TECHNICAL SPECIFICATION: AI-based Enhancing of the Smart City Residents' Safety

Sabina Szymoniak
Czestochowa University of Technology
Czestochowa, Poland *sabina.szymoniak@icis.pcz.pl*

Mariusz Kubanek
Czestochowa University of Technology
Czestochowa, Poland *mariusz.kubanek@icis.pcz.pl*

Shalini Kesar
Southern Utah University
Cedar City, USA *kesar@suu.edu*

1. Framework Technical Assumptions

The proposed system consists of four ingredients. The first is a dangerous situation, like an accident on the street. The second ingredient is the IoT device network that monitors the SC. The third ingredient is the trusted server equipped with the appropriate software. The trusted server is also part of the IoT devices' network because each device is connected to this server. The last ingredient is the rescue services. The proposed framework's purpose is to communicate with various devices that will collect information related to the safety of SC residents.

The network can accommodate an unlimited number of intelligent devices. The number of individuals may be limited due to the capacity of the trusted server or the area under patrol. It is presumed that the municipal authorities of intelligent cities aim to enhance the safety of their residents. Thus, they establish a network of interconnected IoT devices, a trusted server, and a chosen device from the Emergency Notification Center (ENC). Intelligent municipal authorities can designate a specific zone within the city or as a surveilled area. Additionally, they may establish a distinct duration for monitoring, which will be contingent on factors such as crime or accident rates.

Based on the underlying assumptions of the system, it will consistently perform four sequential steps. Initially, a problematic circumstance arises. The IoT device captures the photograph of this particular incident. During the second stage, the device transmits the obtained photograph to a trusted server using the Internet connection. The trusted server utilizes AI-powered software to process the acquired photograph. It is imperative to determine whether the situation is complex. During the third phase, the server will promptly inform the ENC if the captured circumstance is hazardous. Upon reviewing the photograph, the personnel of the ENC will determine the dispatch of rescue services to aid injured individuals or to establish a security perimeter around the location.

The suggested framework requires two essential elements. The first component is the server's software, which facilitates the administration of the monitored area and cameras, data collection, communication assistance, and verification of received photographs. The server should possess robust computational capabilities to resolve each task efficiently. To effectively manage the monitored territory and cameras, it is necessary to perform software-enabled activities such as adding, deleting, activating, or deactivating cameras. Data collection necessitates the establishment of a database that securely stores pertinent information regarding hazardous circumstances within intelligent urban environments. The work of authenticating received images

necessitates the utilization of a specifically tailored AI algorithm to determine whether the obtained snapshot depicts a hazardous circumstance. Identifying hazardous circumstances will rely on Convolutional Neural Networks (CNN). The approach will evaluate the degree to which the snapshot accurately depicts a hazardous condition. If the photo is highly accurate, the pertinent information regarding the scenario will be transmitted to the ENC.

The second crucial element of the framework pertains to the server's communication support task. Every device linked to the system will establish communication with a trusted server. Additionally, the server will establish communication with the ENC. Both communications should be bidirectional, among other reasons, to ensure that the devices can confirm the successful delivery of messages to the intended receiver. It is crucial to ensure the proper security of such communication by implementing a security protocol that is compatible with each device. The security protocol is a crucial component of the framework, ranking second in importance.

The security protocol for SCs with monitoring systems consists of three phases: broadcast, contact, and ENC notification. Following the initial implementation of the security protocol in the SC, the framework administrator must incorporate each camera into the system and database, including its geographical coordinates. Additionally, the administrator must generate the required cryptographic entities, such as the device's identifier or a shared symmetric key between the device and the trusted server. As a result, the framework will carry out communication by utilizing preexisting information. By adding each camera to the database, the system will possess knowledge of all devices, circumventing the need for authentication and logging procedures.

2. Security Protocol for Communication in Framework

To better understand the protocol's operation in the context of the framework idea, Figure 1 shows communication flow in the proposed security protocol. In the subsequent subsections, we will describe each protocol's phase, including its specification in Alice-Bob notation.

2.1. Notation used in this protocol

Firstly, we will present a notation used for the proposed protocol description. In our notation:

- $\{M\}_K$ means the message M encrypted by key K,
- D_i means the i-th device that is connected to the network,
- S means the network server,
- ENC means the Emergency Notification Center,
- T_d means a timestamp generated by device d,
- T_S means a timestamp generated by the server S,
- T_X means a timestamp that indicates the time when the S will end its work,
- UID_D means unique device number identifier,
- $\#hash(x)$ means the execution of the hash function on the object x,
- GC_D means the geographical coordinates of the Device,
- PF_i means i-th photo file sent by the Device,
- K_{SD} means the symmetric key shared between the Server and the Device,
- K_{SENC} means the symmetric key shared between the Server and the Emergency Notification Center.

