m*.

$$C_m = \{ ID_m, E_{K_m}(M_m) \}$$
(16)

For the monitor *m* and TA, K_m is the shared secret produced after the authentication process. Then, the monitor can easily decrypt the group key gk_m as the decryption key from the group key distribution message C_m . All operations take place in the user module of transparent encryption to ensure that the sensitive information including K_m and gk_m is not leaked.

4.5.2. Secure Decryption in Kernel Module

When the monitor obtains the decryption key gk_m correctly, the ciphertext will be decrypted using gk_m in the kernel module of transparent encryption. This process is depicted in Figure 4.



Figure 4. Encryption and decryption process.

We briefly describe the video encryption process firstly. For each encrypted multimedia segment, the encryption information shown in Table 1 is attached to the file header. Assuming that the plaintext *fData* is input into the encryptor, and the group key *fKey* has been prepared well, encryption follows the steps below:

- (1) Encrypt *pData* with *fKey* and obtain the ciphertext *cData* = *fKey*(*pData*);
- (2) Generate the digest of *cData* represented by *digest* and put *digest* in the header of *cData*;
- (3) Set *encFlag* to *ENC* in the prefix, which means the following data are encrypted;
- (4) Pad the file prefix up to a multiple of 32 bytes;
- (5) Count the number of padding data *PSize*. Then, the output is

Table 1. Prefix format.

Name	Size (Byte)	Meaning
encFlag	4	If the file is encrypted
PSize	4	Size of file prefix
Digest	32	Digest of cipher data
Padding	-	Prefix padding

For symmetric encryption, decryption is the inverse process of encryption. However, in practical scenarios, decryption has to face more security risks than encryption. Specifically, (1) the environment for decryption may not be secure enough; and (2) the decrypted

information may be abused or forged. In our scheme, we use two filters in the kernel module, named the process filter and the file filter, to ensure the security of the decryption environment and prevent the plaintext from being abused.

The process filter aims to prevent the plaintext from being leaked out by malicious applications or processes. During the filtering process, the validity of the application in the receiver side will be checked. The MD5 value of the application will be compared with the trusted application list. When the application comparison result is true, we set Flag = True. At the same time, the process filter uses writing blocking to reject the writing operation from any application or process.

After the process filter, file filtering is used to verify the integrity of the encrypted files and decrypt the ciphertext. If the encryption flag in the file header is true, the file integrity will be verified. File filtering has two steps, pre-operation and post-operation, as shown in Table 2. The cipher prefix (including *digest* and *encFlag*) preset in the file prefix will be read out, and *Flag* is checked in pre-operation. In post-operation, a new generated *digest*1 will be compared with *digest* to check whether the file data are identical to the original version before the decryption key request is sent out. Then, the ciphertext *cData* will be decrypted.

Table 2. File filtering process.

Pseudocode 1: Pre-Operation	Pseudocode 2: Post-Operation			
Input: application comparison result <i>Flag</i> , file path <i>f path</i> . Output: preset digest of cipher data <i>digest</i> .	Input: preset digest of cipher data <i>digest</i> , read the file index <i>findex</i> and the data <i>fdata</i> . Output: read file data <i>fdata</i> .			
 var prefix = read_file_header(fpath); if prefix = /ENC/ AND Flag = True then var digest = prefix.digest; var psize = prefix.psize; file_read_start_position + = psize; End 	 var cData = fdata; var digest1 = generate_digest(cData); if digest1 = digest then var key = get_fileKey(findex); var pData = decrypt(cData, key); fdata = pData; End 			

As the core function, the decryption process is realized based on a Minifilter. Integrity is inspected using the message digest algorithm simultaneously while accessing a file. In addition, sensitive applications and processes are kept from writing operations to eliminate data leakage and modification.

