Article:
Yang, J., Gope, P., Cheng, Y. et al. (2021) Design, analysis and implementation of a smart next generation secure shipping infrastructure using autonomous robot. Computer Networks, 187. 107779. ISSN: 1389-1286

https://doi.org/10.1016/j.comnet.2020.107779

Design, Analysis and Implementation of a Smart Next Generation Secure Shipping Infrastructure Using Autonomous Robot

Jiapie Yang[1], Prosanta Gope[1], Yongqiang Cheng[2], and Li Sun[1]

[1]Department of Computer Science, University of Sheffield

[2]Department of Computer Science and Technology, University of Hull

Email: p.gope@sheffield.ac.uk

***Abstract*—** In general, price is the key element in shipping, and half of the costs are tied up in last-mile deliveries. The biggest expense here is the human element, so companies, which can cut down on staff costs, will be able to out-price their competitors. Therefore, we expect sooner, or later robotic delivery systems will become the norm. However, ensuring security in such a system will be a challenge. In this article, we propose a secure shipping infrastructure using robot. In this regard, we first design a cooperative user authentication system for delivering parcel using crypto primitives such as one-way hash function, asymmetric encryption and QR code. Next, we design a non-cooperative user identification scheme using Siamese Network for Person Reidentification, where the client is not ready to cooperate during the authentication process. Therefore, the robot carrier needs to automatically recognize and go towards the client to deliver the parcel in a secure way. Finally, we implement the proof-of-concept system through a cooperative user authentication process using TurtleBot3 robot platform and analyse the security of the proposed scheme using ProVerif. Analysis and experiment results show that our proposed system can complete the delivery mission and resist a variety of security attacks.

Keywords—Cooperative user authentication system, Non-cooperative user authentication system, Person Reidentification, QR Code.

I. INTRODUCATION

As part of commercial activities, delivery services have developed rapidly in recent decades and played a vital role in both business and personal lives. Online shopping has grown rapidly in the past years. For example, Americans are predicted to spend $709.78 billion online in 2020, 18% larger than that in the previous year. E-commerce accounted for 14.5% of total retail sales [1]. Besides, more people are shopping online and using express services. By 2019 the amount of *Amazon Prime members* has been over 100 million, which still climbs every day [2]. In addition, *Fulfilment by Amazon (FBA)* also rocketed when it was

applied by 73% *Amazon* best-selling sellers in the U.S. in December 2018 [3]. Meanwhile, in China the giant e-commerce platforms *Alibaba, JD.com and Pinduoduo* will account for 83.6% of the retail ecommerce market in 2020, larger than that in 2019 (80.3%) [4]. It is also acknowledged that the development of express services boosts sales because online stores supply a wider variety of goods, so people can acquire items that are not available in nearby stores. Moreover, delivery services also contribute to related industries like transportation, additional cargo aircrafts and upgraded IT technologies in stability and massive workload.

A. Problem statement and motivation

Notwithstanding, the last-mile express includes cost issue especially in the pandemic that can't be overlooked, and robots could be a promising way to solve the problem. According to the *2017 FedEx Annual Report* [5], employee pay and benefits account for 36.9% of total cost in economic activities. Furthermore, shopping online in the pandemic like Covid-19 soars. As a result, Amazon hired an additional 175,000 employees in March and April and let the cost of protective goods such as masks and thermometers boosts, which incurs more employee infections risks and cost [6]. However, robot transmission performs better: not only does the robot avoid infection of postman, but also the contamination on packages. More importantly, fast development of IoT and Smart Cities has made the robot package delivery network even as complex as urban traffic network but significantly more reliable and efficient, which shades light to Delivering Industry 4.0 [7], i.e. direct express from factory to client, and thereby reduce cost on contractors.

B. Related work

Some elementary delivery drone and robotic vehicles have been manufactured nowadays: Amazon proposes Prime Air [9] to use drones in delivering limited-weight goods from warehouses, and successfully completed its first task in December 2016 [9]. This drone is capable to do vertical take-off, landing and forward flight, and complete the work of navigation by using data from sensors. Subsequently, *Google Project Wing* has successfully delivered medicines including Tylenol and cough drops [10], UPS Flight Forward received Federal Aviation Administration (FAA) approval to effectively operate a full-scale “drone airline” package delivery in the USA [11], DHL fully automated drone delivery solution in Germany [12], and Flytrex Sky cloud-connected delivery drone in Israel [13]. Besides, delivery robots are in high demand in China during Covid-19 quarantine, and Chinese e-commerce giant JD.com utilized self-driving robotic vehicle to transport water and food to people in lockdown regions[14]. Even though robotic delivery systems have been prototyped for several purposes, authentication and security technologies specifically designed for the delivery are underdeveloped. Some protocols used in similar IoT fields could be referred, such as [15], [16] that use special PUFs for security, and [17], [18] that use

lightweight key exchange and secure mutual authentication scheme for better speed and security.

C. Our contribution

In this article, we propose a next generation secure shipping infrastructure using robot, which concocts of a novel cooperative authentication process, along with a non-cooperative identification process. The security and performance of the proposed system has been comprehensively analysed using some standard platforms such as ProVerif, etc.

The rest of this article can be organized as follows. In Section II, we provide a brief introduction to QR code, person reidentification, Cumulative Match Curve, etc. In Section III, we present our system model. In Section IV, we give details of our cooperative authentication process and its formal security analysis. In Section V, we present our non-cooperative authentication process. Performance analysis is provided in Section VI. Finally, we conclude our article with concluding remarks in Section VII. The symbols and cryptographic functions used in the proposed scheme are defined in Table 1.

Table 1. SYMBOLS AND CRYPTOGRAPHIC FUNCTION

Symbol	Definition
ids	Identity of the server
tid	One-time-alias identity of the client
K	Session key between client and server
N_c	Random nonce generated by the client
N_s	Random nonce generated by the server
$tidnew$	Next one-time-alias identity of the client
$Tqrgen$	life duration of the authentication message
$UAID$	Serial number of the delivery work
PK_{rob}	Public key to encrypt authentication message for client and robot
PR_{rob}	Private key to decrypt messages from client and server
Δ	Authentication information in the server and robot
∂	Authentication information in the client
$QRCode$	A QR code of the client that contains encrypted ∂
t_start	Time of staring generating authentication message in the server
t_end	Time of checking authentication message in the robot
$h(.)$	One-way hash function
$\oplus$	Exclusive-OR operation
$\|\|$	Concatenation operation

Enc_k	Symmetric encryption using K between client and server
$Enc_{PK_{rob}}$	Asymmetric encryption using PK_{rob}
$p1\_HSV$	HSV image of the first input photo
$p2\_HSV$	HSV image of the second input photo
$p1\_Lap$	Laplacian image of the first input photo
$p2\_Lap$	Laplacian image of the second input photo
W	Weights for all sub-networks
$Gw(p1)$	Combined features of the first input photo
$Gw(p2)$	Combined features of the second input photo

II. PRELIMINARIES

a) QR code

Quick Response (QR) code is a type of 2-D barcode used for authentication. Compared to traditional 1-D bar code, it is capable of storing more information. Besides, error correction code enables it to check the integrity of the code, and different version allows size from 21*21 to 177*177 [21]. It consists of function patterns for position and encoding region for data as shown in Fig. 1 [22]. Essentially QR code is an encryption protocol for text, URL, messages, files, pictures and voice to be converted into a pattern with certain structures, which can be scanned and read by specially designed QR code readers. In this process, QR code is decoded as text at first, then the reader identifies keywords in the header of the text to retrieve the type of the contents. For example, if a URL is encoded in QR code and the reader finds the head *http* in the text, the URL will be sent to the browser. Similarly, for secure QR code, as soon as the reader recognizes the flag of the encryption protocol, it requires the user to type in the key and use it to decrypt the text. These processes can be split into a read operation by common readers to get the text followed by a decryption operation on it.

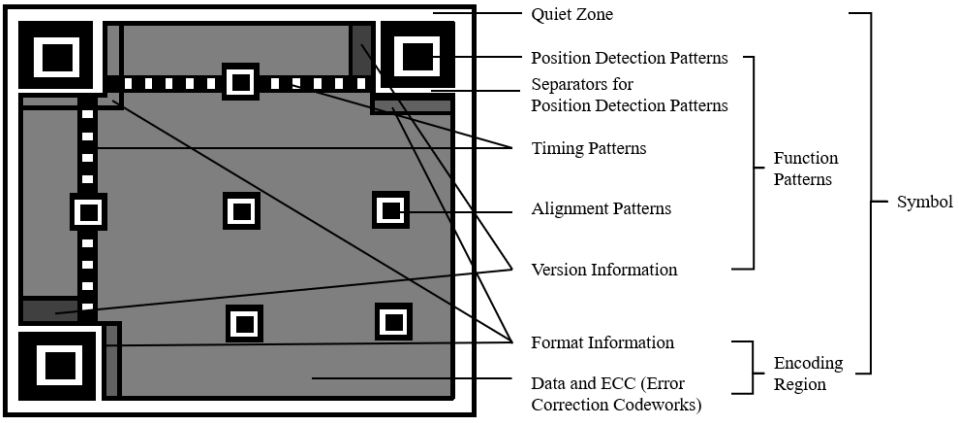


Figure 1. QR code patterns.

b) Person reidentification using machine learning

Person re-identification (Re-ID) is defined as the problem to recognize people in captured photos or videos in diverse times and/or locations over nonoverlapping camera views when a large set of candidates are possible [23]. For example, a person is captured in the first camera and stored in the database, and the second camera could recognize the person when he or she appears in sights, which allows the system to build a track of him or her. Considering the resolution problem, conventional face recognition methods are not suitable for this situation, so a specially designed algorithm for person Re-ID is required. This algorithm is based on the previous work of person detection or person tracking and usually seen as a retrieval problem via a single captured image [24]: Define g as the image gallery (database) that belongs to N identities. For a captured image (query), compute $i^* = argmax_{i \in 1,2 \ldots N} sim(q, g_i)$ where i^* is the identity of q and $sim()$ represents similarity function.

In the process of person Re-ID, the first task is person detection and crop the face image with a unified size. Then machine learning technique is applied on the database to train a model and use it to compare the captured image and photos in the database to get the result of the top N closet identities. The matching strategies can be generally grouped into two categories [25]. The first one is to regard the comparison as a classification problem by comparing the captured image with all pictures in the gallery and applying classification algorithms to find the most N possible recognised identities. Most of the work taking this method is representation learning with features of the person or the image, including [26], [27]. There are also researches using ensemble way to incorporate local prior information with globally trained model to improve the searching efficiency by constraining the searching space [56], [57]. The other stream formulated Person Re-ID as a metric learning problem where researchers focus on the distance of the captured image and a photo from the database to determine the matching result of the two. The training target is to minimize the distance of two photos from the same person and maximize that from different persons, such as contrastive loss [28], triplet loss [29-31], quadruplet loss [32] and Margin sample mining loss (MSML) [33]. Besides, algorithms using neural network to combine features recognition and distance computation has seen a rapid development, such as [34].

