Secure Matching of Dutch Car License Plates

Archana Bindu Sunil*‡§, Zekeriya Erkin†, and Thijs Veugen†‡
*Vrije Universiteit Amsterdam, The Netherlands
†Cyber Security Group, Delft University of Technology, The Netherlands
‡TNO Den Haag, The Netherlands §Amrita University, India

***Abstract*—License plate matching plays an important role in applications like law enforcement, traffic management and road pricing, where the plate is first recognized and then compared to a database of authorized vehicle registration plates. Unfortunately, there are several privacy related issues that should be taken care of before deploying plate recognition systems. As a scientific solution to privacy concerns, we propose a simple and accurate character recognition scheme combined with an integer matching scheme that is designed to work with encrypted license plates. Our analysis and experimental results show that the deployment of such a system can be deemed possible.**

***Index Terms*—License plate matching, character recognition, secure signal processing, cryptography, homomorphic encryption.**

I. Introduction

License plates are the identity of an automobile which helps in distinguishing one from another and hence license plate recognition plays an important role in systems like traffic management, road pricing, security and crime investigation [1]. There is an increasing use of license plate recognition techniques by law enforcement agencies worldwide for tracking vehicles of interest like stolen and wanted vehicles. We see that a license plate itself may not be private, but the use of the collected data can reveal sensitive information about people and vehicles: any vehicle and its owner can be identified and tracked easily. Hence, the security of this system and the privacy have to be guaranteed for the plate detection system to be used in practice.

In this paper, we focus on the privacy aspect of the license plate detection systems. We propose to use “secure signal processing” that relies on cryptographic tools for the privacy protection [2]. This approach relies on processing encrypted data by deploying homomorphic encryption schemes, which makes it possible to perform signal processing algorithms on encrypted data without leaking sensitive information in an untrusted environment. The result of the algorithm is only known to the owner of data.

The fingerprint-based authentication protocol presented in [3] uses homomorphic encryption for identification of matching identities. A non-interactive fully-private outsourced face verification system, working with encrypted feature vectors and using fully homomorphic encryption, is presented in [4]. Recommendations generated by encrypting private data and processing them, and a comparison protocol for encrypted and packed data, are discussed in [5], where the active participation of the user is eliminated by introducing a semi-trusted third party.

To the best of our knowledge, secure plate detection has not been investigated before. The license plate matching technique presented in [6] deals with matching plate readings captured by a dual setup of license plate readers, and the matching utilizes an edit distance technique of text mining, which measures the closeness of two strings. In [7], the integration of license plate and face recognition for security during entry and exit in parking areas is evaluated, but the matching is performed between Peak to Side lobe Ratio (PSR) values of a decrypted plate/face image from the database (as they are stored in the database after encrypting using Hill Cipher matrix manipulation) and a plain plate/face image.

The focus of this work is to develop a system for securely matching license plates such that the license plate and the information associated with it are protected. In this paper we investigate scenarios involving multiple parties, for exact matching of car license plate numbers. Consider a scenario which involves an office parking lot that has the database of employee and visitor vehicle license plates, and a security agency with a license plate of a potential criminal’s vehicle that they are trying to track. The security agency wants to find whether that particular vehicle has entered the parking lot. Both entities would like to keep their data hidden from each other, yet perform the required processing to obtain the results. The license plate value, whose existence in the database is to be checked, is encrypted and sent to the office to do the matching. In such a setting, the office does not learn any information about the license plate it receives from the agency, as it is in encrypted format. The agency might get to know whether the vehicle has entered the office parking lot or not but no details about the other license plates in the database should leak. Thus, the information belonging to both entities is secured.

This work has multiple contributions. Firstly we present a character recognition system which could be easily translated to the encrypted domain. Secondly, we compare the proposed character recognition algorithm with some of the state-of-the-art techniques, and prove that it has high success rates, and is comparable to the currently used techniques. Thirdly, we design secure protocols for exact matching of license plates for three different scenarios using two different cryptographic tools, namely additively homomorphic and fully homomorphic encryption schemes. Lastly, we implement the protocols and analyze the results, and their performances are compared for license plate databases of twenty-one different sizes. Our analyzes shows that the proposed cryptographic protocols have high performance and are promising to be deployed in practice.

