

Article

Machine Learning-Based Malicious X.509 Certificates' Detection

Jiaxin Li , Zhaoxin Zhang* and Changyong Guo*

Network Information Security Research Center, School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai 264200, Shandong, China; 19s030153@stu.hit.edu.cn

* Correspondence: heart@hit.edu.cn (Z.Z.); guocy@hit.edu.cn (C.G.)

Abstract: X.509 certificates play an important role in encrypting the transmission of data on both sides under HTTPS. With the popularization of X.509 certificates, more and more criminals leverage certificates to prevent their communications from being exposed by malicious traffic analysis tools. Phishing sites and malware are good examples. Those X.509 certificates found in phishing sites or malware are called malicious X.509 certificates. This paper applies different machine learning models, including classical machine learning models, ensemble learning models, and deep learning models, to distinguish between malicious certificates and benign certificates with Verification for Extraction (VFE). The VFE is a system we design and implement for obtaining plentiful characteristics of certificates. The result shows that ensemble learning models are the most stable and efficient models with an average accuracy of 95.9%, which outperforms many previous works. In addition, we obtain an SVM-based detection model with an accuracy of 98.2%, which is the highest accuracy. The outcome indicates the VFE is capable of capturing essential and crucial characteristics of malicious X.509 certificates.

Keywords: HTTPS; malicious X.509 certificates; machine learning



Citation: Li, J.; Zhang, Z.; Guo, C. Machine Learning-Based Malicious X.509 Certificates' Detection. *Appl. Sci.* **2021**, *11*, 2164. <https://doi.org/10.3390/app11052164>

Academic Editor: Luis Javier Garcia Villalba

Received: 18 December 2020

Accepted: 24 February 2021

Published: 1 March 2021

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



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The HTTP protocol is one of the basic standards for transporting information between different sites on the Internet. Since data transmitted by HTTP are easily intercepted by the man in the middle (man in the middle: the attacker who would like to steal communication data) of the transmission channel, HTTPS protocol is proposed to ensure security and stability of information communication. The critical contribution of HTTPS protocol is applying X.509 certificates, which are used to exchange asymmetric public keys and authenticate against public keys of communication partners. X.509 certificates are issued by a set of trusted Certification Authorities (CAs: authorities of signing and issuing X.509 certificates), the TBS (TBS: basic information, defined in RFC 5280, about certificate's issuer and subject) information of an end certificate is signed by the private key of its issuer to construct the signature. With the relationship of signing and being signed, a certificate chain is formed. When a client receives a server certificate, the client can check the certificate status obtained via Online Certificate Status Protocol (OCSP), and whether the root certificate in its certificate chain is trusted, the server certificate is expired or included in Certificate Revocation Lists (CRLs). The client will trust only the server certificate which passes the inspection.

The occurrence of X.509 certificates decreases the probability of being monitored by others to some extent, but it provides an illegal person with a helpful tool as well. According to the Phishing Activity Trends Report of 1st quarter 2020 (Phishing Activity Trends Reports: <https://apwg.org/trendsreports/>) released by Anti-Phishing Working Group (APWG), more and more phishing websites utilize X.509 certificates to encrypt communication information and pretend to be regular sites, which avoid the detection of malicious flow analysis tools. Figure 1 shows the percentage of phishing sites with HTTPS recently. From the statistical data, we can observe that the rate of phishing attacks hosted

on HTTPS reaches 74% approximately, which is high enough to attract more attention. In addition to phishing sites, the communication between malware and their Command Control (CC) servers leverage X.509 certificates to avoid the detection of traffic analysis tools frequently. The SSL Blacklist (SSLBL) project exposes many X.509 certificates used by botnet CC servers to transmit information with malware installed in remote bots. The number of this type of X.509 certificate is increasing over time. We call the certificates found during malicious activities like phishing and malware communication malicious certificates or abnormally used certificates.