The challenges of person Re-ID come from following factors: Posture of people, change of lightness and illumination, pedestrian's absolute scale in the photo and influences from global features (such as colour) and local features (patterns or texture).

c) Cumulative Match Characteristic Curve

Cumulative Match Characteristic (CMC) Curve is a line graph for highest rank fusion performance in the evaluation of identification algorithm. It expresses the possibilities of

the correctly identified images shown in the top N recognition results [35]. For example, to draw the curve, researchers should firstly select an image in the testing set, and compute its similarities with all base images before ranking the results. For every testing photo we repeat the above computation to generate a similarity matrix. For each base image K, if the Nth comparing image belongs to the same person, the value is set as 1, where K≥N. Then calculate the matching rate of top N and draw the CMC of it as shown in Table. 2 and Fig. 2.

Table 2. Example Similarity Matrix for Cumulative Match Characteristic (CMC) Curve

Results / Base	The Nth closet photo									
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Image 1		1	1	1	1	1	1	1	1	1
Image 2			1	1	1	1	1	1	1	1
Image 3								1	1	1
Image 4	1	1	1	1	1	1	1	1	1	1
Image 5				1	1	1	1	1	1	1
Image 6						1	1	1	1	1
Image 7	1	1	1	1	1	1	1	1	1	1
Image 8			1	1	1	1	1	1	1	1
Image 9							1	1	1	1
Image 10									1	1
Top-N Precision	2/10	3/10	5/10	6/10	6/10	7/10	8/10	9/10	10/10	10/0

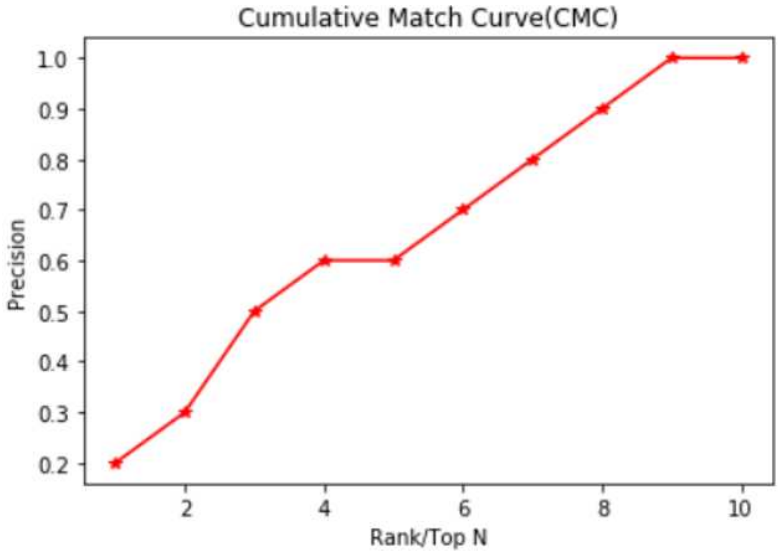


Figure 2. Example of CMC Curve.

d) Siamese Network

Siamese Network [36,37] is a shared-weights neural network used for comparing similarity of two outputs. As shown in Fig. 3, it has input 1 and input 2 that are respectively mapped by two identical sub-networks Network 1 and Network 2 to new spaces. Then we back propagate the gradient through the time till the loss is converged. To output the result whether the two inputs belong to the same identity, researchers can either choose to draw a threshold for two embeddings from sub-networks (metric learning) or merge them by concatenation or adding before inputting into a binary classifier (classification).

The Loss function (EQ. 1) is usually designed with the contrastive loss to get better results of training:

$$L(input\ 1, input\ 2) = (1 - Y) * L_s(input\ 1, input\ 2) + Y * L_d(input\ 1, input\ 2) \quad (\mathbf{EQ.\ 1})$$

Where Y=1 with the loss Ls referring to the situation that two inputs are from the same identity and Y=0 with the loss L_d representing different owners. For designing Loss, Ls should have the same trend of L that the difference between inputs of the same identity decrease when Loss goes down, but L_d has opposing trend to L, which makes the same identity's inputs become closer and the different ones goes further.

It is worth noting that the sub-networks Network 1 and Network 2 are not specified in forms thus they can be chosen according to field of use, such as CNN.

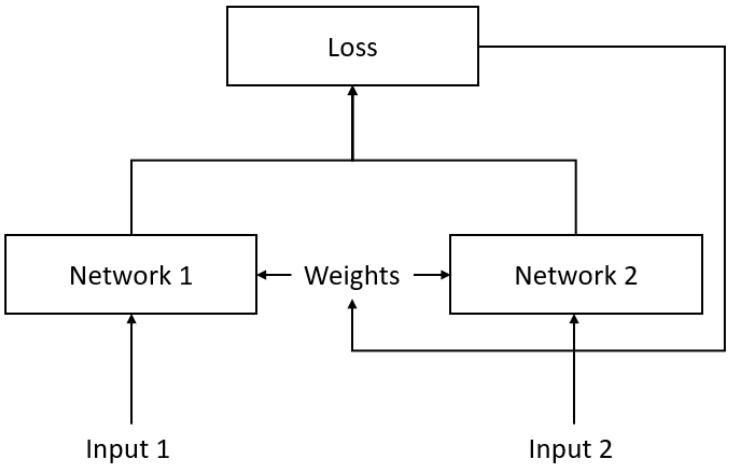


Figure 3. Siamese Network Structure.

e) HSV image

HSV is a global feature which includes Hue, Saturation and Value. HSV can be mapped from RGB color mode [38] from R, G, B that have values of 0-255 to Hue (0-180), Saturation (0-255) and Value (0-255). HSV is more resistant to luminance changes than RGB because HSV resembles more perceptual color models like human eyes, and an individual

luminance component to resist to light change when the other vectors are not influenced by the luminance. Therefore, HSV is more suitable for computer vision, especially recognition tasks.

f) Laplacian image

Laplacian image is a low-level feature of a picture to represent outlines and texture of items. It is generated by putting image pixels values in Laplacian operator (EQ. 2) that is defined as [39]:

$$Laplace(f) = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2} \quad (\mathbf{EQ.2})$$

It computes the 2nd spatial derivative of pixel values. As soon as outlines of person and items or texture of packages and clothes are met, pixel values change quickly as shown in Fig. 4, and it is obvious that the Laplacian 2nd spatial derivative of those edge is 0. Therefore, this feature can be used to derive items' edge of images after filtering noise.

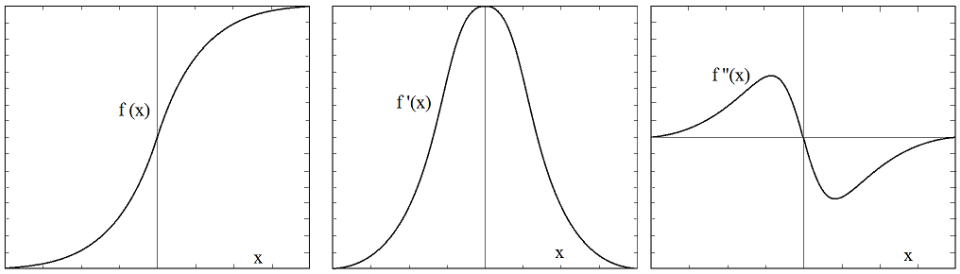


Figure 4. Pixel value and derivatives of outlines and texture.

III. SYSTEM AND ADVERSARY MODEL

a. *System model*

To analyse requirements of robotic delivery, we set a system model as the running situation, and analyse the structure as well as requests. Fig. 5 shows the system model for the secure shipping infrastructure using autonomous robots, which consists of three entities: a client who sends the request and serves as the client of the communication process. This component is in charge of outgoing requests, obtaining messages for authentication like QR codes with the robot, participating in client-robot verification. Therefore, it is the originator of the entire system. The second module is the server who accepts orders from clients over the Internet and gives carriage directions in offline warehouses. The server serves as the server of the communication process. It should fulfil its mandates, including confirming orders, distributing messages for authentication to the client as well as transmitting key information to the robot using the secure channel. The communication between the server and the client are via the public Internet whilst private network of the server is used for message exchanges between the server and the robot.

Lastly, robot fulfils the role of transportation of parcels and handover it to the clients once they pass the checks of authentication using information the client received via both secure and insecure channel from the server to verify whether the client is legitimate and matched.

Assuming that the user, the server and the robot have exchanged some basic and necessary information in a secure channel in advance, the workflow of the whole system can be divided as following steps:

Step S1: The client makes a request via online shopping to the server.

Step S2: Then the server generates a pair of authentication information upon the receipt of the order and confirms the identity of the client. Next, it sends the encrypted authentication information back to the client for the final picking up authentication.

Step S3: The server sends the identical encrypted information to the robot using the insecure channel in the server's private network. Besides, when the robot returns to the warehouse after daily work, the server distributes necessary information for recognition and decryption to the robot through secure channels, for example, a monitored and protected hard copy.

Step S4: The robot then navigates to the destination at the next shift and authenticates the client in cooperative method by decrypting and comparing the client's verification information to that stored in the robot. If they match, a secure robotic delivery mission is completed. Otherwise, the robot switches non-cooperative authentication modes to try to recognize and identify the client until the max waiting time expires.

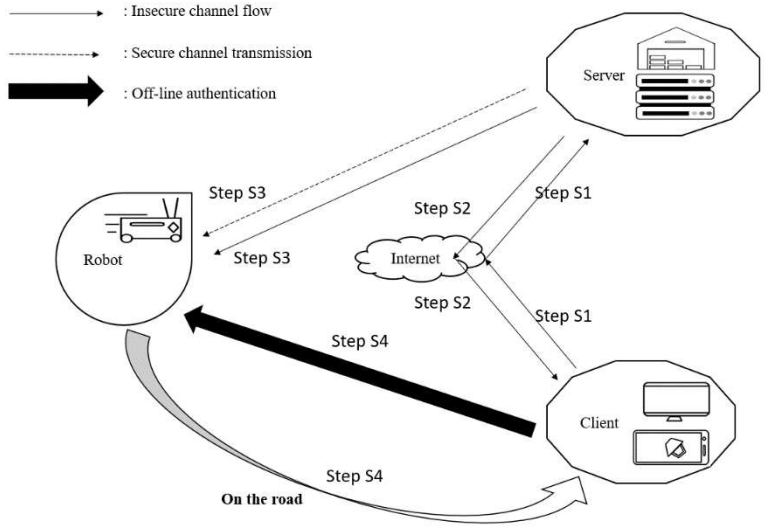


Figure 5. System model for the proposed system.

B. Off-line authentication approaches

Fig. 6 illustrates two methods of the off-line authentication process (as shown in Fig. 5) i.e., cooperative off-line authentication and non-cooperative off-line authentication. To begin

with, the robot waits for the client to present the authentication messages for the cooperative authentication process. When the QR code is successfully scanned, we deliver assignment completes successfully. Otherwise, the camera continues to detect QR code until it reaches the max number of attempts for scanning. Then the robot switches to the non-cooperative authentication mode where it uses the camera to capture photos and automatically do people detection as well as person re-identification using the trained model to find the target client. If the robot can identify the client, then it asks the client to present the verification information. In addition, the Robot also checks whether the delivery mission is timeout for the possibilities of cancellation.