II. Preliminaries

The primary operations involved in a character recognition system are thresholding, vertical projection histogram ([8], [9]) and feature extraction ([9], [10]). Discretization of the colors in an image into a number of bins, and then counting the number of pixels in each bin, forms a color histogram; in a vertical projection histogram, the columns form the bins instead of the color intensity. In feature extraction, the input image is transformed to a reduced set of features that contain the relevant information from the input, and the desired processing operations are performed by using this reduced representation.

The important cryptographic algorithms that are used for designing the secure data matching systems are Paillier [11] and a version of Gentry's fully homomorphic cryptosystem [4]. These are algorithms for public key or asymmetric cryptography, which requires two separate mathematically linked keys: a private and a public key. Paillier encryption is a probabilistic algorithm, whose security is based on the difficulty of computing n^{th} residue classes, whereas the extended Gentry's system is Goldreich-Goldwasser-Halevi (GGH)-type based on ideal lattices. For a detailed description of the fully homomorphic system, we refer the readers to [12] and [4].

III. License Plate Matching

The secure license plate matching system comprises of two main phases: the extraction of characters from the plate, and the exact matching of a license plate with a database of license plates.

A. *Extraction of license plate characters*

The input license plate image is subject to thresholding and border removal. The characters are segmented from the thresholded image by using vertical projection histogram method. Using this method, the areas containing continuous black pixels are detected and those areas are cropped out. The cropped out areas, which correspond to character images, are converted to one-dimensional arrays, both height-wise (y-direction) and width-wise (x-direction). These one-dimensional arrays are then converted to an array of blocks, where each block holds the sum of pixel intensities of the pixels in that block. The array of blocks is then converted to a feature array which consists of $+1$'s and -1's. Feature array value at position $i = \begin{cases} +1, & \text{if block}[i+1] < \text{block}[i]; \\ -1, & \text{otherwise.} \end{cases}$

This feature array is compared to the standard feature arrays, obtained using the same method, for each character in a standard character set. The character of the standard set yielding the minimum difference is predicted as the correct character. To make the algorithm robust, predictions are made for different block sizes, and the most common or most likely character is considered as the desired character.

Let the feature of license plate characters be represented by $P = p_1, p_2, \ldots p_b$, and the feature for the standard array be represented by $Q = q_1, q_2, \ldots q_b$, where b represents the number of blocks into which the one-dimensional array is divided. The difference of features P and Q, denoted by diff, is measured as

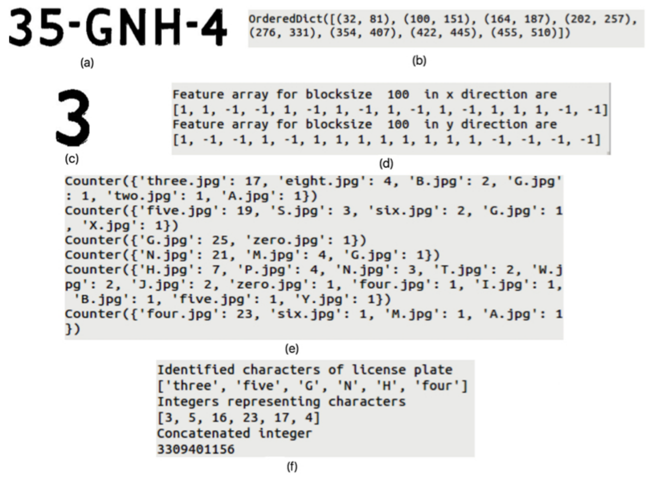
$$\text{diff} = \sum_{i=1}^{b} (p_i - q_i)^2 \tag{1}$$

where the p_i's and q_i's are either $+1$ or -1.