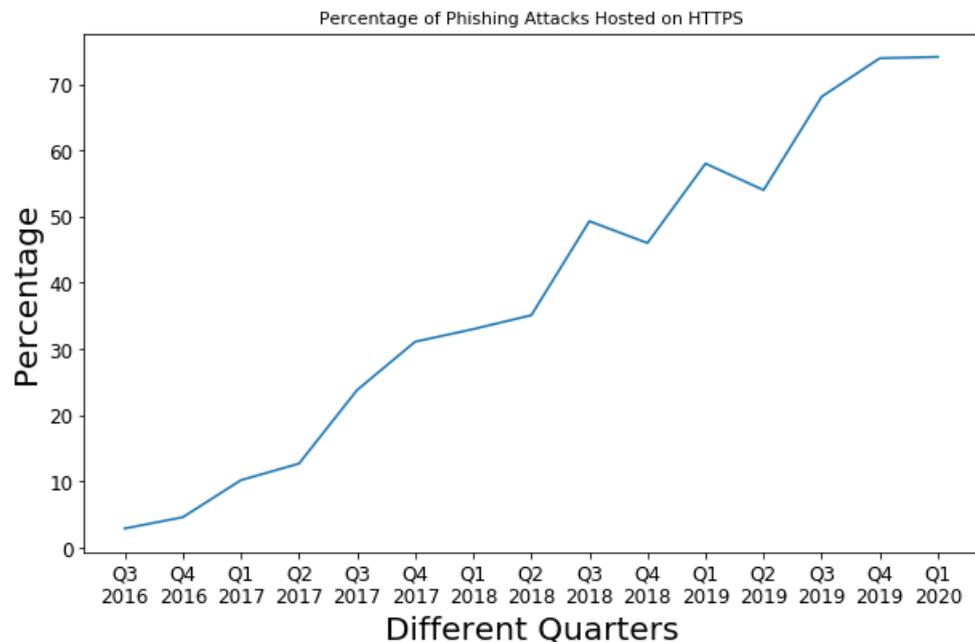


Figure 1. Percentage of phishing attacks hosted on HTTPS.

Being shielded by malicious certificates, phishing sites are more likely to be trusted by web browsers and even camouflage other popular websites and steal private information, which has enormous damage to the Internet. X.509 certificates are indeed created for safeguarding against malicious network impersonators. However, a trusted Certificate Authority (CA) may reissue X.509 certificates for a website by accident, which provides a chance to camouflage this website with the reissued certificate. For example, in 2011, the CA Comodo was compromised and reissued certificates for Microsoft and Google (Source: <https://blog.comodo.com/other/the-recent-ra-compromise/>). Suppose an attacker holds a reissued certificate for Google. In that case, he can set up a phishing website with the same interface as Google and use the reissued certificate for communication during the HTTPS connection, which is trust by browsers. If users login into this phishing website with their accounts, their information is stolen. The detail of the HTTPS connection with the X.509 certificate is illustrated in RFC 5280 [1]. In addition, the communications between malware and CC servers are unobservable, which has a destructive influence by way of snooping privacy and stealing important information. How to detect those malicious certificates is imperative and significant work.

There are many solutions to detect malicious certificates. The most straightforward method is filtering with known malicious ones. For example, Ghafir et al. [2] proposed a Malicious SSL certificate Detection (MSSLD) model to detect Advanced Persistent Threat (APT) communications based on a blacklist of malicious SSL certificates to filter connections. In addition, many researchers applied machine learning methods for detecting malicious certificates. Categorized by the type of detecting model, machine learning-based methods contain two types of models: one is the use of classical machine learning model, the other is applying neural network to the detection of malicious certificates. For example,

Mishari et al. [3] applied classical machine learning models to train separate models for distinguishing phishing and typosquatting with gathered certificates. Dong et al. [4] used deep neural networks for training rogue certificates detection models with artificial rogue certificates. The details of previous methods are illustrated in Section 2.

Although those previous methods have high accuracy in detecting malicious certificates, they cannot handle the situation which is strict with detection accuracy. Moreover, filtering malicious certificates with known malicious certificates has a low performance while encountering new malicious certificates. Both phishing certificates and malware-used certificates are malicious certificates. A detection method that can distinguish those two types of malicious certificates together will be time-saving and effortless. Therefore, we combine phishing certificates and malware-used certificates as malicious certificates to train models. Furthermore, the ensemble learning model has shown excellent performance in many competitions. To improve the detection accuracy as much as possible, we leverage the ensemble learning model to detect malicious certificates. In addition, features selected for training models are various, and most of them are insufficient.