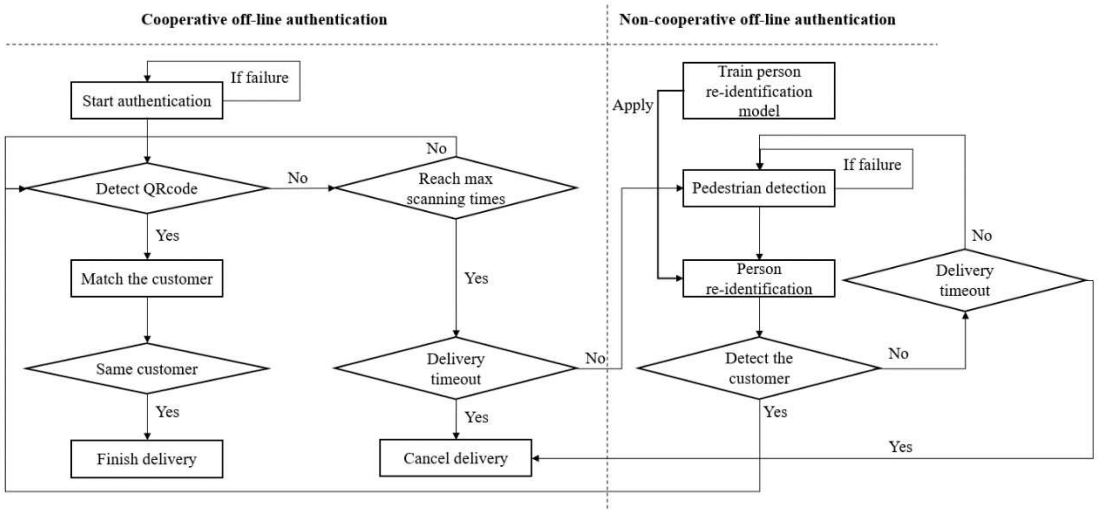


Figure 6. The off-line authentication approaches of the proposed system.

C. Adversary model

Here, we consider the so called *Dolev-Yao model* [40], in which there are possibilities that for an adversary to eavesdrop information or modify them, leading to the failure of proper system functioning. In addition, confidentiality and privacy are also important issues under this circumstance by considering an adversary pretending to be the client and tricking the robot. Consequently, the system must establish a secure key distribution scheme to ensure authentication, availability as well as confidentiality so that it works well with correct client and rejects adversaries.

IV. COOPERATIVE AUTHENTICATION PROCESS OF THE PROPOSED SCHEME

In this section, we will introduce our cooperative user authentication process of the proposed scheme, which has been designed for the robotic-based courier services. We then give formal and informal analysis on the security of the cooperative authentication process. Our authentication process involves clients as clients, servers as servers and robots. The

client obtains encrypted verification information from the server, and then generates a QR code after decryption on the information. In parallel, when the robot receives encrypted messages and the necessary auxiliary information from the server, and then it verifies the messages corresponding to the QR code. Our cooperative user authentication process includes two phases: pre-authentication phase and the authentication phase.

a. *Pre-authentication phase*

Before executing the authentication phase of the proposed scheme, the client and the server exchange their device IDs and generate the key for the following communication. This phase can be divided into two steps:

Step P1: The client sends a registration request message to the server, along with the client ID through a secure channel.

Step P2: When the server receives the request from the client, it creates the client's account in the database. Next, a random and non-reusable parameters *tid* and K are generated by the server, where *tid* represents client's temporary id that can only be used only once and K denotes the secret key shared between the client and the server as a shared symmetric encryption key between them. After this, the server stores $\{tid, K\}$ in the database.

Step P3: Next, the server sends "*tid* and K" to the client through a secure channel.

B. Authentication phase

In order to achieve secure communication and authentication, the server must confirm that the request is from a legitimate registered client, and the client ensures that the returning information is from the correct server who owns the device identity ids and the shared key K. In this phase, the server distributes encrypted QR code authentication information both to the robot and the client, and they will calculate theirs to the same value. QR code is scanned and read by the robot so the two values can be compared. In addition, with regular updating, QR code, generally speaking, has considerably low possibilities of being stolen from the terminal device or in the showing time because this action only lasts for a short time and the short distance between the reader and the terminal means peeking is not easy. In this way, QR code scanning can be considered as safe. To be more specific, the certification process can be divided into the following steps as depicted in Fig. 7.

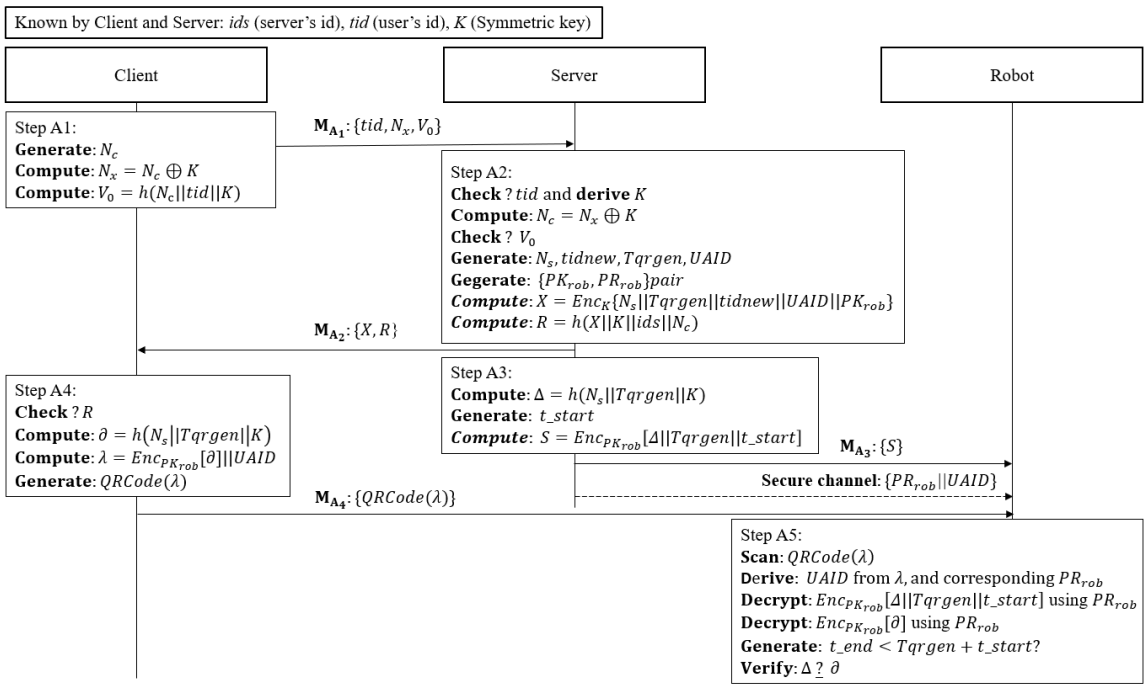


Figure 7. Steps for cooperative authenticated proposed scheme.

Step A1: Client generates a random nonce N_c, and uses the shared symmetric key K to calculate $N_x = N_c \oplus K$, and also computes a key-hash [19-20] value $V_0 = h(N_c||tid||K)$. Then it composes a service request message $\mathrm{M_{A_1}}: \{tid, N_x, V_0\}$ and sends it to the server.

Step A2: Upon arrival of the message $\mathrm{M_{A_1}}$, the server finds the tid in the database. If it can find the *tid*, then obtains the corresponding key K. Then decodes N_c by computing $N_c = N_x \oplus K$. Next the server computes and checks the value of $h(N_c||tid||K)$ with V_0 received in $\mathrm{M_{A_1}}$. Subsequently, the Server generates a nonce N_s, and $tidnew$ (a new temporary client id for next communication), $Tqrgen$ (life duration of the authentication QR code.) and $UAID$ (serial number for index). Next, the server randomly generates a pair of asymmetric key $\{PK_{rob}, PR_{rob}\}$: A private key PK_{rob} and a public key PR_{rob}, which are created for each request and used only once. Thereafter, server computes $X = Enc_K\{N_s||Tqrgen||tidnew||UAID||PK_{rob}\}$ to deliver PK_{rob} to the client, as well as $R = h(X||K||ids||N_c)$ for the aim of checking the integrity of X. Finally, server composes and sends $\mathrm{M_{A_2}}: \{X, R\}$ to the client.

Step A3: In this step server firstly generates the authenticated key-hash value $\Delta = h(N_s||Tqrgen||K)$ that will be provided to the robot. Next, server sets the start point of authentication QR code life duration t_start, and performs an asymmetric encryption $S = Enc_{PK_{rob}}[\Delta||Tqrgen||t\_start]$. Next, the server composes and sends a message $\mathrm{M_{A_3}}: \{S\}$ to the robot using the secure local area network (SLAN) of the warehouse of the delivery company (such as amazon). After that, the server also sends $\{PR_{rob}||UAID\}$ to the robot through the secure SLAN.

Step A4: Next, upon arrival of message $\mathrm{M_{A_2}}$, the client verifies the parameter R. When the verification succeeds, the client decrypts X to update $tidnew$ and obtains N_s, $Tqrgen$, $UAID$ and PK_{rob}. Next, the

client computes the authenticated key-hash value $\partial = h\left(N_s \left|\left|Tqr_{gen}\right|\right| K\right)$ and with the public key PK_{rob} and serial number $UAID$, client also calculates $\lambda = Enc_{PK_{rob}}[\partial]||UAID$. Finally, the client generates a QR code $QRCode(\lambda)$ and sends $QRCode(\lambda)$ to the robot in the message M_{A_4}: $\{QRCode(\lambda)\}$.

Step A5: After reaching to the delivery place, the robot turn-on its camera and tries to scan the QR code. Upon a successful scanning of $QRCode(\lambda)$ from the client, robot validates $UAID$ derives from λ . Next, robot decrypts S (received from the server) using PR_{rob} to get the authenticated key-hash value Δ, its life duration $Tqrgen$, and life's start point t_start. Then records current time t_end, and checks whether $t\_end < Tqrgen + t\_start$ to check the timeout-time of the authentication of the QR code. Finally, the robot verifies Δ with ∂. By successfully verification on $\Delta \stackrel{?}{=} \partial$, the robot confirms the right client and delivers package.

For the above scheme, some nonce can only be used once, including the client device identity tid, asymmetric encryption keys $\{PK_{rob}, PR_{rob}\}pair$ and request serial number $UAID$. In addition, any of the unsuccessful verification steps can result the termination of the execution of the protocol. When robot tries to scan a valid QR code for several times and fails repeatedly, the robot considers that the client is not active in cooperation mode, it then triggers the non-cooperative scheme. In case the recognition of the client in non-cooperative mode is successful, robot will switch back to cooperative mode to start scanning QR code. If the correct QR code has not been received for a long waiting time in a delivery task, the robot will cancel current delivery.