5. System Analysis

In this section, we analyze the security and the real-time performance of the proposed video surveillance scheme. In the security analysis part, we mainly focus on the forward/backward security of the group key management scheme and the monitor environment security during the video decryption process. For the real-time performance, we believe that the requirement of real time in video surveillance systems is in monitor decryption. There are two reasons. First, when the video surveillance system works smoothly, it is not common for a camera to join or leave frequently. Second, the camera that captures and encrypts the video can be specified and customized to meet the real-time performance. However, the monitor may be a personal computer with poor properties or a dedicated server with a strong calculation capacity. Therefore, it is more meaningful to discuss the real-time performance of decryption in monitors. In monitor decryption real-time performance analysis, the experiments were implemented by PHP and C++ and run on a computer (Windows 7 32-bit OS, 4*3.2 GHz CPU, 12 GB memory, and 16 MB + 7200 r/min HDD). All tests were carried out under the Gigabit LAN.

5.1. Security Analysis

In this section, we discuss the forward and backward security of the group key management scheme and the security of the monitor decryption environment.

5.1.1. Forward Security

When any leaf node of the tree built in our key management scheme is evicted from the group, the group key update algorithm must guarantee forward security to prevent the evicted node from reading the new message. As shown in Section 4.4, the new group key is renewed by the equation $gk'_i = (x \mod P_i) \bigoplus g_i^{a_i b_i}$, where *i* belongs to the remaining members in the group. The new group key is randomly generated by the TA and has no relation with the old key. For the evicted member, it is necessary to guess the secret parameters P_i and $g_i^{a_i b_i}$ that belong to one of the participating members. However, $g_i^{a_i b_i}$ is shared only between the TA and the *i*th camera, and P_i is also the secret prime number only known by the TA and the *i*th camera. Therefore, a brute force attack is the only way for the evicted member to obtain the correct group key. Hence, forward security is maintained.

5.1.2. Backward Security

Similar to forward security, the secure anchor is on the security of the secret $g_i^{a_i b_i}$. As we know from Section 4.3, the $g_i^{a_i b_i}$ is shared between the TA and the *i*th camera. The secrecy of $g_i^{a_i b_i}$ is guaranteed by using the computational Diffie–Hellman assumption (CDH). To resist MITM and replay attacks, we add a message authentication code (*MAC_i*) and timestamp *T* into the CDH process. In *MAC_i*, k_i is only shared between the TA and the *i*th terminal, meaning the MITM attack cannot work. The timestamp *T* is obtained when the authentication really occurs, and both the TA and terminal will verify the validity of *T*. The invalid *T* will cause an authentication failure. Then, the replay attack is resisted.

5.1.3. Decryption Security

From the perspective of confidentiality, sensitive information includes multimedia data and secret keys. The main risk of them is that they may be stolen through devious ways. In a monitor, a sensitive video can be accessed only when the decryption is executed with a correct key. Additionally, after decryption, a Minifilter ensures that decrypted data are only obtainable for verified trusted applications until they are destroyed and will not be leaked out by disk writing or network transmission. The secret key is stored in the kernel memory and insulated from unauthorized access until it is destroyed, meaning the security of the secret key in the monitor can be maintained.

From the perspective of integrity, multimedia may be tampered or damaged. However, tampering definitely changes the digest of the file. When accessing a forged file, file filtering usually fails because the file cannot pass the integrity verification. Even if the digest on the file prefix can be replaced by that of the forged file, it is impossible to obtain a correct key that was indexed by the original file digest. Therefore, tampering will be detected immediately. Damage makes encrypted multimedia unusable. However, authorized users can retrieve an original copy from the cloud at any time. Unauthorized users cannot obtain any useful information from the damaged file.

5.2. Real-Time Analysis

5.2.1. Secure Decryption Speed

As mentioned before, we focus on the decryption speed in the monitor. When the monitor obtains the decryption key, the next step is to use the key to decrypt the encrypted multimedia. In the decryption process, the validity of the application and the integrity of multimedia will be verified firstly. In the decryption test, we consider the efficiency of the decryption, that is, the decryption speed. The validity of the application happens before decryption, and it needs to communicate with the server. In the actual implementation, we submit the MD5 value of the application together with the key request to the server. That ensures all operations in filters are local.