To solve these problems and achieve our objectives, we apply machine learning models, including classical models, ensemble learning models, and deep learning models, to detect malicious certificates that combine phishing certificates and malware-used certificates. Features play a crucial role in training different models. Considering previous methods do not have comprehensive and minute feature extraction towards X.509 certificates, we design and implement Verification for Extraction (VFE). The VFE is intended for analyzing basic fields of certificates, checking certificates' conformation of criterion defined in RFC 5280, constructing and verifying certificate chain, and recording information for features extraction. The result shows that the VFE can capture crucial traits of certificates. The best model achieves an accuracy of 98.2% for detecting malicious and benign certificates, which is a higher score than state-of-the-art. Torroledo et al. [5] implemented a system that is capable of identifying malware certificates with an accuracy of 94.87%. Fasllija et al. [6] achieved a phishing attempts detecting system with an accuracy of 91% approximately. Our contributions can be summarized as follows:

1. Design and implement the VFE for verification and extracting characteristics from certificates. The result shows features extracted from the VFE are good at distinguishing malicious certificates from benign certificates.
2. Apply and compare various machine learning models for malicious certificate detection. We had tested the performance of different models in malicious certificate detection and have a good grasp of the advantages and disadvantages of those models.
3. Find the best model for detecting malicious certificates with an accuracy of 98.2%. After trying and optimizing different models, we found a model with the highest accuracy.

The remainder of this paper is organized as follows. In Section 2, we review related work in this field. Then, in Section 3, we illustrate the design and implementation of the VFE in detail. We analyze the total features extracted from the VFE and explain the reason for choosing them in Section 4. In Section 5, we describe selective machine learning models and their structures during experiments. Our experiments, including data collection, experimental design, and experimental results, are illustrated in Section 6. In Section 7, we make conclusions about the total work.

2. Related Work

Malicious certificate detection is a complementary and effective method to distinguish phishing websites and communications between malware and their CC servers. There were many studies related to this field. Apart from filtering malicious certificates with known malicious certificates, many researchers work out this problem with machine learning. Those machine learning methods were differential in data sources, feature engineering, model selection, and focused certificates.

Ghafir et al. [2] proposed a Malicious SSL certificate Detection (MSSLD) model to detect Advanced Persistent Threat (APT) communications based on a blacklist of mali-

cious SSL certificates to filter connections. To enhance detection ability, they updated the blacklist of malicious SSL certificates from different sources each day at 3:00 am. It is the most direct way to find APT communications. This model relies on malicious certificates from other sources, which cannot ensure the timeliness of detection while meeting new malicious certificates.

In an article by Mishari et al. [3], they trained Random Forest, Decision Tree, and Nearest Neighbor to detect web-fraud domains. They collected nine features of the certificate and some sub-features to train models. In addition, they analyzed the selected features minutely. The result shows that the highest phishing detection accuracy is 88%.

Xianjing et al. [7] analyzed attribute correction of certificates in the certificate chain and built up a probabilistic model SSLight to model attribute correction. Training the SSLight with a large number of regular certificates, they applied it to detect fake certificates. The model built in this paper is comprehensive and demands extensive data for training.

In an article by Dong et al. [8], they designed and implemented a real-time detecting system with certificate downloader, feature extractor, classification executor, and decision-maker parts. In addition, they collected 95,490 phishing certificates and 113,156 non-phishing instances for training Decision Tree, Random Forest, Naive Bayes Tree, Logistic Regression, Decision Table, and K-Nearest Neighbors. One advantage of this work is the timeliness of detection.

Faslija et al. [6] made use of Certificate Transparency (CT) logs to extract features and train classical models to detect phishing certificates. One innovation of this research was that their models divided certificates into five categories.

In another paper by Dong et al. [4], they applied deep neural networks to train detection models for rogue certificates. Considering data imbalance of rogue certificates, they changed tiny content of benign certificate to construct rogue certificate, which has an outstanding performance. Their experiments obtained the highest accuracy of 97.7%.

Torroledo et al. [5] leveraged long short-term memory (LSTM) to extract features from subject and issuer information. They used those features combined with other numerical features to train models for detecting phishing certificates and malware-used certificates, respectively. This work inspires our research, and we change the method to deal with subject information.

However, all these previous works of detecting malicious certificates suffer the disadvantages introduced in Section 1. To cope with these drawbacks, we design and implement the VFE to analyze basic fields of certificates, check certificates' conformation of criterion defined in RFC 5280, construct, and verify the certificate chain. During this process, we collect plentiful features of certificates, which is feature engineering. Obtaining sufficient features, we select classical machine models, ensemble learning models, and deep learning models to detect malicious certificates. In addition, we propose some novel tricks for extracting features from subject and issuer information of certificates.

The detection of malicious certificates is beneficial for detecting phishing sites or malware. There are some other researches of finding phishing sites or malware without X.509 certificates. In an article by Hutchinson et al. [9], they proposed a method of using the features of the URL to detect phishing ones. They considered different feature sets to detect phishing URLs with Random Forest (RF). The best model achieved an accuracy of 96.5%. Kulkarni et al. [10] implemented four classifiers with MATLAB to detect phishing websites with dataset from machine learning repository of The University of California, Irvine. Among four classifiers of the Decision Tree, Naïve Bayesian classifier, SVM, and neural network, the Decision Tree reached the highest accuracy of 91.5%.