C. Informal Security Analysis of the Cooperative Authentication Process

In this section, we show how our proposed scheme can ensure security against several imperative attacks such as forgery attacks, modification attacks and interruption attacks.

a) Forgery attacks:

In the proposed scheme, attackers will not be able to impersonate as a client, since he/she cannot produce a valid one-time used tid (Identity number of the client) and secret K (Shared key between the client and the server). Precisely, in our proposed scheme in order to construct and send a legitimate request at the beginning, the client calculates $N_x = N_c \oplus K$ and $V_0 = h(N_c||tid||K)$, where only a legitimate client knows the secret key K. Then the server uses tid to identify the client, and authenticates it by checking V_0. However, tid is used only once and attackers don't know the latest tid. Besides, attackers will not able to derive K, hence e/she cannot get N_c from N_x to compute correct V_0. In this way, they fail to act as a client. Besides, the attacker cannot act as a fake client in the QR scanning step because of the one-time used public key PK_{rob}.. In this period, the server generates a pair of one-time used asymmetric keys. Without knowing PK_{rob}., attackers cannot compute $\lambda = Enc_{PK_{rob}}[\partial]||UAID$, which is stored in the QR code. So, they cannot show correct authentication QR code to the robot to impersonate as a legitimate client.

b) Eavesdropping attacks:

In the proposed scheme, if an attacker eavesdrops/intercepts any authentication message communicated between the entities, then he/she cannot obtain the sensitive information, because the attacker is not familiar with the of the shared secret key K and one-time used asymmetric keys PK_{rob}, PR_{rob}. For each client's request, the server generates a pair of asymmetric keys randomly. When the server replies to the client with $X = Enc_K\{N_s||Tqrgen||tidnew||UAID||PK_{rob}\}$ and $R = h(X||K||ids||N_c)$, attackers are unable to decrypt X because of the unexposed K, and cannot derive information from the irreversible one-way hash values R. Similarly, they still fail to decrypt $S = Enc_{PK_{rob}}[\Delta||Tqrgen||t\_start]$ without PK_{rob} in the message from the server to the robot. Therefore, even if messages in the system are captured by attackers, they cannot obtain information in it.

c) Modification attacks:

The proposed scheme is resistant to modification attacks because in our proposed scheme we utilize a secure *one-way-has-function* to check integrity of the transferred messages communicated throughout the authentication process. When the client sends a request message $M_{A_1}: \{tid, N_x, V_0\}$ to the server, any modification in M_{A_1} leads to the computing value$V_0 = h(N_c||tid||K)$ not equal to delivered V_0in this way we can detect any modification attack. Similarly, when the server replies $M_{A_2}: \{X, R\}$to the client, and the hash value R is generated with X to check its integrity.

e) Protection Against Replay Attacks:

In the proposed scheme, an adversary will not be able to resend $M_{A_1}: \{tid, N_x, V_0\}$ because tid changes in session. Similarly, $M_{A_2}: \{X, R\}$ cannot be reused because the parameter X consists tid. Besides, the adversary cannot reuse $M_{A_3}: \{S\}$ and $M_{A_4}: \{QRCode(\lambda)\}$ since S and λ are encrypted by the one-time used public key PK_{rob}. Therefore, we ensure security against replay attacks.

f) Protection Against Compromised Client's Device or Account:

We now consider a situation when an attacker may gain access the client's device (due to the loss of the device or the device is stolen) or account of delivery app. Under this circumstance, the attacker may try to log into the delivery app as a legitimate client, and then send a request to the server. After that, the adversary may also show the QR code when the delivery robot reaches the destination. This would bring security risks and economic losses to the client. To solve this problem, two countermeasures are suggested. First, a legitimate client needs to set a strong password along with a thumb-impression for device access. Second, a two-factor security measures can be applied to deal with the circumstances when the attacker may try to gain access of the client's delivery app account (e.g. through phishing). In this case, client needs to select another password (different from

the device access one) along with OTP (based on the account access request the OTP will be sent to a legitimate device).

C. Formal Security Analysis of the Cooperative Authentication Process Using ProVerif

ProVerif [41] is a popular security protocol analysis tool. By defining parameters and events of algorithms, it proves the confidentiality of them as well as logical consequence between events. First of all, the protocol has been simplified and changed to be more appropriate for ProVerif platform, as shown in Fig. 8. It includes all details of the protocol except two changes: Firstly, we use the shared key K of the client and the server to encrypt random nonce Nc rather than original XOR operator in the scheme, because XOR is not directly supported in ProVerif, but it can be regarded as a kind of electronic codebook encryption (ECB), which allows us to replace it. Then, the checking of life duration of authentication information is also omitted, including parameters' transferring and comparison about timeout in step. Therefore, the simplified protocol also meets the requirements of authentication and has a similar structure as the original scheme and is suitable for experimental analysis.

For the predefinition of the protocol, it consists of three parts:

In Fig. 9.a, we put down channels and definition of functions. *C_cs* represents the public channel between the client and the server, and *c_rs* and *c_cr* respectively refer to the channel between the robot with the server and that between the client with the robot. Besides, *c_r* is the secure channel from the server to the robot. In the process of asymmetric encryption, we use *pk* to define a mapping from the private key to the public key, and *aenc* as encryption and *adec* as decryption. Then we compile the relationship equation adec(aenc(x, pk(y)), y)= x: By applying the public key *pk (y)* that corresponds to the private key *y* on the message *x,* we calculate the secret text. Then *x* can be restored by decryption using private key *y* on the text. Likewise, we have definition of symmetric encryption, which includes *senc* as encryption and *sdec* as decryption and *senc'*, *sdec'* for multiple parameters symmetric encryption. In addition, hash functions are widely used with different elements in the protocol, so they are individually defined as *connect3*, *connect3'* and *connect4* before being computed to a *hash* value with fixed length.

Fig. 9.a also illustrates events in the protocol and the examination on their relationships. To meet the goal of proof on no fabricated information from attackers, we focus on three parts: The first one is whether attackers forge the request of client. As shown in the graph, *event c2s_start* is the start of transferring from the client to the server, and *event c2s_end* is the acceptance of that on the server, so *query event(c2s_end)➔event(c2s_start)* means for every message received by the server, there should be at least one corresponding message

sent by the client. In this way fraud of client requirement can be excluded. Secondly, we track the flow the authentication information, which is distributed to the client and the robot by the server, and then transferred from the client to the robot. Likewise, we define each event and check those three processes. For example, *query inj-event(s2c_end) ==>inj-event(s2c_start)* indicates that for every time the client receives a message from the server, the server must has already sent it only once, and it is similar to the remaining two queries. Lastly, we check credence of the matching result, as shown in nested correspondences, to figure out whether all requested and returned messages are truly sent and received by participants in the protocol when the matching *event match_secret* happens.

In Fig. 9.a, parameters of the algorithm are predefined, and we check confidentiality of them. It is notable that the service identity *ids* and shared key *K* saved in a user can be totally different from that in another client, and elements such as *Ns*, *Tqrgen*, *UAID*, *sk* and *secretQR* can be only used once. In this part parameters are defined with sentences that begin with *free*, and *private* means these elements should not be shown to others. Next, *query attacker* is a kind of examination command about whether the element in it can be revealed to attackers in the communication. Those parameters include the server identity *ids*, shared key *K*, nonce *Ns*, serial number *UAID*, private key *sk* and information for scanning and authentication *secretQR*. What is more, *noninterf sk* is for examination on strong security, including IND-CPA. In addition, *weaksecret* function finds out the possibility of leaking if the elements *K*, *UAID* and *sk* are weak passwords.

After predefinition, we implement client, server, robot and the main function to call them repeatedly in Fig. 9.b (see [42] for detailed code). In the implementation of the client, client generates a nonce and encrypts it as Nx, and then sends tid,Nx,V0 to the server in step1. After receiving feedback from the server, which is authenticated through checking (=R) in step4, client decrypts X using *sdec'* and computes the pick-up authentication messa 16 ecretetC_send with the public key pk before post to the robot. In the implementation of the server, server accepts requests from the client and checks tid as well as V0 in step2. Next some parameters are randomly created for delivery including asymmetric keys, followed by encryption on X with shared k and hash value R for verification before returning them back to the client. As for step3, server transfers the secret key sk and serial number UAID to the robot in a private channel, and calculates the pick-up authentication message secretR to send it in a public channel. In the implementation of the robot, robot successively get messages from the server and the client. Then they are both applied asymmetric decryption with secret key sk to mat16ecretetC with secretR. Upon a successful match, robot finds the correct client. Besides, the whole process is allowed to loop for repeated requests in the main function so that leaking in statistical information can be analysed. In addition, some parameters, for example tid, are used in

multiple terminals so they are generated in the main function and loaded in components' functions in the code.

The program is successfully executed as shown in Fig. 10. It is clear, that most of queries are secure: Event relationships defined in Fig. 9.a are proved true, which means our protocol is able to figure out a forged request and feedback by attackers. Then all Query not attacker ( ) are secure so attackers cannot reveal the values. Besides, Non-interference sk is also true thus *sk* is not achieved with advanced analytical tools like IND-CPA. In addition, simple UAID and *sk* are not vulnerable, moreover they are used only once to ensure security. However, a weak choice of shared key K leads to its disclosure. ProVerif points out it occurs with a successful guess of K in the beginning rather than analysis on communication flow. In this way, we can solve the problem in the pre-exchange phase, including requirements of strong K generation as well as regular key replacement. Therefore, the proposed protocol is successfully proved to be secure in ProVerif environment. If we want to make the protocol more secure, designing in anti-DoS attacks is a great direction in the future.

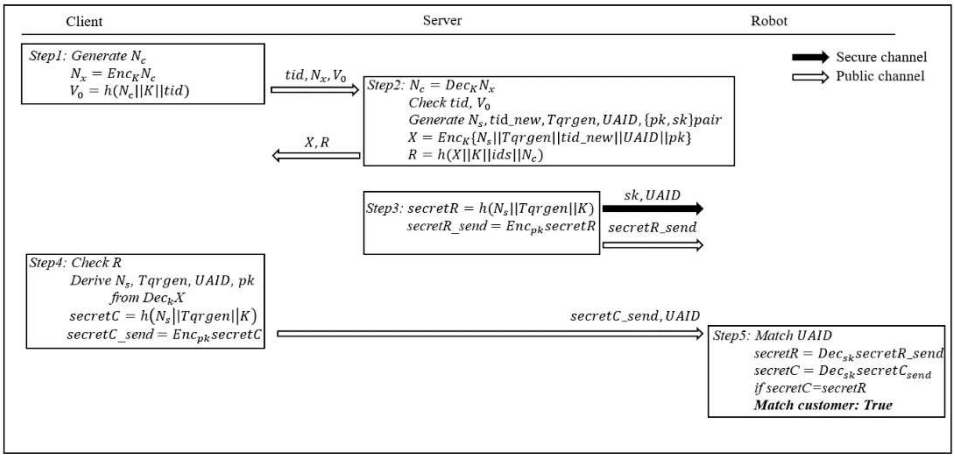


Figure 8. Simplified protocol

```
free c_cs : channel.
free c_rs : channel.
free c_cr : channel.
free c_r : channel[private].

(*asymmetric encryption*)
type pkey.
type skey.
fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y:skey; adec(aenc(x, pk(y)), y)= x.

(*symmetric encryption*)
type key.
fun senc(bitstring, key ): bitstring.
fun senc'(bitstring, bitstring, bitstring, bitstring, pkey, key ): bitstring.
reduc forall m: bitstring, K: key; sdec(senc(m, K),K)= m.
reduc forall m1: bitstring,
                m2: bitstring,
                m3: bitstring,
                m4: bitstring,
                m5: pkey,
                K: key;
        sdec'(senc'(m1,m2,m3,m4,m5,K),K)= (m1,m2,m3,m4,m5).

(*hash*)
fun connect3(bitstring,key,bitstring): bitstring.
fun connect3'(bitstring,bitstring,key): bitstring.
fun connect4(bitstring,key,bitstring,bitstring):bitstring.
fun hash(bitstring):bitstring.
```

```
(*correspondences and authentication*)
event c2s_start.
event s2c_start.
event s2r_start.
event c2r_start.
event c2s_end.
event s2c_end.
event s2r_end.
event c2r_end.
event match_secret.

query event(c2s_end)==>event(c2s_start).
query inj-event(s2c_end)==>inj-event(s2c_start).
query inj-event(s2r_end)==>inj-event(s2r_start).
query inj-event(c2r_end)==>inj-event(c2r_start).

(*nested correspondences*)
query inj-event(match_secret)
        ==> (((inj-event(c2r_end)))
        ==>((inj-event(s2c_start))&&(inj-event(c2r_start)))).
```

```
(*Reachability*)
free tid:bitstring.
free ids:bitstring[private].
free K:key[private].
free Ns:bitstring[private].
free Tqrgen:bitstring[private].
free UAID:bitstring[private].
free sk:skey[private].
free secretQR:bitstring[private].

query attacker(ids).
query attacker(K).
query attacker(Ns).
query attacker(Tqrgen).
query attacker(UAID).
query attacker(sk).
query attacker(secretQR).

(*strong secrecy*)
(*for example, IND-CPA*)
noninterf sk.

(*weak bitstrings/keys*)
weaksecret K.  (*weak K between c&s*)
weaksecret UAID.
weaksecret sk.
```