Therefore, there are two factors that affect the efficiency of the transparent decryption, the AES decryption speed and the digest generation speed. In this experiment, we used 128-bit ECB mode AES for encryption, and 256-bit MD5 for digest generation. The result is shown in Figure 5. In the total 30 tests, the first 10 are the results of only performing AES decryption; the next 10 are the result of only performing digest generation; the last 10 are the performance of the whole transparent decryption. As we can see from Figure 5, the AES decryption speed is about 132.16 Mb/s, and the digest generation speed is about 2.3 times the decryption speed, at about 303.57 Mb/s. Generally, the total transparent decryption speed is about 91.47 Mb/s.



Figure 5. Speed of transparent decryption.

5.2.2. Comparison

The efficiency of multimedia encryption and decryption is closely related to the encryption method and encryption algorithm. In our demo, we adopt full encryption for multimedia, including videos. The decryption speeds for different types of multimedia using different encryption algorithms are shown in Table 3. Compared with the image encryption in [35–37] and the video encryption in [38], our scheme provides a much faster decryption speed, reaching 91.47 Mb/s. This indicates that our scheme can be applied in the decryption environment of the reference schemes and provide security and a real-time decryption service. In addition, if the decryption algorithm changes from full encryption to selective encryption, the demo can realize a better real-time performance.

Table 3.	Decryption	speed	comparison.

Schemes	[35]	[36]	[37]		[38]		Proposed Scheme
Multimedia type	Image		Video		All		
Encryption algorithm	Chaos-based		AES	RC6	DES	All	
Speed (Mb/s)	0.45	1.85	1.87	17.89	8.94	3.59	91.47

6. Conclusions

More and more video surveillance systems have been deployed in the smart cities now. The security of video surveillance faces serious challenges, especially video data. The main challenge of data security faced by video surveillance systems is how to ensure the video integrity, confidentiality, and availability. The key to solving the problem is to build an appropriate scheme for device authentication, key management, and data encryption/decryption to prevent criminals from attacking the system on different attack surfaces. In this manuscript, we built a secure video surveillance model which includes four participants: trusted authority (TA), media cloud, monitor, and camera. Then, terminal registration, mutual authentication, and a group key management method named KMSNT were implemented. By adding secrets and timestamps into the Diffie–Hellman protocol, the proposed mutual authentication protocol can resist MITM attacks and replay attacks. Security analysis proved KMSNT can guarantee the forward and backward security of the key update. Considering the decryption environment in monitors of small-scale video surveillance systems may not be safe, we constructed a video security decryption suite based on transparent encryption. The suite provides a secure environment for identity authentication, key acquisition, and data decryption in the monitors. At the same time, it can also prevent the terminal from illegally copying and modifying the video. The video security decryption scheme has a fast decryption speed which can reach 91.47 Mb/s on average.

Future work. We are considering building a more complex tree to achieve more practical key management requirements in video surveillance systems, such as regional division and cross-level access. We will also try to optimize the video encryption scheme to improve the encryption efficiency and reduce the encryption consumption in the cameras.

Author Contributions: Writing—original draft preparation, formal analysis, H.L.; conceptualization, software, methodology, T.X.; supervision, resources, L.D.; supervision, writing—review and editing, P.Y.; supervision, project administration, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the China Postdoctoral Science Foundation 2019M663358, Fundamental Research Funds for the Central Universities of the Communication University of China (CN) (CUC19ZD003), and the National Key R&D Program of China 2019YFB1706003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