3. Verification for Extraction (VFE)

The VFE is implemented to analyze certificates' basic fields, checking whether certificates conform with constraints agreed on RFC 5280, constructing and verifying the certificate chain. We can seize characteristics of certificates, as many as possible for picking up features during those processes. This process is conforming with capturing comprehen-

sive aspects about X.509 certificates. The integral architecture of the VEF includes basic analysis, standard checking, certificate chain construction, and certificate chain validation, four parts. The basic analysis part parses primary fields in the content of certificates for feature use. The standard checking part exams certificates' conformation to RFC 5280, not only the type of areas but also some fields' presence. The certificate chain construction part builds the certificate chain according to two methods. One is searching certificates database, and the other is obtaining upper certificates with the help of Authority Information Access (AIA: information for obtaining issuer's certificate). The certificate chain validation part verifies the certificate chain in multiple dimensions, including certificate policy mapping, path length constriction, name constrict, etc. In the following part of this section, the details of the VFE are illustrated.

3.1. Basic Analysis

The basic analysis part parses basic fields of a certificate, including certificate version, validation, serial number, public key, subject information, issuer information, extensions, error information, existence information, etc. Table 1 shows the details of basic fields. Those basic fields are contents of the certificate itself, and they play an essential role in the processes of the following parts and are basic information about certificates. We accomplish this part with the help of the python package of OpenSSL (OpenSSL: <https://www.openssl.org/>). The basic analysis part takes DER (DER: binary encoding scheme of X.509 certificates) or PEM (PEM: Base64 encoding scheme of X.509 certificates) type certificates as inputs and output contents of certificates with OpenSSL's help.

Table 1. Detail of basic fields of X.509 certificates.

Name	Description	Value Type
PEM certificate	PEM type certificate	bytes
version	version of certificate	integer
fingerprint	sha1 and sha256 fingerprints of certificate	string
serial number	serial number	integer
public key	public key of subject	dictionary
validation	validation information of certificate	dictionary
issuer	issuer information of certificate	dictionary
subject	subject information of certificate	dictionary
signature hash algorithm	hash algorithm used in signature	string
signature algorithm	asymmetrical algorithm of signing	string
signature	the signature of certificate	bytes
tbs certificate	the tbs information of certificate	bytes
extensions	extensions of certificate	dictionary

3.2. Standard Checking

The standard checking part carries out criterion checking with the guide of RFC 5280. For example, if the serial number is a positive integer no longer than 20 octets, whether certificates of versions 1 and 2 have extensions or the length of explicit text in certificate policy exceeds 200, etc. Table 2 shows the details of checking items. In a word, what we do in this part is find restrictions of RFC 5280 and check whether certificates are conforming with those restrictions. This is a necessary step for the reason that RFC 5280 is an agreement about X.509 certificates.

Table 2. Details of checking items.

Name	Description	Value Type
serial_number_not_conforming	Whether serial number is positive and no longer than 20 octets	integer
after_smaller_than_before	Whether after-time is small than before-time in validation	integer
extension_exist_in_wrong_version	Whether extensions are math with certificate version	integer
decipher_and_encipher_error	Whether the appearance of decipher and encipher is conforming	integer
explicit_text_exceed	Whether the length of explicit text exceeds 200	integer
only_consist_reasons_error	Whether cRLDistributionPoint only consist reasons	integer
keycertsign_not_conform_ca	Whether the set of keycertsign is conforming with ca field	integer
cA_with_empty_subject	Whether subject information is empty and the subject is ca	integer
CRLissuer_with_empty_subject	Whether subject information is empty and the subject is CRLissuer	integer

3.3. Certificate Chain Construction

The certificate chain construction part locates certificates' issuers until positioning their root certificates. The task of constructing the certificate chain for an end certificate is completed as its root certificate is exposed. There are two manners to locate the issuer's certificate. One is searching in the certificates database CCADB (CCADB: a common certificate authority database with root and intermediate certificates from <https://www.ccadb.org/forsearching>). The other is obtained according to access information in AIA. To promote the rate of building a complete certificate chain, we integrate two methods. Algorithm 1 describes the overall process of forming the certificate chain. Once the root certificate is found, or no issuer's certificate is obtained by trying two methods, the certificate chain construction is completed. To reduce the time of building a certificate chain, we give priority to the database for searching issuer's certificate since leveraging AIA information to obtain the issuer's certificate is time-consuming.