Figure 9.a. Protocol predefinition of channels & functions, events and parameters

```
(*client*)
let client()=

  (*step1*)
  new Nc;
  let Nx = senc(Nc, K) in
  let v0 = hash(Nc, K, tid) in
  event c2s_start;
  out(c_cs,(tid,Nx,v0));

  (*step4*)
  in(c_cs,(X, R));
  let (=R)=hash(X, K, ids, Nc) in
  event s2c_end;
  let (Ns, Tqrgen, tid_new, UAID, pk)=sdec'(X, K) in
  let secretC=hash(Ns, Tqrgen, K) in
  let secretQR=secretC in
  let secretC_send=aenc(secretC,pk) in
  event c2r_start;
  out(c_cr,(secretC_send, UAID)).
```

```
(*server*)
let server()=

  (*step2*)
  in(c_cs,(tid_check, Nx, v0));
  if tid=tid_check then
  let Nc=sdec(Nx, K) in
  let (=v0)=hash(Nc, K, tid) in
  event c2s_end;
  new tid_new;
  let pk= pk(sk) in
  let X=senc'(Ns,Tqrgen,tid_new,UAID,pk,K) in
  let R=hash(X, K, ids, Nc) in
  event s2c_start;
  out(c_cs,(X,R));

  (*secure channel*)
  out(c_r,(sk,UAID));

  (*step3*)
  let secretR=hash(Ns, Tqrgen, K) in
  let secretR_send=aenc((secretR),pk) in
  event s2r_start;
  out(c_rs, secretR_send).
```

```
(*robot*)
let robot()=

  (*step3*)
  in(c_r,(sk, UAID));
  in(c_rs, secretR_send);
  event s2r_end;

  (*step4*)
  in(c_cr,(secretC_send, UAID_check));
  if UAID=UAID_check then
  event c2r_end;

  (*step5*)
  let (secretR)=adec(secretR_send, sk) in
  let secretC=adec(secretC_send,sk) in
  if secretC=secretR then
  event match_secret.
```

Figure 9.b. Protocol process in terminals

```
--------------------------------------------------------------
Verification summary:
Query event(c2s_end) ==> event(c2s_start) is true.
Query inj-event(s2c_end) ==> inj-event(s2c_start) is true.
Query inj-event(s2r_end) ==> inj-event(s2r_start) is true.
Query inj-event(c2r_end) ==> inj-event(c2r_start) is true.
Query inj-event(match_secret) ==> (inj-event(c2r_end) ==> inj-event(s2c_start) && inj-event(c2r_start)) is true.
Query not attacker(ids[]) is true.
Query not attacker(K[]) is true.
Query not attacker(Ns[]) is true.
Query not attacker(Tqrgen[]) is true.
Query not attacker(UAID[]) is true.
Query not attacker(sk[]) is true.
Query not attacker(secretQR[]) is true.
Non-interference sk is true.
Weak secret K is false.
Weak secret UAID is true.
Weak secret sk is true.
--------------------------------------------------------------
```

Figure 10. Running results of Proverif analysis

V. NON-COOPERATIVE RE-IDENTIFICATION OF THE PROPOSED SCHEME

a. In this section, we introduce our non-cooperative recognition system to meet the challenge that the client is not active in presenting the QR code. This proposed scheme is a supplement of the cooperative authentication. When the task of QR code recognition failed for a maximum threshold times, the robot switches to this method to search and identify the nearby client. This part is subdivided into modelling phase and identification phase. The former modelling phase is done before the delivery request arriving in the server, including pre-processing and model training. Then, the identification phase takes place upon receiving the order, which consists of preparation of comparison, pedestrian detection and resizing image, and person re-identification. If succeeded, the robot will go toward the target client to remind him to show the QR code and then continue cooperative authentication process

A. Modelling phase

In this section, the delivery company takes photos of clients that are previously acquired during registration and completed deliveries as datasets and uses both global and local features of images for an improved Siamese Network training. Then the network model can be tested and replicated in the robot. Therefore, it is efficient considering this process is completed before orders begin. This section demonstrates the details as below:

Step M1: The proposed non-cooperative recognition scheme begins with pre-processing. We could use a person re-identification dataset or create a dataset by collecting photos of more than 100 people in four angled shots (front, back, left and right). Each folder in the dataset should represent the identity of a person and contains four images. Then, we divide the dataset into a training set (70% folders) and a testing set (30% folders).

Step M2: In model training of the proposed scheme, we apply data of the training set to an improve Siamese Network to get a trained network parameter. The structure of our Siamese Network is displayed in Fig. 11, and pseudocode of the training algorithm in Fig. 12. Then, the process of training is explained as followings:

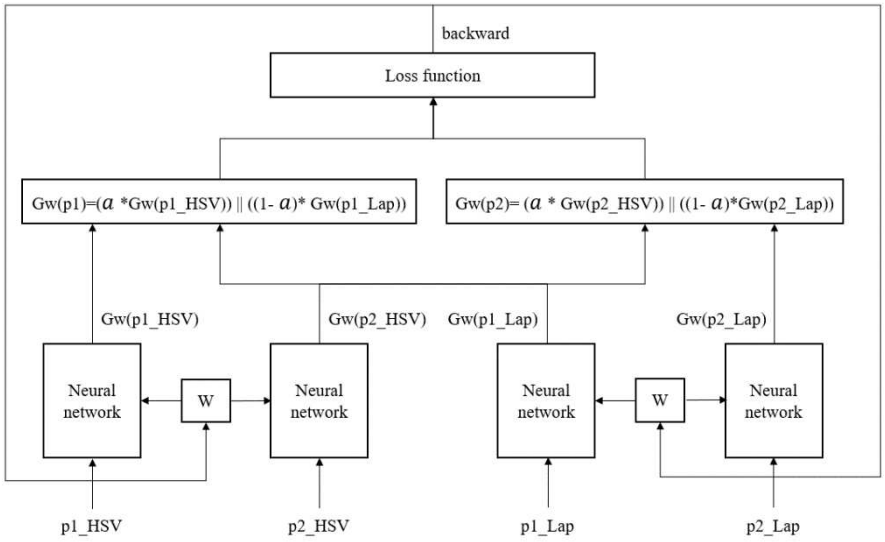


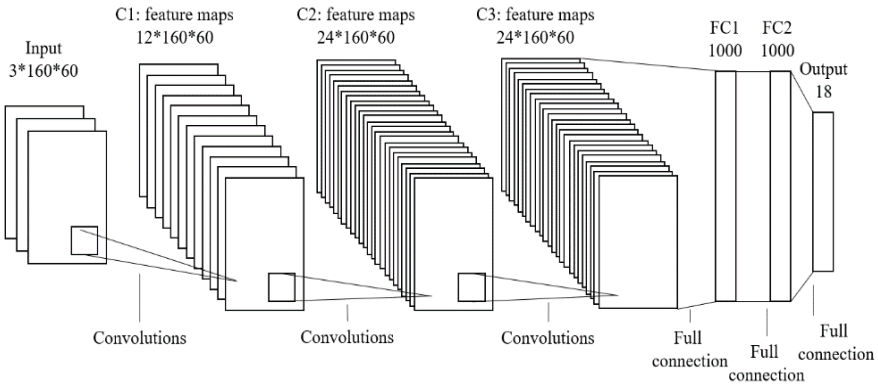
Figure 11. Improved Siamese Network Architecture

Algorithm 1. Siamese Network model training for proposed noncooperative authentication

Input: Labeled photos in training set D_{train} where each file stores images of a person, *epoch*, *batch size*, tuning parament *a* to weight HSV and Laplacian, tolerance *m*, learning rate η.
Output: Trained model *net*.

1: Initialize two-input Siamese Network *net*
2: **for** each *epoch* **do**
3: read D_{batch} from D_{train} by *batch size*
4: read a pair of photo *p1* and *p2* from the same person(set a flag $Y = 0$)or different person($Y = 1$) equally
5: convert *p1*, *p2* to HSV image *p1_HSV*, *p2_HSV* and Laplacian gradient image *p1_Lap*, *p2_Lap*
6: /*start training*/
7: *Gw(p1_HSV), Gw(p2_HSV) = net(p1_HSV, p2_HSV)*
8: *Gw(p1_Lap), Gw(p2_Lap) = net(p1_Lap, p2_Lap)*
9: *p1_output = a* Gw(p1_HSV) || (1-a)* Gw(p1_Lap)*
10: *p2_output = a* Gw(p2_HSV) || (1-a)* Gw(p2_Lap)*
11: /*loss function*/
12: $euclidean\_distance = \sqrt{(p1\_output - p2\_output)^2}$
13: $cos\_distance = \sqrt{((norm(p1\_output) - norm(p2\_output))^2}$ /*equivalent replacement*/
14: *distance = euclidean_distance * cos_distance*
15: $loss = (1-Y) * distance^2 + Y * exp(max(m - distance, 0)^2)$
16: **do** *Backpropagation(loss, η)* for weights in *net*
17: **end for**
18: **save** *net*

Figure 12. Pseudocode of the proposed training algorithm



a) **Figure 13. The neural network structur**a) **Inputs of HSV and Laplacian images**

In the proposed training scheme, the training set is divided into numeral epochs, and data in each epoch is split into many batches by settled batch size. When we start reading inputs, equally and randomly pick two pictures $p1$ and $p2$ from the same or different people and change the size of images into 160*60. Generally, they are read in the form of RGB. To efficiently recognise persons, we produce both global and local features: Convert RGB to HSV to get $p1\_HSV$ and $p2\_HSV$ as a better global feature because HSV is less sensitive to changes of luminance. Then compute Laplacian images to generate $p1\_Lap$ and $p2\_Lap$ as local features to analyse outline and texture. As a result, four inputs are generated: $p1\_HSV$, $p2\_HSV$, $p1\_Lap$ and $p2\_Lap$

b) Training the neural networks

In the proposed training scheme, four neural networks have the same weight and receive the four inputs. For each network as defined in Fig. 13, it receives an input of HSV or Laplacian image whose depth is 3. Then in three convolution layers, the depth increases from 3 to 12, and then 24. Consider that the size of image is 160*60 and the depth is 24, inputs of full connection layers is 1*24*160*60. Next, flatten the image to get the feature vector, and shorten the vector from 24*160*60 to 1000 and finally 18, and then output the vector. In this way, four vectors shaped 1*18 flow to loss function.