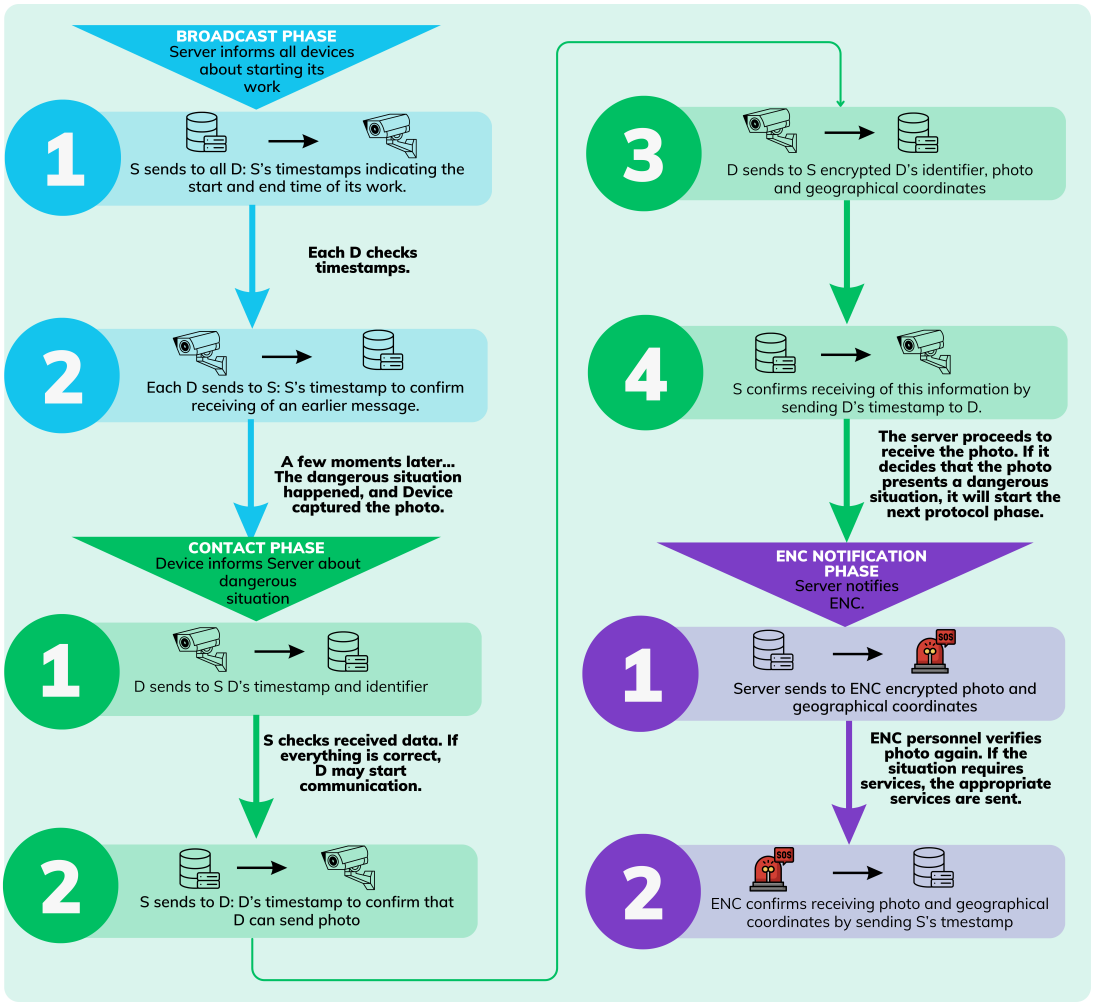

Fig. 1. Communication flow in proposed security protocol.

2.2. Protocol's broadcast phase

The scheme of this phase in Alice-Bob notation is as follows:

$$\begin{aligned} \alpha_1 \quad & S \rightarrow D_i: \quad \{\#hash(T_S) \cdot T_S \cdot T_X\}_{K_{SD_i}} \\ \alpha_2 \quad & D_i \rightarrow S: \quad \{T_S\}_{K_{SD_i}} \end{aligned}$$

The broadcast phase informs IoT devices that the framework will commence operations. The operational duration of the framework in a SC may vary depending on the prevailing circumstances, including the frequency and severity of threats and dangerous situations. It can operate for the entire day or a limited duration, such as during nighttime. Additionally, the framework can oversee the entire city or selectively target specific sections inside the SC. Therefore, the trusted server must inform every device that the framework will commence its operations.