Algorithm 1: construct certificate chain

```

Input: certificate X
Output: certificate chain X_chain
X_chain=[];
X_chain.append(X);
temp_certificate=X;
while temp_certificate is not root certificate do
  if obtain upper certificate of temp_certificate by searching the CCADB then
    temp_certificate=upper certificate;
    X_chain.append(temp_certificate);
  else
    if obtain upper certificate of temp_certificate according to AIA information then
      temp_certificate=upper certificate;
      X_chain.append(temp_certificate);
    else
      print("Fail to construct certificate chain");
      break;
    end
  end
end
end
return X_chain;

```

3.4. Certificate Chain Validation

The certificate chain validation part is implemented for examining some fields for related certificates in the certificate chain. For instance, certificate policy mapping, path length constriction, name constrict, etc. Any implementation of certificate chain validation is required to be conforming to RFC 5280. Considering the time cost of implementing a tool

for verifying the certificate chain, we accomplish the certificate chain validation part with the help of OpenSSL. Many trusted certificates and Certificate Revocation Lists (CRLs) are required to be added into the validation context during the validation process. It is worth mentioning that we validate the certificate chain of each certificate in the certificate chain to find more possible errors. The detail of certificate chain validation is recognized by checking results of OpenSSL. The details of validation values are displayed on the official website of OpenSSL (OpenSSL verify: <https://www.openssl.org/docs/man1.0.2/man1/verify.html>).

We record the necessary information for extracting as many as possible features during the processing flow of four parts. With the help of the VFE, comprehensive and useful traits of certificates are collected.

4. Feature Engineering

Feature engineering is based on certificates' characteristics obtained from the basic analysis, standard checking, certificate chain construction, and certificate chain validation of the VFE. In addition, some outputs of packages and tools we use are taken into consideration. We also attempt to acquire six extra features based on what we gain already. Considering the number of features we extracted is comparably huge, we upload a description of all the features to our website (Our website: <https://github.com/fight-think/features-extraction>). In the following parts of this section, more details about the features of a certificate are explained.

4.1. Features from Cryptography

Cryptography is a python package that supports us to parse basic fields of certificates. There are few limitations, including versions, algorithms, and extensions supported in this package. Sometimes inputs cannot be parsed by this package. When one of those four situations occurs, the corresponding feature will be stored. In addition, the parsing process does not always go well, so 26 features about parsing errors are extracted. Table 3 illustrates some sample features from Cryptography. The full features from Cryptography are listed on our website (Our website: <https://github.com/fight-think/features-extraction>).

Table 3. Sample features from Cryptography.

Feature Name	Feature Description	Value Type
fail_load_with_cryptography	Whether fail to load certificate with Cryptography	integer
version_not_support	Whether version of certificate is supported	integer
algorithm_not_support	Whether signature algorithm is support	integer
extension_not_support	Whether has unsupported extensions	integer
version_parse_error	Whether meet error while parsing version information	integer
fingerprint_produce_error	Whether meet error while parsing fingerprint	integer
serial_number_parse_error	Whether meet error while parsing serial number	integer
public_key_parse_error	Whether meet error while parsing public key	integer
validation_parse_error	Whether meet error while parsing validation information	integer

4.2. Features from Basic Analysis

The basic fields of certificate contents are essential features. For example, signature algorithm, public key algorithm, version, country of the issuer, country of the subject, etc. Therefore, we collect 16 features about the basic contents of certificates and 11 features that are easily computed from the contents of certificates. Extensions of certificates are supplementary and explanatory information about certificates. Therefore, not all extensions are essential in a certificate. The presence of extensions shows the importance and completeness of certificates in some way. Each extension has a property about whether this

extension is critical or not. The difference between critical and not critical extensions is that critical extensions must be parsed and verified during certificate chain validation. To seize all possible keys, we sort out 14 existential features and 14 critical features. Table 4 shows some sample features from the basic analysis. The full features from the basic analysis are listed on our website (Our website: <https://github.com/fight-think/features-extraction>).

Table 4. Sample features from basic analysis.