c) Designs of improved loss function

In the proposed training scheme, output vectors $Gw(p1\_HSV), Gw(p1\_Lap)$ or $Gw(p2\_HSV), Gw(p2\_Lap)$ from the same picture are connected by tuning parament a as weight, which is applied to adjust influences of global and local features in (EQ. 3) and (EQ. 4):

$$Gw(p1) = \big(a * Gw(p1\_HSV)\big)|||\big((1-a) * Gw(p1\_Lap)\big) \quad (\mathbf{EQ.3})$$

$$Gw(p2) = \big(a * Gw(p2\_HSV)\big)|||\big((1-a) * Gw(p2\_Lap)\big) \quad (\mathbf{EQ.4})$$

Next, combined vectors are transferred into loss functions with back propagation regression in net. As shown from line 11 to 15 in Fig. 12, we calculate two images' $euclidean\_distance$ (EQ. 5) and $cos\_distance$ (EQ. 6) by:

$$euclidean\_distance = \sqrt{(p1\_output - p2\_output)^2} \quad (\mathbf{EQ.5})$$

$$cos\_distance = \sqrt{((norm(p1\_output) - norm(p2\_output))^2} \quad (\mathbf{EQ.6})$$

Then multiple (EQ.5) and (EQ.6) to get (EQ. 7) to acquire a complex $distance$:

$$distance = euclidean\_distance * cos\_distance \quad (\mathbf{EQ.7})$$

which is the key value in the identification phase to tell whether two images are from the same person or not. In this way, the $loss$ function is defined in (EQ. 8) as:

$$loss = (1-Y) * distance^2 + Y * exp\,(max(m - distance, 0)^2) \quad (\mathbf{EQ.8})$$

When Y = 1, which means the two input images are from different people, the decrease of $loss$ indicates wider distance between them until the maximum m is reached(setting ceiling m helps avoid steep gradient in back propagation). On the contrary, when Y = 0 the pair of photos are from the same person,

and the lower $loss$ is, the smaller the distance is. Therefore, we can tell whether the person in the captured picture are the target client by calculating $distance$ of their images because the smaller distance is, the more possible they are from the same person. Next, $loss$ is applied to back propagate and update the two two-input Siamese networks net. Training is repeated in batches until all epochs are finished and then the Siamese network net is saved.

Remark 1: Reasons for choosing four inputs structure

The two inputs of a traditional two-input Siamese Network are paired RGB images from the same or two different people. However, it brings two disadvantages: For one thing, RGB images are affected by luminance in all three channels, which results in different outputs when lamination conditions change, e.g. recognition in the morning is different from that in the evening. So RGB is not suitable for our task. For another thing, RGB only represents images' global feature rather than local features, such as contour of person and texture of clothes and backpacks.

Therefore, we design a four-inputs Siamese Network. It consists of two original Siamese Network to superlatively analysis HSV images and Laplacian images. Unlike RGB, HSV use 1 channel to represent luminance and other two are for Hue and Saturation. Therefore, HSV has fewer channels influenced by lighting conditions, thus are more resistant to them. Besides, Laplacian image recodes the second-order differential feature of a picture, which shows the edge where colors change significantly. Those edges usually describe local features such as outlines, texture and equipment shape.

Remark 2: $cos\_distance$ in loss function

In the proposed algorithm, we modify the loss function by adding cosine distance and use it in the equation of computing loss. Considering the complexity of cosine calculation, a better way for it is to use equivalent normalization Euclidean distance as shown in (EQ. 14). The replacement is proved as below: Considering two points *A (x1, y1)*, *B (x2, y2)*.

Normalization vector:

$$A\left(\frac{x_1}{\sqrt{{x_1}^2+{y_1}^2}},\frac{y_1}{\sqrt{{x_1}^2+{y_1}^2}}\right), B\left(\frac{x_2}{\sqrt{{x_2}^2+{y_2}^2}},\frac{y_2}{\sqrt{{x_2}^2+{y_2}^2}}\right) \quad (\mathbf{EQ.9})$$

Cos (A, B):

$$\cos(A,B)=\frac{x_1}{\sqrt{{x_1}^2+{y_1}^2}}*\frac{x_2}{\sqrt{{x_2}^2+{y_2}^2}}+\frac{y_1}{\sqrt{{x_1}^2+{y_1}^2}}*\frac{y_2}{\sqrt{{x_2}^2+{y_2}^2}} \quad (\mathbf{EQ.10})$$

Normalized Euclidean distance:

$$euc=\sqrt{(\frac{x_1}{\sqrt{{x_1}^2+{y_1}^2}}-\frac{x_2}{\sqrt{{x_2}^2+{y_2}^2}})^2+(\frac{y_1}{\sqrt{{x_1}^2+{y_1}^2}}-\frac{y_2}{\sqrt{{x_2}^2+{y_2}^2}})^2} \quad (\mathbf{EQ.11})$$

By (EQ. 10) and (EQ. 11), normalized Euclidean distance:

$$euc = \sqrt{2 - 2 * cos(A, B)} \quad (\mathbf{EQ.12})$$

Consider definition of cosine distance:

$$cos\_distance = 1 - cos(A, B) \quad (\mathbf{EQ.13})$$

Finally, we replace $cos\_distance$ by euc^2 (coefficient omitted):

$$cos\_distance = \left(\frac{x_1}{\sqrt{{x_1}^2 + {y_1}^2}} - \frac{x_2}{\sqrt{{x_2}^2 + {y_2}^2}}\right)^2 + \left(\frac{y_1}{\sqrt{{x_1}^2 + {y_1}^2}} - \frac{y_2}{\sqrt{{x_2}^2 + {y_2}^2}}\right)^2 \quad (\mathbf{EQ.14})$$

B. Identification phase

During the identification phase, the server prepares a separate hyperplane for comparison and stores it in the robot in advance. Then, when the robot switches to the non-cooperative mode, it carries out pedestrian detection, resizes the person from the image, and then calculate the similarity distance between the client's image and the stored one to judge the presence of the target client.

Step B1: To begin with, the server loads the trained model and photos of the client. Then calculate the distance and the loss function between any two of them to select the optimized separation line for identifying the later captured image and the client's image. Any matched pictures should have a similarity distance not larger than the store value. Next, store the separation line and the client's image in the robot.

Step B2: When switches to non-cooperative mode, the robot continuously taking snapshots and detecting the presence of pedestrians. Once a person is present, then resize the captured person from image. We use standard off the shelf object detection tools to fulfil this task, and any package or function to achieve the goal of detection is acceptable.

Step B3: The trained Siamese Network model is loaded into the robot with inputs of the stored target client's image and the captured image. Next, we calculate the distance of any two images. The distance should be smaller than the pre-stored one using the pretrained separation line. When match is found, the robot moves towards the recognized client and switches back to cooperative authentication mode. Otherwise, repeat step B2 and step B3 to try to identify the client until the max mission time is reached and cancel the delivery.

VI. PERFORMANCE ANALYSIS

In this section an experiment is taken to assess the performance of the proposed system. As shown in Fig. 14, Fig. 15 and Fig. 16 we use two PCs with *ubuntu 16.04* as the client and the server, and *ROS kinetic* system [43] in a *Turtlebot3* robot [44] as the robot to build an experiment verification platform based on *python* and *ROS*. We also set the server as the master of *ROS*, and the robot as the *client*.

In cooperative mode, we use *TCP socket* to transfer data, and call some *ROS* topics to move the robot and take photos. We separately run three python programs on three terminals. Upon recipient of a request sent by the client, the server distributes encrypted messages to the client and the robot. Then robot navigates to the destination by using *Rviz*, a ROS visualisation tool, and starts to scan the QR code as shown in Fig. 17.

In the non-cooperative mode, we set an additional python3 environment, and take *CUHK01* dataset [45] for model training and testing in the server. Then load the model to compute the separate hyperplane between the client's images and others. Next, we move the trained model in the robot, and waits for the point that the system switches to non-cooperative mode. When the QR code scanning failed for five times, we change the mode, call a pedestrian detection function in *OpenCV* [46], resize the captured image by *imutils*. Then we calculate the distance of the captured photo and the image in the database, and compare it with the separate hyperplane to reidentify the client as shown in Fig. 18.

The experiment results are shown in two parts: the testing result of reidentification model and communication cost performance. The former one happens after the model training in the non-cooperative algorithm to assess its performance with comparison to other reidentification methods. And the latter records time cost of main computing process in both cooperative authentication and non-cooperative reidentification sections.