Here, the framework will carry out the broadcast portion of the protocols. This phase has two distinct steps. Initially (step α_1), the trusted server transmits two timestamps to each device, specifying the framework's activity time (T_S and T_X) and hashed T_S to maintain the integrity of the timestamp value. As a response (step α_2), each device shall transmit the timestamp to the server, indicating the beginning time (T_S). The messages in both steps are encrypted using a symmetric key shared between the device and the trusted server (K_{SD_i}).

2.3. Protocol's contact phase

The scheme of this phase in Alice-Bob notation is as follows:

$$\begin{array}{lll} \alpha_1 & D \to S: & \{T_D, UID_D\}_{K_{SD}} \\ \alpha_2 & S \to D: & \{T_D\}_{K_{SD}} \\ \alpha_3 & D \to S: & \{T_D \cdot UID_D \cdot \{GC_D\}_{K_{SD}} \cdot \{PF_i\}_{K_{SD}}\}_{K_{SD}} \\ \alpha_4 & S \to D: & \{T_D\}_{K_{SD}} \end{array}$$

The contact phase initiates a linkage between the device and the trusted server. This phase has four distinct steps. Initially (α_1), the device attempts to authenticate itself to the trusted server and provide a report regarding the potential occurrence of a hazardous circumstance. It transmits its identity (UID_D) and a freshly generated timestamp (T_D) to the server. The trusted server verifies the device's identity by cross-referencing it with its database. Upon successful identity verification, the trusted server transmits the device's timestamp to indicate that communication is feasible, constituting the second stage (step α_2). The messages in both steps are encrypted using a symmetric key shared between the device and the trusted server. During the third step (α_3), the device transmits the recorded photograph (PF_i), device identifier (UID_D), and geographical coordinates (GC_D) to the trusted server. Furthermore, this message is encrypted using a symmetric key shared between the device and the trusted server. Furthermore, this identical key encrypts the photo file and the geographical coordinates to prevent cyberattacks. The trusted server verifies the message's receipt based on the device's timestamp (step α_4). Next, the trusted server can analyze and authenticate the received photograph and inform the ENC if required.

2.4. Protocol's ENC notification phase

The scheme of this phase in Alice-Bob notation is as follows:

$$\begin{array}{lll} \alpha_1 & S \to ENC: & \{T_S \cdot \{GC_D\}_{K_{SENC}} \cdot \{PF_i\}_{K_{SENC}}\}_{K_{SENC}} \\ \alpha_2 & ENC \to S: & \{T_{ENC}\}_{K_{SENC}} \end{array}$$

If the trusted server chooses to inform the ENC, it will carry out the ENC notification phase of this protocol. The final stage comprises two distinct steps. Initially (α_1), the trusted server transmits the acquired photograph and the scenario's geographical coordinates to ENC. Both items are encrypted using a symmetric key shared between the ENC and the trusted server (K_{SENC}). Furthermore, the trusted server appends its timestamp to this message. During the second step (α_2), the ENC verifies the message's receipt and transmits the recently generated timestamp to the trusted server. Subsequently, the ENC personnel will examine the photograph and determine whether to dispatch rescue services to aid injured individuals or safeguard the vicinity.

3. Concept of a System for Recognizing Dangerous Situations

The concept of a system for recognizing dangerous situations will be based on deep learning techniques such as CNN. It will be necessary to prepare a set of training, validation, and testing data, which will then be used to train the created system and analyze and verify the correct operation of the system. The system should assess the extent to which the photo shows a dangerous situation (accuracy). The value of the coefficient will increase the security level of the entire framework. If devices send photos with a low value of this indicator (so-called fake photos), the system may block them to prevent further photo uploads and the use of server resources. The accuracy limit that determines whether an image represents an unsafe situation and whether the user should be blocked will be determined during system verification testing.

One of the possible architectures for detecting and recognizing objects in real-time is the YOLO v4 architecture [1]. The YOLO v4 model is trained using an extensive dataset encompassing many object categories. The YOLO v4 architecture incorporates the following components. The backbone component is tasked with extracting features from input photos. The neck component consolidates multi-scale characteristics extracted from the backbone. The head component of the system is responsible for estimating the probability of different object classes, determining the coordinates of the bounding boxes, and calculating objectness scores.