Feature Name	Feature Description	Value Type
issuer_country	The country of issuer	integer
subject_country	The country of subject	integer
digital_signature	The keyusage information	integer
content_commitment	The keyusage information	integer
serial_number_length	The length of serial number	integer
has_expire	Whether certificate has expired	integer
issuer_is_empty	Whether issuer information is empty	integer
public_key_parse_error	Whether meet error while parsing public key	integer
validation_parse_error	Whether meet error while parsing validation information	integer

4.3. Features from Standard Checking

RFC 5280 documents common agreements of CAs, certificate users, governments, and several organizations to X.509 certificates. Any implementation of RFC 5280 should be conforming to its criterion. Kumar et al. [11] implemented a frame for checking whether certificates issued by CAs complied with standards defined by RFC 5280, CAs, and browser corporations. The results showed the number of unmatched certificates was decreasing and the percentage reduced to 0.02% in 2017. Although the rate of inconsistent certificates is lower, it is necessary to check certificates' match with RFC 5280. Therefore, we extract 10 features to represent checking results. Table 5 illustrates the details of features from standard checking.

Table 5. Details of features from standard checking.

Feature Name	Feature Description	Value Type
serial_number_not_conforming	Whether serial number is positive and no longer than 20 octets	integer
after_smaller_than_before	Whether after-time is small than before-time in validation	integer
extension_exist_in_wrong_version	Whether extensions are math with certificate version	integer
decipher_and_encipher_error	Whether the presence of decipher and encipher is conforming	integer
explicit_text_exceed	Whether the length of explicit text exceeds 200	integer
only_consist_reasons_error	Whether cRLDistributionPoint only consist reasons	integer
keycertsign_not_conform_ca	Whether the set of keycertsign is conforming with ca field	integer
cA_with_empty_subject	Whether subject information is empty and the subject is ca	integer
CRLissuer_with_empty_subject	Whether subject information is empty and the subject is CRLissuer	integer
subject_with_subaltno_critical	Whether subject information is empty and subject alternative name is not critical	integer

4.4. Features from Certificate Chain Construction

We care about the result of construction, error occurrence, and the subject information of certificates in the certificate chain during the certificate chain construction. Drury et al. [12] compared phishing certificates with regular certificates in many dimensions. The results indicated subject and issuer information of certificates were vital to distinguish them. Fadai et al. [13] analyzed trusted SSL root CAs of different modern browsers and operating systems. It shows that the trustworthiness of SSL root CAs is relative to their original countries. Torroledo et al. [5] analyzed issuer and subject information minutely while carrying out feature engineering. Inspired by previous work, we connect subject information of each certificate in the certificate chain to an integral fragment, which is regarded as a trick for discovering the relationship between issuer and subject. We apply two ways to extract features from the text of subject information, and one is the bag of words (BOW) [14], the other is using trained fast text [15] to build embedding word vector of text information. We apply two types of neural networks to deal with embedding word vector [16]. Apart from the text of subject information, we record six more features about errors and the results of certificate chain construction. Table 6 illustrates the details of features from certificate chain construction.

Table 6. Details of features from certificate chain construction.

Feature Name	Feature Description	Value Type
mid_cert_parse_error	Whether middle certificate in chain parses with error	integer
mid_cert_without_ca_issuer	Whether middle certificate in chain does not have ca issuer information	integer
mid_cert_aia_error	Whether error occurs in obtaining issuer's certificate with AIA	integer
chain_len	The length of certificate chain	integer
construct_result	The construction result	integer
subject_infor_chain	The chain contains subject information	text

4.5. Features from Certificate Chain Validation

Akhawe et al. [17] illustrated the procedure of browsers' checking to server certificate and analyzed reasons that many Transport Layer Security (TLS) warnings were wrongly issued. Certificate chain validation is an essential tactic to identify certificates' authenticity. Therefore, we obtain 74 related features, which compose 71 validation flags of an end certificate from OpenSSL and three features indicating the validation of middle certificates in the certificate chain. As a result that the number of features is relatively large, we illustrate some sample features from certificate chain validation in Table 7. The full features from certificate chain validation are listed on our website (Our website: <https://github.com/fight-think/features-extraction>).

Table 7. Sample features from certificate chain validation.

Feature Name	Feature Description	Value Type
mid_cert_verify_error	Whether error occurs in verifying chain of middle certificate	integer
mid_cert_basic_validate_error	Whether error occurs in validating basic fields of middle certificate	integer
verify_result_error	The result of verifying chain of an end certificate	integer
unspecified_certificate_verification_error	An unspecified verification error	integer
unable_to_get_issuer_certificate	The certificate cannot get issuer's certificate	integer
CRL_path_validation_error	Error occurs while validating Certificate revocation list(CRL)	integer
application_verification_failure	An application specific error set by application callback	integer
unable_to_get_CRL_issuer_certificate	Cannot get issuer certificates of certificates in CRL	integer
unhandled_critical_extension	The return result of OpenSSL for verifying chain of an end certificate	integer