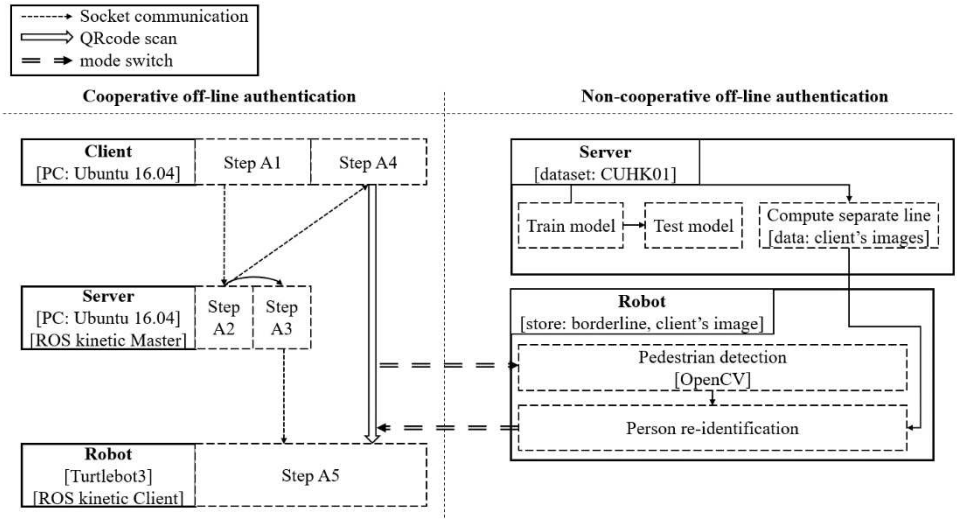


Figure 14. The proposed system platform

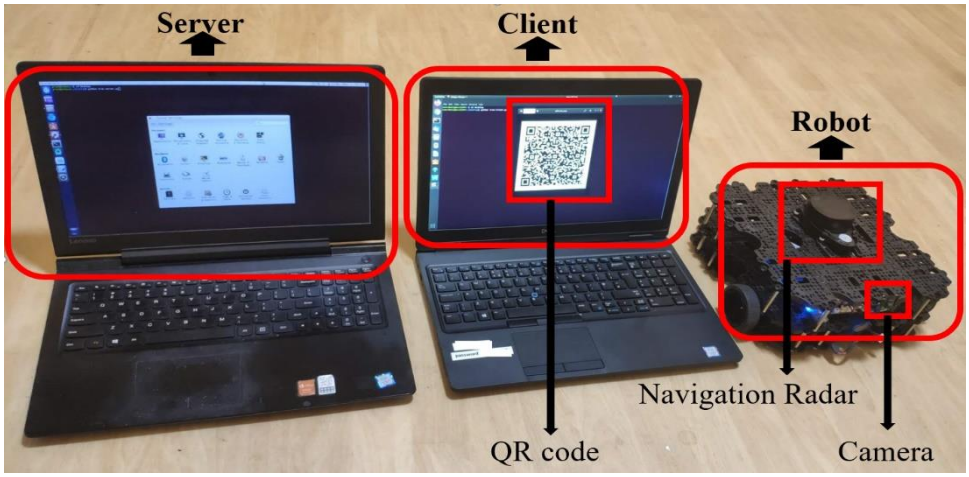


Figure 15. Our real-life experimental platform

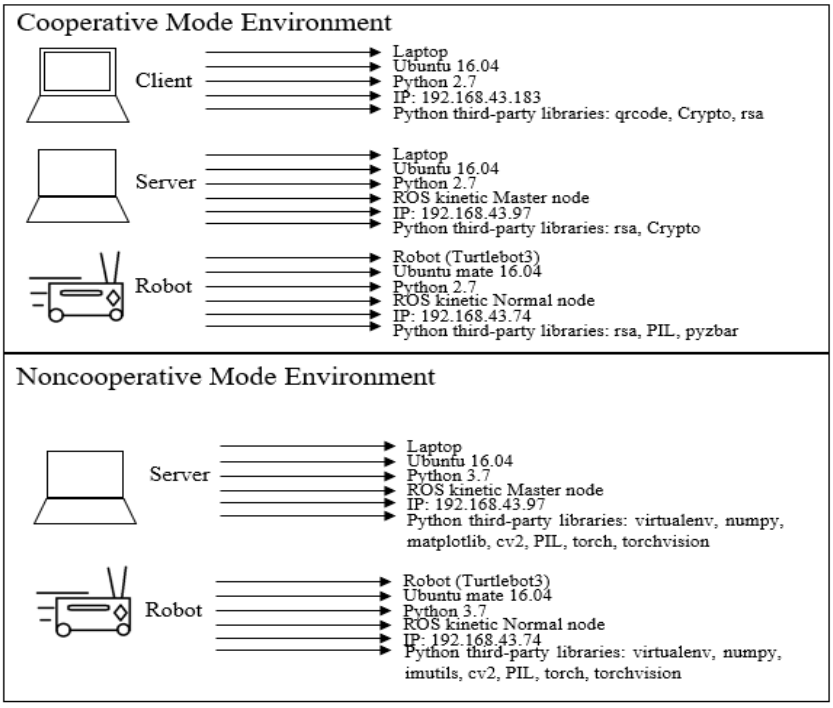


Figure 16. Implementation Environment.

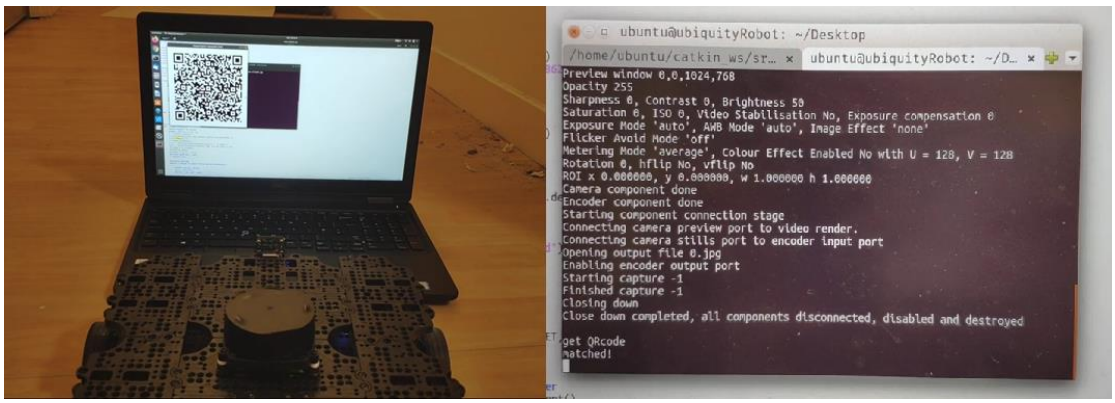


Figure 17. QR code scanning and the result

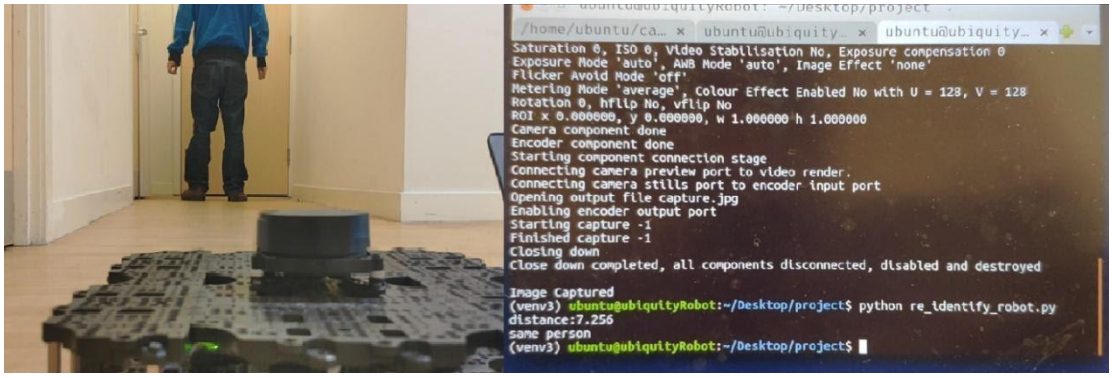


Figure 18. Person Re-ID and the result

A. Test of Re-ID model

After pre-processing in *CUHK01* and modelling training in the training dataset, we load the trained model in the testing program. Then, for each person in the testing dataset, take just one image of the client and calculate all distances between other photos and it, and rank them. In this way we derive a matrix where for each person we find the order of the closet pictures to the client. From it we draw our algorithm's CMC curve, and compare it with other state-of-the-art person reidentification algorithms, including *Filter Pairing Neural Network (FPNN)* [47], *Information-Theoretic Metric Learning (ITML)* [48], *Large Margin Nearest Neighbour (LMNN)* [49] , *Kernelized Relaxed Margin Components Analysis (KRMCA)* [50], *Logistic Discriminant-based metric learning (LDM)* [51], *Symmetry-Driven Accumulation of Local Features (SDALF)* [52], *Unsupervised Salience Learning with Salience and Dense Correspondence eSDC* [53], *Keep It Simple and Straightforward metric learning (KISSME)* [54] *and Local Maximal Occurrence and Cross-view Quadratic Discriminant Analysis (LOMO+XQDA)* [55]. As shown in Fig. 19 and Table. 3, our proposed reidentification algorithm performs better than most of other algorithms with higher Rank value, which indicates higher precision and the match assertion is more convincing with the same times of comparison. Therefore, our non-cooperative identification algorithm is suitable for recognition.

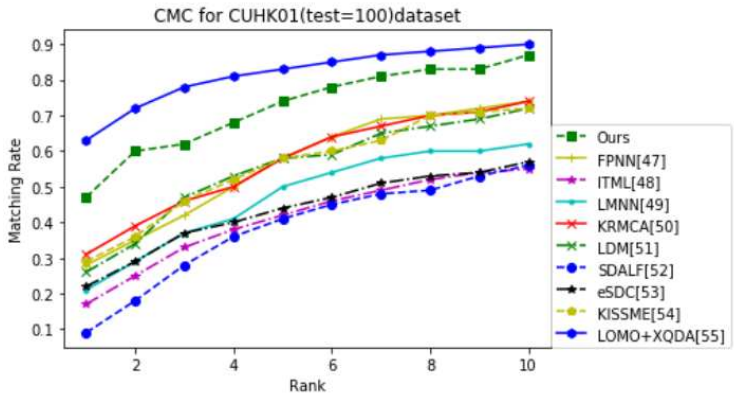


Figure 19. CUHK01 (test=100) dataset performance

Table 3. Comparison of performance on CUHK01 (test=100) dataset

Method	Rank1	Rank5	Rank10	Rank20
OURS	0.47	0.74	0.87	0.94
FPNN[47]	0.28	0.58	0.73	0.86
ITML[48]	0.17	0.42	0.55	0.71
LMNN[49]	0.21	0.49	0.62	0.78
KRMCA[50]	0.31	0.57	0.73	0.86
LDM[51]	0.26	0.57	0.72	0.84
SDALF[52]	0.09	0.41	0.56	0.66
eSDC[53]	0.22	0.43	0.57	0.69
KISSME[54]	0.29	0.57	0.72	0.86
LOMO+XQDA[55]	0.63	0.83	0.90	0.94

B. Communication cost performance

With the help of related repository of Siamese network [56], We record time cost in main computing stages of the system for analysis. For the cooperative protocol, QR code scanning, encryption, decryption and hash process are counted. For the non-cooperative recognition, we count the time for mode switch (5 times of failed scanning in this experiment), pedestrian detection and person reidentification. The result of communication cost is illustrated in Table. 4. We can observe that all cryptographic computation processes in cooperative authentication are fast. Besides, time tolerance for switching mode is long enough for the client to show the QR code, and low cost of detection and identification ensures high frequency of searching for the client. Therefore, our delivery authentication system has advantages of efficiency and high speed in running.

Table 4. Time cost for the proposed system's main stages.

Operation	Cooperative authentication	Non-Cooperative authentication
$T_{Nx\&V0}$ (step A1)	6.65ms	-
$T_{X\&R}$ (step A2)	10.38ms	-
T_{Δ} (hash in step A3)	5.42ms	-
T_{S} (public encryption in step A3)	25.64ms	-
T_{λ} (public encryption in step A4)	26.16ms	-
T_{Scan} (QRcode scanning in step A5)	3.26s	-
$T_{Decrypt\&match}$ (step A5)	52.93ms	-
T_{Switch} (mode switch)	-	14.5s

T_{Line} (computing separate line)	-	317.24ms
T_{Detect} (pedestrian detection & cut)	-	189.45ms
T_{Reid} (Person re-identification)	-	228.92ms
Total	3.26s+127.18ms	735.61ms (except switch)

VII. CONCLUSION

In this paper, we designed and implemented a robotic delivery authentication system, which includes the client, the server and the robot. A cooperative authentication protocol has been proposed and the security of the system has been formally analysed using ProVerif. The abilities of our proposed system to mitigate various threats has been verified in a home-delivery scenario. In order to deal with cases where the clients are not active in presenting their QR code, we designed the non-cooperative mode to reidentify and recognize the client. Experiments have been designed out to compare our algorithms with other state of the art ones, and experimental results show that our non-cooperative recognition design is more accurate than most of them, which helps to find the client. Furthermore, the result of time cost in the experiment demonstrates that our lightweight system calculates quickly and identifies pedestrians efficiently. In conclusion, our proposed system is secure, accurate, fast and efficient.