The above mentioned steps of the algorithm are applied for each of the segmented characters yielding a list of characters corresponding to the characters in the license plate. The character list thus obtained is first converted to integers by mapping the characters A to Z to integers 10 to 35. Then the six integers representing the license plate characters are concatenated to form an integer. Each integer value is between 0 and 35, and hence requires 6 bits for representing. Let $a_5, a_4, a_3, a_2, a_1, a_0$ be the integers, then a is the concatenated integer.

$$a = \sum_{i=0}^{5} a_i \cdot 2^{6 \cdot i} \tag{2}$$

Fig.1 shows the different steps involved in extraction of characters for the dutch license plate "35-GNH-4".



```
OrderedDict([(32, 81), (100, 151), (164, 187), (202, 257),
(276, 331), (354, 407), (422, 445), (455, 510)])
```

```
Feature array for blocksize  100  in x direction are
[1, 1, -1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 1, 1, -1, -1]
Feature array for blocksize  100  in y direction are
[1, -1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1]
```

```
Counter({'three.jpg': 17, 'eight.jpg': 4, 'B.jpg': 2, 'G.jpg'
: 1, 'two.jpg': 1, 'A.jpg': 1})
Counter({'five.jpg': 19, 'S.jpg': 3, 'six.jpg': 2, 'G.jpg': 1
, 'X.jpg': 1})
Counter({'G.jpg': 25, 'zero.jpg': 1})
Counter({'N.jpg': 21, 'M.jpg': 4, 'G.jpg': 1})
Counter({'H.jpg': 7, 'P.jpg': 4, 'N.jpg': 3, 'T.jpg': 2, 'W.j
pg': 2, 'J.jpg': 2, 'zero.jpg': 1, 'four.jpg': 1, 'I.jpg': 1,
 'B.jpg': 1, 'five.jpg': 1, 'Y.jpg': 1})
Counter({'four.jpg': 23, 'six.jpg': 1, 'M.jpg': 1, 'A.jpg': 1
})
```

```
Identified characters of license plate
['three', 'five', 'G', 'N', 'H', 'four']
Integers representing characters
[3, 5, 16, 23, 17, 4]
Concatenated integer
3309401156
```

Fig. 1. (a) License plate image after thresholding, (b) Ranges of continuous black pixels in (a), (c) Segmented character in range (32,81) of (b), (d) Feature array of character '3' for block-size 100 in x and y direction, (e) Predicted characters for the license plate in (a), (f) Integers representing the alphabets and the concatenated integer representing the characters of the license plate in (a)

B. *Matching of the license plate values*

This is performed in the encrypted domain, and involves operations on integers. The license plate for which the match has to be found is referred to as the test plate, and is matched against a database of license plates. The license plate characters are represented as integers and for matching, the difference between the integer corresponding to the test plate and each entry in the database (which is also an integer obtained using same method) is calculated. If there is a zero among the differences, it implies that the test plate value has a matching entry in the database.

The integer comparison protocol described in [13] is a bit-wise operation, performing computations on encrypted bits of two values. The protocol for equality testing described here finds the difference between two encrypted numbers and is a more suitable approach for this scenario as we need to find the difference of one value to a set of values in a database.

A	entity with the test license plate
x	integer representing the test license plate characters
B	entity with a database of license plates
s	number of plates in the database
y_i	integers representing license plate characters for i = 0...s
k	number of bits representing the license plate characters
T	trusted third party
pk	public key of cryptosystem
sk	secret or private key of cryptosystem
$\sum$	sum of the elements
$\prod$	product of the elements
$[[var]]$	variable var encrypted using Paillier encryption
$[var]$	variable var encrypted using Gentry's extended fully homomorphic system

TABLE I
NOTATIONS

Three scenarios are considered here for secure matching of license plates using homomorphic encryption. We assume that all the entities involved in these scenarios are honest.

1) Scenario 1: In this scenario two entities **A** and **B** are involved. **A** has the public and private keys and a test plate, and wants to find whether this plate is present in the database, which is in possession of **B**. The exact matching operation is performed at **B** but the result, whether there is a match or not, is known by **A** only. Here the database at **B** is stored in plaintext format, as the encryption depends on the keys, which are possessed by **A**, and hence **B** cannot be encrypted and stored.

A can be a police agency whereas **B** could be the security agency for an organization who keeps track of all the vehicles entering and leaving the office premises.

The two protocols for secure license plate matching in Scenario 1, i.e. using Paillier encryption and Gentry's extended fully homomorphic cryptosystem, are described here.