- Nazare, A.C., Jr.; Schwartz, W.R. A scalable and flexible framework for smart video surveillance. *Comput. Vis. Image Underst.* 2016, 144, 258–275. [CrossRef]
- 2. Shao, Z.; Cai, J.; Wang, Z. Smart monitoring cameras driven intelligent processing to big surveillance video data. *IEEE Trans. Big Data* **2018**, 4, 105–116. [CrossRef]
- Sultana, T.; Wahid, K.A. IoT-guard: Event-driven fog-based video surveillance system for real-time security management. *IEEE Access* 2019, 7, 134881–134894. [CrossRef]
- Obermaier, J.; Hutle, M. Analyzing the security and privacy of cloud-based video surveillance systems. In *Proceedings of the 2nd* ACM International Workshop on IoT Privacy, Trust, and Security; Association for Computing Machinery: New York, NY, USA, 2016; pp. 22–28.
- Zhou, J.; Cao, Z.; Dong, X.; Vasilakos, A.V. Security and privacy for cloud-based IoT: Challenges. *IEEE Commun. Mag.* 2017, 55, 26–33. [CrossRef]
- 6. Du, H.; Chen, L.; Qian, J.; Hou, J.; Jung, T.; Li, X.-Y. PatronuS: A system for privacy-preserving cloud video surveillance. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1252–1261. [CrossRef]
- Guo, J.; Zheng, P.; Huang, J. An efficient motion detection and tracking scheme for encrypted surveillance videos. ACM Trans. Multimed. Comput. Commun. Appl. 2017, 13, 1–23. [CrossRef]
- Ma, X.; Zhu, B.; Zhang, T.; Cao, S.; Jin, H.; Zou, D. Efficient privacy-preserving motion detection for HEVC compressed video in cloud video surveillance. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 813–818.
- 9. Shifa, A.; Imtiaz, M.B.; Asghar, M.N.; Fleury, M. Skin detection and lightweight encryption for privacy protection in real-time surveillance applications. *Image Vis. Comput.* **2020**, *94*, 103859. [CrossRef]
- 10. Nandhini, S.A.; Radha, S. Efficient compressed sensing-based security approach for video surveillance application in wireless multimedia sensor networks. *Comput. Electr. Eng.* **2017**, *60*, 175–192. [CrossRef]
- 11. Zhang, X.; Seo, S.-H.; Wang, C. A Lightweight encryption method for privacy protection in surveillance videos. *IEEE Access* 2018, 6, 18074–18087. [CrossRef]