References

1. eMarketer., (2020). US Ecommerce Will Rise 18% in 2020 amid the Pandemic. [online]. *eMarketer*. [Viewed 17 August 2020]. Available from: https://www.emarketer.com/content/us-ecommerce-will-rise-18-2020-amid-pandemic
2. Clement, J., (2020). Amazon: Global Net Revenue By Product 2019. [online]. *Statista*. [Viewed 17 August 2020]. Available at: https://www.statista.com/statistics/672747/amazons-consolidated-net-revenue-by-segment
3. Clement, J., (2020). Fulfilment by Amazon Usage Among Top Sellers Worldwide 2017-2018. [online]. *Statista*. [Viewed 17 August 2020]. Available at: https://www.statista.com/statistics/1020046/global-fba-usage-top-amazon-sellers
4. eMarketer., (2020). China Ecommerce 2020. [online]. *eMarketer*. [Viewed 17 August 2020]. Available at: https://www.emarketer.com/content/china-ecommerce-2020
5. FedEx., (2018). 2017 Investment + Integration + Innovation. [Online]. *FedEx*. [Viewed 17 August 2020]. Available at: https://investors.fedex.com/financial-information/annual-reports/default.aspx

6. Davis, D., (2020). Amazon's Q2 North America Revenue Surges 43% As Web Grocery Sales Triple. [online]. *Digital Commerce 360*. [Viewed 17 August 2020]. Available at: https://www.digitalcommerce360.com/article/amazon-sales
7. The Manufacturer., (2020). Delivering Industry 4.0: The Hype And Challenges. [online]. *The Manufacturer*. [Viewed 17 August 2020]. Available at: https://www.themanufacturer.com/articles/delivering-industry-4-0-the-hype-and-challenges
8. Lardinois, F., (2019). A first look at Amazon's new delivery drone [online]. *Extra Crunch*. [Viewed 17 August 2020]. Available at: https://techcrunch.com/2019/06/05/a-first-look-at-amazons-new-delivery-drone
9. BURGESS, M ., (2016) Amazon has made its first Prime Air drone delivery in the UK [online]. *Wired UK*. [Viewed 17 August 2020]. Available at: https://www.wired.co.uk/article/amazon-first-drone-delivery-uk
10. Dormehl, L., (2020). When it comes to delivery drones, Google's Wing is miles above the competition. [online]. *Digital Trends*. [Viewed 17 August 2020]. Available at: https://www.digitaltrends.com/cool-tech/google-wing-drone-deliveries/
11. Etherington, D., (2020). UPS partners with Wingcopter to develop new multipurpose drone delivery fleet. [online]. *Extra Crunch*. [Viewed 17 August 2020]. Available at: https://techcrunch.com/2020/03/24/ups-partners-with-wingcopter-to-develop-new-multipurpose-drone-delivery-fleet/
12. DOUGLAS, A., (2019). DHL launches first regular fully-automated urban drone delivery service with EHang. [online]. *Commercial drone professional*. [Viewed 17 August 2020]. Available at: https://www.commercialdroneprofessional.com/dhl-launches-first-regular-fully-automated-urban-drone-delivery-service-with-ehang/
13. Garrett-Glaser, B., (2019). Flytrex Brings Drone Delivery to North Carolina, Eyes Expansion in Reykjavik. [online]. *Aviation Today*. [Viewed 17 August 2020]. Available at: https://www.aviationtoday.com/2019/08/21/flytrex-brings-drone-delivery-north-carolina-eyes-expansion-reykjavik/
14. Fannin, R., (2020). The Rush to Deploy Robots in China Amid the Coronavirus Outbreak. [online]. *CNBC International*. [Viewed 17 August 2020]. Available at: https://www.cnbc.com/2020/03/02/the-rush-to-deploy-robots-in-china-amid-the-coronavirus-outbreak.html
15. P. Gope et al. "A Provably Secure Authentication Scheme for RFID-enabled UAV Applications." to appear in Computer and Communication. 2020.
16. P. Gope, and B. Sikdar, "An Efficient Privacy-Preserving Authenticated Key Agreement Scheme for Edge-Assisted Internet of Drones," IEEE Transactions on Vehicular Technology, DOI: 10.1109/TVT.2020.301877, 2020.

17. P. Gope et al. ``A secure IoT-based modern healthcare system with fault-tolerant decision-making process,'' IEEE Journal of Biomedical and Health Informatics, DOI: 10.1109/JBHI.2020.3007488, 2020
18. P. Gope, O. Millwood, N Saxena, A provably secure authentication scheme for RFID-enabled UAV applications. Computer Communications, 166, 19-25, 2021.
19. "Announcing Approval of Federal Information Processing Standard (FIPS) Publication 180-3, Secure Hash Standard, a Revision of FIPS 180-2, Secure Hash Standard." *The Federal Register / FIND* 73.202 (2008): 61783. Web.
20. P. Gope, and B. Sikdar, "An Efficient Privacy-Preserving Authenticated Key Agreement Scheme for Edge-Assisted Internet of Drones," IEEE Transactions on Vehicular Technology, DOI: 10.1109/TVT.2020.301877, 2020
21. Matthew, H. "An Introduction to QR Codes: Linking Libraries and Mobile Patrons." *Medical Reference Services Quarterly* 30.3 (2011): 295-300. Web.
22. Peng, K, Sanabria, H, Wu, D, and Zhu, C. "Security overview of QR codes." *Student project in the MIT course* 6.14 (2014). Web.
23. Zhong, Z, Zheng, L, Cao, D, and Li, S. "Re-ranking person re-identification with k-reciprocal encoding." *IEEE Conference on Computer Vision and Pattern Recognition* (2017): 1318-1327. Web
24. Zheng, L, Yang, Y, and Hauptmann, A. "Person Re-identification: Past, Present and Future." (2016). Web.
25. Ye, M, Shen, J, Lin, G. Xiang, T. Shao, L, and Hoi, S. "Deep Learning for Person Re-identification: A Survey and Outlook." (2020). Web.
26. Cao, J, Ding, Y, and Li, X. "Person Reidentification Based on View Information and Batch Feature Erasing." *Journal of Electronic Imaging* 29.4 (2020): 1. Web.
27. Lin, Y, Zheng, L, Zheng, Z, Wu, Y, Hu, Z., Yan, C., and Yang, Y. "Improving person re-identification by attribute and identity learning." *Pattern Recognition* 95 (2019): 151-161. Web.
28. Varior, R, Haloi, M, and Wang, G. "Gated Siamese Convolutional Neural Network Architecture for Human Re-identification." 9912 (2016): 791-808. Web.
29. Schroff, F, Kalenichenko, D, and Philbin, J. "FaceNet: A Unified Embedding for Face Recognition and Clustering." (2015). Web.
30. Liu, H, Feng, J, Qi, M, Jiang, J, and Yan, S. "End-to-End Comparative Attention Networks for Person Re-Identification." *IEEE Transactions on Image Processing* 26.7 (2017): 3492-506. Web.
31. Cheng, D, Gong, Y, Zhou, S, Wang, J, and Zheng, N. "Person re-identification by multi-channel parts-based cnn with improved triplet loss function." *IEEE conference on computer vision and pattern recognition* (2016): 1335-1344. Web.

32. Chen, W, Chen, X, Zhang, J, and Huang, K. "Beyond Triplet Loss: A Deep Quadruplet Network for Person Re-identification." (2017). Web.
33. Xiao, Q, Luo, H, and Zhang, C. "Margin Sample Mining Loss: A Deep Learning Based Method for Person Re-identification." (2017). Web.
34. Yu, H, Wu, A, and Zheng, W. "Unsupervised Person Re-Identification by Deep Asymmetric Metric Embedding." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2020): 956-73. Web.
35. Bolle, R.M, Connell, J.H, Pankanti, S, Ratha, N.K, and Senior, A.W. "The Relation between the ROC Curve and the CMC." 2005 (2005): 15-20. Web.
36. Bromley, J, Guyon, I, LeCun, Y, Säckinger, E, and Shah, R. "Signature verification using a siamese time delay neural network." *Advances in neural information processing systems* (1994): 737-744. Web.
37. Chopra, S, Hadsell, R, and LeCun, Y. "Learning a Similarity Metric Discriminatively, with Application to Face Verification." 1 (2005): 539-46 Vol. 1. Web.
38. Reinhard, E, Adhikhmin, M, Gooch, B, and Shirley, P. "Colour transfer between images." *IEEE Computer graphics and applications* 21.5 (2001): 34-41. Web.
39. Shubin, M.A., (2020). Laplace Operator. [online]. *Encyclopedia Of Mathematics*. [Viewed 17 August 2020]. Available at: http://encyclopediaofmath.org/index.php?title=Laplace_operator&oldid=47581
40. Dolev, D, and Yao, A. "On the Security of Public Key Protocols." *IEEE Transactions on Information Theory* 29.2 (1983): 198-208. Web.
41. Blanchet, B., Cheval, V., Allamigeon, X., Smyth, B. and Sylvestre, M., (2020). *Proverif*. [Viewed 17 August 2020]. Available from: https://prosecco.gforge.inria.fr/personal/bblanche/proverif
42. Source-code-for-Proverif-Implémentation. https://github.com/Jiapei-Yang/Secure-Robotic-Shipping/blob/master/Proverif/verifier.pv
43. TullyFoote., (2018). ROS Kinetic installation instructions. [online]. *Open robotics*. [Viewed 17 August 2020]. Available from: http://wiki.ros.org/kinetic/Installation
44. ROBOTIS., (2020). ROBOTIS E-Manual TURTLEBOT3. [online]. *ROBOTIS*. [Viewed 17 August 2020]. Available at: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview
45. Li, W, Zhao, R, and Wang, X. "Human Reidentification with Transferred Metric Learning." *Asian conference on computer vision* (2012): 31-44. Web.
46. OpenCV team., (2020). OpenCV. [online]. *OpenCV team*. [Viewed 17 August 2020]. Available at: https://opencv.org
47. Li, W, Zhao, R, Xiao, T, and Wang, X. "Deepreid: Deep filter pairing neural network for person re-identification". *IEEE conference on computer vision and pattern recognition* (2014): 152-159. Web.

48. Davis, J. V, Kulis, B, Jain, P, Sra, S, and Dhillon, I. S. "Information-theoretic metric learning." *24th international conference on Machine learning* (2007): 209-216. Web.
49. Hirzer, M, Roth, P. M, and Bischof, H. "Person re-identification by efficient impostor-based metric learning." *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (2012): 203-208. Web.
50. Liu, H, Qi, M, and Jiang, J. "Kernelized relaxed margin components analysis for person re-identification." *IEEE Signal Processing Letters* 22.7 (2014): 910-914. Web.
51. Guillaumin, M, Verbeek, J, and Schmid, C. "Is that you? Metric learning approaches for face identification." *IEEE international conference on computer vision* (2009): 498-505. Web.
52. Farenzena, M, Bazzani, L, Perina, A, Murino, V, and Cristani, M. "Person re-identification by symmetry-driven accumulation of local features." *IEEE computer society conference on computer vision and pattern recognition* (2010): 2360-2367. Web.
53. Zhao, R, Ouyang, W, and Wang, X. "Unsupervised salience learning for person re-identification." *IEEE conference on computer vision and pattern recognition* (2013): 3586-3593. Web.
54. Kostinger, M, Hirzer, M, Wohlhart, P, Roth, P. M, and Bischof, H. "Large Scale Metric Learning from Equivalence Constraints." (2012): 2288-295. Web.
55. Liao, S, Hu, Y, Zhu, X, and Li, S. Z. "Person re-identification by local maximal occurrence representation and metric learning." *IEEE conference on computer vision and pattern recognition* (2015).: 2197-2206. Web.
56. Gupta, H (2018), Facial-Similarity-with-Siamese-Networks-in-Pytorch [Source code]. https://github.com/harveyslash/Facial-Similarity-with-Siamese-Networks-in-Pytorch
57. D. Xue, X. Wang, J. Zhu, D.N.Davis, B. Wang, W. Zhao, Y. Peng, and Y. Cheng, "An adaptive ensemble approach to ambient intelligence assisted people search." Applied System Innovation, 2018: 1(3), p. 33