4.6. Extra Features

The seven extra features are discovered with the help of what we gain already. The first one is judging whether the certificate is an Extended Validation (EV) certificate. EV certificates are issued with more checking and more expense, which makes them more credible. Torroledo et al. [5] selected whether the certificate was an EV certificate while extracting features as well. We accomplish it with a list of certificate policies that demonstrate whether one certificate is an EV certificate or not. In addition, whether the root certificate of an end certificate is trusted by hardware and software companies, including Microsoft, Apple, Cisco, and Mozilla, is an important mark of credibility. Therefore, we obtain four features indicating the trust of Microsoft (Microsoft: <https://ccadb-public.secure.force.com/microsoft/IncludedCACertificateReportForMSFT>), Apple (Apple: <https://support.apple.com/en-us/HT209143>), Cisco (Cisco: <https://www.cisco.com/security/pki/>), and Mozilla (Mozilla: <https://ccadb-public.secure.force.com/mozilla/CACertificatesInFirefoxReport>) according to comparison of certificates' fingerprints collected from websites. Finally, the Alexa rank of the subject domain name is searched from the ranking file (Alexa rank: <https://www.alexa.com/topsites>), and the result shows that the Alexa rank is a useful feature. Table 8 illustrates the details of extra features.

Table 8. Details of features from extra features.

Feature Name	Feature Description	Value Type
is_ev	Whether certificate is extended validation certificate	integer
ios_trust	Whether root certificate is trusted by Apple	integer
cisco_trust	Whether root certificate is trusted by Cisco	integer
microsoft_trust	Whether root certificate is trusted by Microsoft	integer
mozilla_trust	Whether root certificate is trusted by Mozilla	integer
subject_domain_alexa_rank	The Alexa rank of subject domain name	integer
error_occur	Whether error occurs during extracting features	integer

5. Model Illustration

We apply classical machine learning models, ensemble machine learning models, and deep learning models to detect malicious certificates. Among three types of models, classical machine learning models and ensemble machine learning models are proposed by previous researchers, which are illustrated in the following parts. We do not modify them for applying them to detect malicious certificates. The structures of deep learning models are designed and implemented by us with the help of third packages including Pytorch (Pytorch: <https://pytorch.org/>), Sklearn (Sklearn: <https://scikit-learn.org/stable/>), Numpy (Numpy: <https://numpy.org/>), nltk (nltk: <https://www.nltk.org/>), and Pandas (Pandas: <https://pandas.pydata.org/>). With a comparison of different models, we have a profound command to the detection of malicious certificates. In the following parts of this section, specific models and their structures are illustrated.

5.1. Classical Machine Learning

The classical machine learning models we select include Logistic Regression (LR), Decision Tree (DT), and Support Vector Machine (SVM). The Logistic Regression was proposed by Joseph Berkson [18], it leveraged a sigmoid [19] function to accomplish nonlinearization and control the outcome range 0 to 1 which presents the probability of outputting value 1. Support Vector Machine is aimed at looking for a hyperplane which distinguishes two types of data with the help of a kernel trick if linear classifier cannot work [20]. Training the Decision Tree model is a process of building a Decision Tree with the best decision rules while making a decision about which feature is selected as judgment [21]. Those classical models have a long history and show their performances in many problems, especially in the problems where data are not so large, and the relationship between data is relatively simple. We apply classical machine learning to the detection of malicious certificates with the purpose of reference experiments.

5.2. Ensemble Machine Learning

Ensemble machine learning models [22] use multiple machine learning algorithms to predict the result rather than obtain the predictive result from one of the constituent learning algorithms alone, which always have a better performance. Among several competitions about machine learning, the ensemble machine learning method achieves a relatively high score. Considering different characters and performance of varying ensemble machine learning models, we select four models, including Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), and Category Boosting (CatBoost), to model four classifiers. Random Forest combines multiple decision trees to make the final predictive result, which always has a better performance than a single decision tree [23]. XGBoost is a scalable end-to-end tree boosting system with a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning, which has achieved great success in several machine learning challenges [24]. LightGBM promotes the efficiency and scalability of Gradient Boosting Decision Tree (GBDT) implementation while handling high dimension and extensive data with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [25]. Compared with XGBoost and LightGBM, CatBoost focuses on prediction shift with two algorithmic advances composing ordered boosting and an innovative algorithm for processing categorical features [26]. The data processing of classical and ensemble models are displayed in Figure 2. After splitting and clearing text information of `subject_infor_chain`, we count the most frequent 30 words and label their presence in one `subject_infor_chain` as features. Combining those features with 182 numerical features, we train several classical and ensemble models.