- 12. Senior, A.; Pankanti, S.; Hampapur, A.; Brown, L.; Tian, Y.-L.; Ekin, A.; Connell, J.; Shu, C.F.; Lu, M. Enabling video privacy through computer vision. *IEEE Secur. Priv. Mag.* 2005, *3*, 50–57. [CrossRef]
- 13. Zhang, K.; Ni, J.; Yang, K.; Liang, X.; Ren, J.; Shen, X.S. Security and privacy in smart city applications: Challenges and solutions. *IEEE Commun. Mag.* 2017, 55, 122–129. [CrossRef]
- 14. Alsmirat, M.A.; Obaidat, I.; Jararweh, Y.; Al-Saleh, M. A security framework for cloud-based video surveillance system. *Multimed. Tools Appl.* **2017**, *76*, 22787–22802. [CrossRef]
- 15. Du, L.; Zhang, W.; Fu, H.; Ren, W.; Zhang, X. An efficient privacy protection scheme for data security in video surveillance. J. Vis. Commun. Image Represent. 2019, 59, 347–362. [CrossRef]
- Xu, C.; Ren, W.; Yu, L.; Zhu, T.; Choo, K.-K.R. A Hierarchical encryption and key management scheme for layered access control on H.264/SVC bitstream in the internet of things. *IEEE Internet Things J.* 2020, *7*, 8932–8942. [CrossRef]
- 17. Qiu, S.; Cui, Y.; Meng, X. A data encryption and fast transmission algorithm based on surveillance video. *Wirel. Commun. Mob. Comput.* **2020**, 2020, 1–12. [CrossRef]
- Kalbo, N.; Mirsky, Y.; Shabtai, A.; Elovici, Y. The security of IP-Based video surveillance systems. *Sensors* 2020, 20, 4806. [CrossRef]
 Rashwan, H.A.; Solanas, A.; Puig, D.; Martínez-Ballesté, A. Understanding trust in privacy-aware video surveillance systems. *Int. J. Inf. Secur.* 2016, 15, 225–234. [CrossRef]
- Kim, J.; Lee, D.; Park, N. CCTV-RFID enabled multifactor authentication model for secure differential level video access control. *Multimed. Tools Appl.* 2020, 79, 23461–23481. [CrossRef]
- 21. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* 2013, *1312*, 6199.
- Chatterjee, U.; Govindan, V.; Sadhukhan, R.; Mukhopadhyay, D.; Chakraborty, R.S.; Mahata, D.; Prabhu, M.M. Building PUF based authentication and key exchange protocol for iot without explicit CRPs in verifier database. *IEEE Trans. Dependable Secur. Comput.* 2019, 16, 424–437. [CrossRef]
- 23. Kim, J.; Park, N. Role-based access control video surveillance mechanism modeling in smart contract environment. *Trans. Emerg. Telecommun. Technol.* 2021, e4227. [CrossRef]
- 24. Kung, Y.-H.; Hsiao, H.-C. GroupIt: Lightweight group key management for dynamic iot environments. *IEEE Internet Things J.* 2018, *5*, 5155–5165. [CrossRef]
- 25. Magli, E.; Grangetto, M.; Olmo, G. Transparent encryption techniques for H. 264/AVC and H. 264/SVC compressed video. *Signal Process.* 2011, *91*, 1103–1114. [CrossRef]
- Deng, R.H.; Ding, X.; Wu, Y.; Wei, Z. Efficient block-based transparent encryption for H.264/SVC bitstreams. *Multimed. Syst.* 2013, 20, 165–178. [CrossRef]
- 27. Asghar, M.N.; Kousar, R.; Majid, H.; Fleury, M. Transparent encryption with scalable video communication: Lower-latency, CABAC-based schemes. J. Vis. Commun. Image Represent. 2017, 45, 122–136. [CrossRef]
- 28. Cui, J.; Zhang, Y.; Cai, Z.; Liu, A.; Li, Y. Securing display path for security-sensitive applications on mobile devices. *Comput. Mater. Contin.* **2018**, 55, 17–35.
- 29. Pinto, S.; Santos, N. Demystifying arm TrustZone: A comprehensive survey. ACM Comput. Surv. 2019, 51, 130. [CrossRef]
- Asghar, M.N.; Fleury, M.; Makki, S. Interoperable conditional access with video selective encryption for portable devices. *Multimed. Tools Appl.* 2016, 76, 13139–13152. [CrossRef]
- Thiyagarajan, K.; Lu, R.; El-Sankary, K.; Zhu, H. Energy-aware encryption for securing video transmission in internet of multimedia things. *IEEE Trans. Circuits Syst. Video Technol.* 2018, 29, 610–624. [CrossRef]
- 32. Ma, X.; Yang, L.T.; Xiang, Y.; Zeng, W.K.; Zou, D.; Jin, H. Fully reversible privacy region protection for cloud video sur-veillance. *IEEE Trans. Cloud Comput.* 2015, *5*, 510–522. [CrossRef]
- Brakerski, Z.; Kalai, Y.T.; Katz, J.; Vaikuntanathan, V. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, Las Vegas, NV, USA, 23–26 October 2010; pp. 501–510.
- 34. Berti, F.; Pereira, O.; Peters, T.; Standaert, F.-X. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.* 2017, 271–293. [CrossRef]
- Chai, X.; Zheng, X.; Gan, Z.; Han, D.; Chen, Y. An image encryption algorithm based on chaotic system and compressive sensing. Signal Process. 2018, 148, 124–144. [CrossRef]
- Li, Z.; Peng, C.; Li, L.; Zhu, X. A novel plaintext-related image encryption scheme using hyper-chaotic system. *Nonlinear Dyn.* 2018, 94, 1319–1333. [CrossRef]
- 37. Li, H.; Deng, L.; Gu, Z. A Robust image encryption algorithm based on a 32-bit chaotic system. *IEEE Access* **2020**, *8*, 30127–30151. [CrossRef]
- Aljawarneh, S.; Yassein, M.B.; Talafha, W.A. A resource-efficient encryption algorithm for multimedia big data. *Multimed. Tools Appl.* 2017, 76, 22703–22724. [CrossRef]