Protocol 1 - Paillier encryption:

1) **A** sends pk and $[[x]]$, the latter encrypted using pk to **B**.
2) The difference between the test plate integer $[[x]]$ and each element $[[y_i]]$ at **B** is computed by **B** in the encrypted domain as

$$[[d_i]] = [[x - y_i]] = [[x]] \cdot [[y_i]]^{-1}. \tag{3}$$

3) At **B**, the differences are blinded so that when the results are sent to **A**, **A** does not learn any information about the data at **B** other than if there is a match or not:

$$[[\hat{d_i}]] = [[d_i \cdot R]] = [[d_i]]^R, \tag{4}$$

where R is a random integer. The difference value being zero implies that there is a match and this value will remain unaltered when multiplied by R. For other difference values, multiplication by R yields a random value from which no information about the database entries can be inferred by **A**.
4) For reducing communication costs, the $[[\hat{d_i}]]$ values are packed as follows before being sent back to **A**:

$$[[\hat{d_p}]] = [[\sum_{i=0}^{s-1} \hat{d_i} \cdot 2^{k \cdot i}]] = \prod_{i=0}^{s-1} [[\hat{d_i}]]^{2^{k \cdot i}}, \tag{5}$$

where k is the number of bits representing the license plate characters, and s represents the number of license plates in the database.
5) The differences are randomized (before packing) so that if there is a match, one cannot predict which entry of the database was the match, which would otherwise leak some information about the position of the test plate entry in the database. The randomization is performed by taking database entries in scattered order while packing (i.e. the$i$'s in Equation 5 are permuted).
6) The results are decrypted at **A** using sk and if there is a zero among the decrypted values, then **A** can conclude that there was a match for the test plate in the database.

$(\mathrm{mod} n^2)$ is not mentioned at each formula.

Protocol 2 - Gentry's extended fully homomorphic cryptosystem: The steps are the same as in Protocol 1 but the corresponding operations using Gentry's extended fully homomorphic cryptosystem are:

The difference calculation at **B** (similar to Equation 3)

$$[d_i] = [x] - [y_i], \tag{6}$$

the blinding of differences at **B** (similar to Equation 4)

$$[\hat{d_i}] = [R \cdot d_i], \tag{7}$$

where R is a random integer, and packing of the blinded difference values at **B** (similar to Equation 5)

$$[\hat{d_p}] = \sum_{i=0}^{s-1} [\hat{d_i}] \cdot 2^{k \cdot i}, \tag{8}$$

where k is the number of bits representing the license plate characters, and s represents the number of license plates.

2) Scenario 2: Two entities **A** and **B** are involved. **B** has the public and private keys, and sends the public key and the y_i's in encrypted form to **A**. The difference calculation is carried out in encrypted domain at **A**, and the encrypted results are sent back to **B**. **B** decrypts the differences, and finds if there is a match or not, and sends this to **A**. Here the database at **B** can be stored in encrypted format but the result is known to both entities. This is useful if the database has to be stored in encrypted form, preventing information leakage from the database. It is also reasonable to allow the computation to be performed at

A, as it is **A** who wants to find whether there is a match or not.

3) Scenario 3: Three entities **A**, **B** and **T** are involved. The public and private keys are at **T**, and the public key is sent to **A** and **B**. **A** sends x in encrypted form to **B**. The difference calculation is performed at **B** for each y_i and the results are sent to **T** who has the private key to decrypt the result. **T** informs **A** whether a match is found or not. Here the database is stored in encrypted format and the result is known to **A** and **T**. The high communication costs of Scenario 2 are compensated with the involvement of a trusted third party.

A can be the police agency, whereas **B** could be the security agency to whom an organization has outsourced its security related works. **T** could be the organization who is trusted by both the security agency and the police.

The protocols using Paillier encryption and Gentry's extended fully homomorphic cryptosystem for Scenarios 2 and 3 are not described in detail due to space constraints.

IV. Results

The implementation [14] consists of two sections: a character recognition algorithm and a secure data matching algorithm. The character recognition algorithm was developed, in spite of numerous techniques already being used for this purpose, such that it is easily realizable in the encrypted domain.