Our method for recognizing dangerous situations will consist of the following stages. In the first stage, the system will acquire input images from devices connected to the network. Next, the system will apply adaptive histogram equalization and Gaussian filtering-based preprocessing of the received images. The system will perform object detection and recognition and calculate the uncertainty associated with each bounding box prediction during the next stage. Next, the system will exclude inaccurate forecasts by applying a threshold for uncertainty and integrate data from several sources by employing a probabilistic graphical model. As an output, the system will develop an in-depth understanding of the seen situation, encompassing identified objects and their respective positions.

To improve our method, we will consider the following assumptions. We utilize adaptive histogram equalization and Gaussian filtering techniques on the input photos to account for different lighting conditions and sensor noise. By performing this preprocessing step, the contrast is heightened, and noise is diminished, resulting in an overall enhancement of the input data's quality for object detection. We integrate an uncertainty modelling method into the YOLO v4 architecture to manage imprecise input effectively. This process entails the computation of the uncertainty linked to each bounding box prediction. This uncertainty is determined by merging the objectness score and the probabilities of each class. Uncertainty values are subjected to a threshold to exclude incorrect predictions. To enhance the resilience of our approach, we utilize data fusion methodologies to amalgamate data from many sources, including security cameras and drones. The fusion method entails aligning spatial and temporal data and utilizing a probabilistic graphical model [3] to merge the information and generate a cohesive comprehension of the observed scene.

4. Automated Security Analysis Results

Next, we specified the proposed protocol in ProToc language [2] to perform a preliminary investigation of our protocol and check its correctness and security using an automated verification method. We used a tool for automated verification mentioned in [4]. We researched using a computer unit with the Linux Debian operating system, an Intel Core i7 processor, and 16 GB RAM. According to the mentioned tool, we executed a timed analysis (using fixed time parameter values) and simulations of the proposed protocol (using random time parameter values generated according to selected probability distributions). This tool uses an abstract time unit that describes any period. Based on the methodology described in [4], we determined the following assumptions for time parameter values and performed timed analysis:

- time of information generation (like a timestamp): T_g = 3 [tu],
- time of composing the message: T_c = 3 [tu],
- time of symmetric encryption and decryption: $T_e^s = T_d^s$ = 6 [tu],
- time of executing hash function: H = 4 [tu],
- delay in the network range: 1 - 5 [tu].

During timed analysis, the mentioned tool generated a report summarizing the number of cryptographic operations executed in each protocol step. Also, the report contained values of

minimal step time (T^k_{min}), maximal step time (T^k_{max}), timeout for each step (T^k_{out}) and minimal (T^{ses}_{min}) and maximal (T^{ses}_{max}) session time calculated by the tool. Table 1 shows this report. All values are in the mentioned time unit ([tu]).

Table 1. Summary of the number of cryptographic operations executed in each protocol step.

Phase	**Step**	T_g	T_c	T^s_e	T^s_d	H	T^k_{min}	T^k_{max}	T^k_{out}	T^{ses}_{min}	T^{ses}_{max}
Broadcast	1	1	1	1	1	1	23	28	49		
Broadcast	2	0	1	1	1	0	16	21	21	39	49
Contact	1	1	1	1	1	0	19	24	91		
Contact	2	0	1	1	1	0	16	21	87		
Contact	3	0	1	3	3	0	40	45	66		
Contact	4	0	1	1	1	0	16	21	21	91	111
ENC notification	1	1	1	3	3	0	43	48	69		
ENC notification	2	0	1	1	1	0	16	21	21	59	69

The minimal (T^{ses}_{min}) and maximal (T^{ses}_{max}) session time values for each protocol phase were calculated using different values of delay in the network according to assumed delay in the network value range. These values were calculated for each phase separately. The tool generated seven executions for the whole protocol. In these executions, the Intruder was included, so they differed from each other by the participant lists and the Intruder's capabilities. The tool used during research includes the following Intruder models: Dolev-Yao model, Lazy Intruder, restricted Dolev-Yao, and restricted Lazy Intruder [4]. So, the tool analyzed situations where only honest devices communicate with the trusted server and situations where the Intruder tried to cheat the server by impersonating a device from the network. During the timed analysis, the tool did not find attacks for any protocol phase, which means that the Intruder cannot possess the appropriate knowledge (for example, about keys) to execute each step in time. These executions did not meet the imposed time conditions.