A. Character recognition

The character recognition algorithm was implemented in Python using OpenCV library. Dutch car license plates consisting of a sequence of six characters composed of alphabets and digits in a single line are considered. The current system uses black letters on a yellow background and the font is assumed to be Gill Sans, one of the commonly used license plate fonts.

Each character of the plate is predicted twenty-six times, as there are thirteen block sizes for x-direction and y-direction each, and the most commonly predicted character is considered correct. The block sizes, used for prediction, are 5, 10, 15, 20, 25, 30, 40, 45, 50, 60, 75, 90 and 100.

The character recognition scheme works with an overall accuracy of 98.5%. Out of the 528 characters tested, 520 were identified correctly. The characters B and H were the most mistaken characters as per the analysis of the results. But even if one of the characters of the license plate was wrongly identified, the recognition of the license plate, as a whole, was considered wrong. From the 88 license plates tested, 81 were identified correctly, yielding an accuracy of 92% for the recognition of license plates. Table II shows that our algorithm has rates comparable to many of the state-of-the-art techniques used for character recognition. The method described in [15] has success rates closest to our implementation but it uses an open source optical character recognition (OCR) engine, ABBYY. In spite of techniques existing for character identification, this was developed so as to avoid the use of any classifiers and non-linear methods. The proposed algorithm uses mainly arithmetic operations and integer comparisons to recognize characters. This makes it possible to be implemented in encrypted domain using homomorphic encryption, if needed, even though our implementation is in plain domain.

Reference	Method	Success rate (%)
[10]	Statistical feature extraction	85%
[16]	Template matching	90.3%
[9]	SVM Integration with feature extraction	93.7%
[17]	Hierarchical Neural Network(HNN)	95.2%
[16]	Genetic algorithm	96.8%
[15]	ABBYY OCR software	98.7%
Proposed	**Feature extraction and minimum difference**	**98.5%**

TABLE II
Success rates of different character recognition algorithms

B. Secure data matching

We implemented the Paillier encryption for integer matching in Python and C++. In C++, the Secure Computation Library (SeComLib) was used for realizing cryptographic schemes in the encrypted domain. The time taken for Paillier key generation in C++ was about 0.4 seconds, and for one encryption it took about 5 microseconds. The extended version of Gentry's cryptosystem as described in [4] was implemented in C++ for data matching purposes. The time required for key generation in this system was about 1 second, and it took about 1 millisecond for encrypting one integer. In both cases, the keys were generated with a key size of 1024 bits. The concatenated integer was encrypted using the public key, and the results after computing the difference were decrypted using the private key.

The comparison of execution time for secure matching in Scenario 1 using Paillier encryption and extended Gentry's system in C++, are represented as a graph of the execution time vs. database sizes in Fig.2. The execution time depends on the size of the database: for Paillier encryption it is between 0.12 to 6.75 seconds, whereas for Gentry's extended system it is between 1.57 to 88.90 seconds. The time taken by Gentry's extended system is more as it is a fully homomorphic cryptosystem, whereas Paillier is an additively homomorphic cryptosystem. The fully homomorphic system can handle multiple operations (addition and multiplication) without decryption error, but this integer matching implementation does not make complete use of the fully homomorphic properties. Even though Paillier performs faster for this implementation, Gentry's extended method could be used for extending the encrypted domain applications performing processing on images. The extraction of characters can also be performed in encrypted domain using the fully homomorphic scheme.

The success rate of matching in encrypted domain using Paillier encryption and extended Gentry's fully homomorphic cryptosystem is 100% and hence the success rate of the whole system depends on successfully recognizing the license plate characters.