Also, the tool can perform simulations of protocols to show how users of the proposed protocol (especially Intruder) may behave in a real network and what situations may occur if the network's delay value exceeds the established limit. The tool generated delay in the network values according to uniform, normal, exponential and Couchy's probability distributions from the assumed range 1 - 5 [tu]. However, the tool can draw values out of range to simulate different loads on the computer network and show when the protocol may fail due to lack of time. We perform simulations for each phase, including executions with honest devices and the Intruder. All the executions with the Intruder did not meet the imposed time conditions because the Intruder could not possess the appropriate knowledge. This means the tool did not find attacks for any protocol phase during the simulations.

Figure 2 summarizes the session time values with the generated network delay values according to the selected probability distributions for the Contact phase of the proposed protocol. We performed one thousand test series for honest executions only. We observed that values generated according to uniform and Cauchy's probability distribution significantly influenced the session time, so the imposed time conditions could not be met. So, uniform and Cauchy's probability distribution simulate very load networks with high delay in the network values. On the other hand, normal and exponential probability distribution simulates networks normally loaded with low or acceptable delays in the network values. All test series ended successfully below the maximal session time value for simulations with normal probability distribution. Only a few test series exceeded the maximal session time value for the exponential probability distribution.

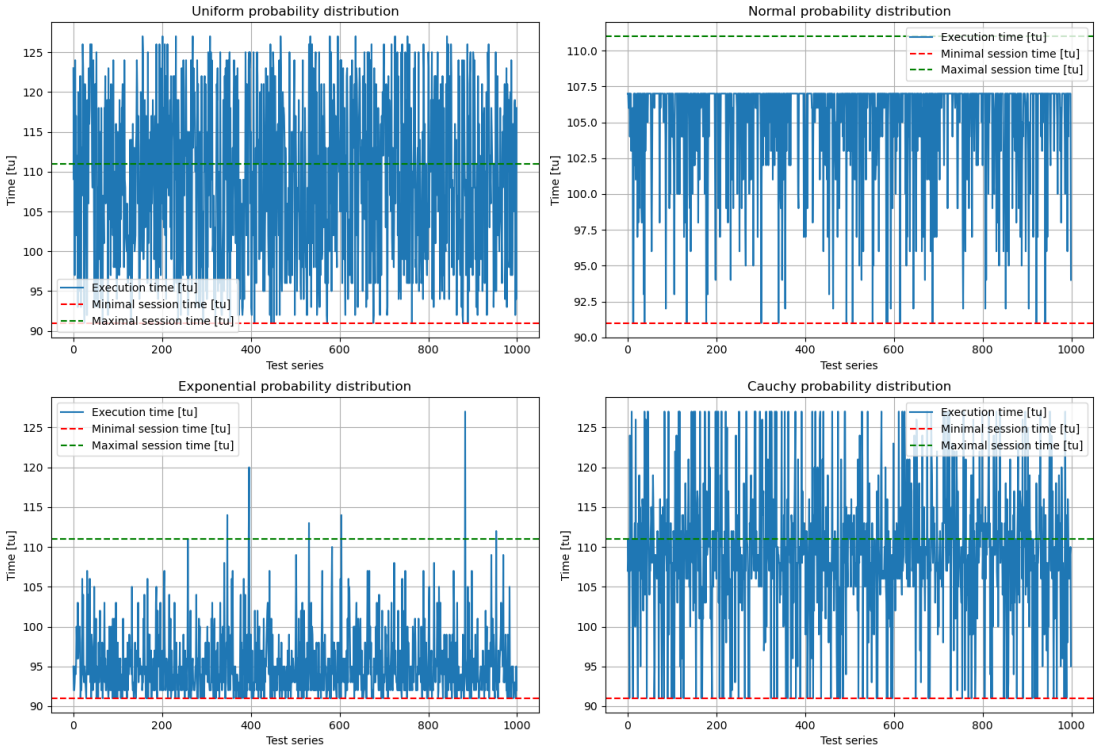

Fig. 2. Time simulations for Contact phase.

References

1. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
2. Andrzej Grosser, Mirosław Kurkowski, Jacek Piątkowski, and Sabina Szymoniak. Protoc—an universal language for security protocols specifications. In *Soft Computing in Computer and Information Science*, pages 237–248. Springer, 2015.
3. Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
4. Sabina Szymoniak. Security protocols analysis including various time parameters. *Mathematical Biosciences and Engineering*, 18(2):1136–1153, 2021.