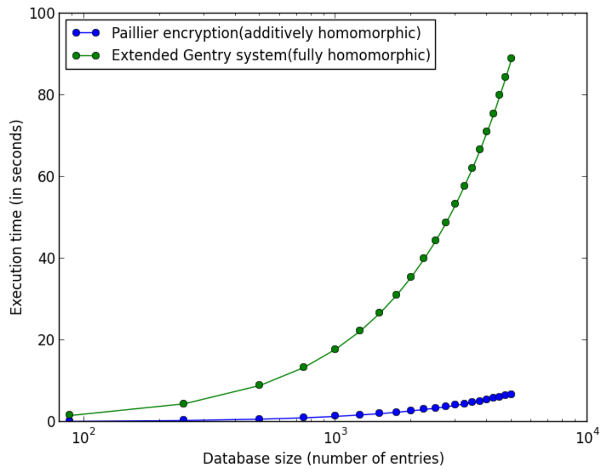


Fig. 2. Graph for the execution time taken for secure matching

V. Conclusion

With the increasing number of privacy and security threats, it is important to ensure that there is minimal information leak. We propose a novel system which includes a character recognition system for (Dutch) car license plates, which can be easily translated to the encrypted domain, and a secure integer matching algorithm using two protocols: Paillier encryption, and a version of Gentry's fully homomorphic cryptosystem. Experimental results for the different scenarios using two different cryptographic tools show that this line of research can be easily adopted in real life, eliminating a serious privacy concern for deploying car plate recognition systems. This work is one step towards integrating the fields of cryptography and license plate matching for privacy.

Acknowledgment

Thanks to Vrije Universiteit (VU) Fellowship Program and TNO for the funding. We would also like to thank Arjen Veenhuizen (TNO) for the support extended during the course of this work. This publication was supported by the Dutch national program COMMIT.

References

[1] D. J. Roberts and M. Casanova, "Automated License Plate Recognition (ALPR) Use by Law Enforcement: Policy and Operational Guide, Summary," Tech. Rep., Document 239605, IACP Technology Center, International Association of Chiefs of Police, September 2012.

[2] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *Signal Processing Magazine, IEEE*, vol. 30, no. 1, pp. 82–105, 2013.

[3] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. D. Labati, and P. Failla, "Privacy-preserving fingercode authentication," in *Proceedings of the 12th ACM workshop on Multimedia and security*. ACM, 2010, pp. 231–240.

[4] J. R. Troncoso-Pastoriza, D. Gonzalez-Jimenez, and F. Perez-Gonzalez, "Fully Private Noninteractive Face Verification," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 7, pp. 1101–1114, 2013.

[5] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 3, pp. 1053–1066, 2012.

[6] F. M. Oliveira-Neto, Lee D Han, and M. K. Jeong, "An Online Self-Learning Algorithm for License Plate Matching," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 1806–1816, 2013.

[7] S. S. M. Noor and N. M. Tahir, "Fusion of License Plate and Face Recognition for Secure Parking," *Jurnal Teknologi*, vol. 61, no. 1, 2013.

[8] L. Zheng and X. He, "Character segmentation for license plate recognition by k-means algorithm," in *Image Analysis and Processing–ICIAP 2011*, pp. 444–453. Springer, 2011.

[9] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. Von Deneen, and P. Shi, "An Algorithm for License Plate recognition Applied to Intelligent Transportation System," in *Intelligent Transportation Systems, IEEE Transactions on 12*, 2011, pp. 830–845.

[10] P. Kulkarni, A. Khatri, and P. Banga, "Automatic Number Plate Recognition (ANPR)," in *RADIOELEKTRONIKA 19th International Conference*, 2009, pp. 243–250.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology-EUROCRYPT'99*. Springer, 1999, pp. 223–238.

[12] C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," in *Advances in Cryptology–EUROCRYPT 2011*, pp. 129–148. Springer, 2011.

[13] J. Garay, B. Schoenmakers, and J. Villegas, "Practical and secure solutions for integer comparison," in *Public Key Cryptography–PKC 2007*, pp. 330–342. Springer, 2007.

[14] "Implementation codes," https://github.com/archanabs/Secure-LP-match.

[15] L. Zheng, X. He, B. Samali, and L. T. Yang, "Accuracy enhancement for license plate recognition," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 511–516.

[16] S. Karungaru, K. Terada, and M. Fukumi, "Character Recognition from Virtual Scenes and Vehicle License Plates Using Genetic Algorithms and Neural Networks," in *INTECH Open Access Publisher*, 2012.

[17] N. Thome, A. Vacavant, L. Robinault, and S. Miguet, "A cognitive and video-based approach for multinational License Plate Recognition," in *Machine Vision and Applications, Springer-Verlag*, 2011, pp. 389–407.