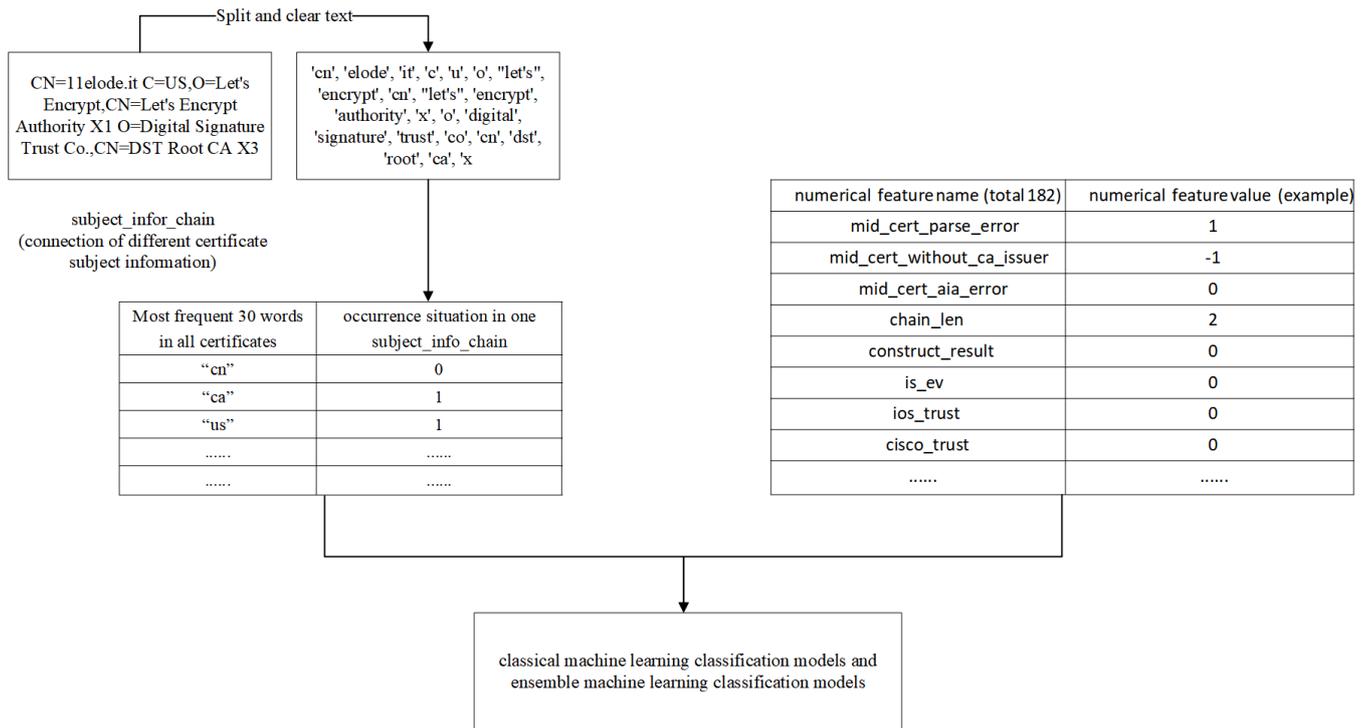


Figure 2. The data processing of classical and ensemble models.

5.3. Deep Learning

Since Krizhevsky et al. [27] applied a convolutional neural network (CNN) to an image classification challenge in 2012 and won the championship in this competition, deep learning [28] ravaged the world and achieved outstanding results in many fields. To further deal with the words embedding vector of subject information and acquire possible high accuracy models, we use CNN and LSTM to extract features from the words embedding vector and merge those new features with previous features to train classifiers. The critical operations of the convolutional neural network are convolution and pooling, which have superb efficiency in seizing critical features [27]. Long short-term memory is an improved version of the recurrent neural network (RNN) with a cell, an input gate, an output gate, and a forget gate in each unit. It reduces the possibility of gradient vanishing and gradient explosion, which are frequent in RNN [29]. The architecture of networks with CNN and LSTM used in our experiment are displayed in Figure 3. The total architecture includes handling words embedding vectors and combining previous numerical features and features extracted by neural networks to train a classifier. The words embedding vector is obtained by a pre-trained model with an input of 30 words in one subject_infor_chain at most. The difference between CNN based classification model and LSTM based classification model is the neural network for handling word embedding vectors. In addition, we illustrate the detail of hidden layers in CNN-based and LSTM-based models in Appendix A.

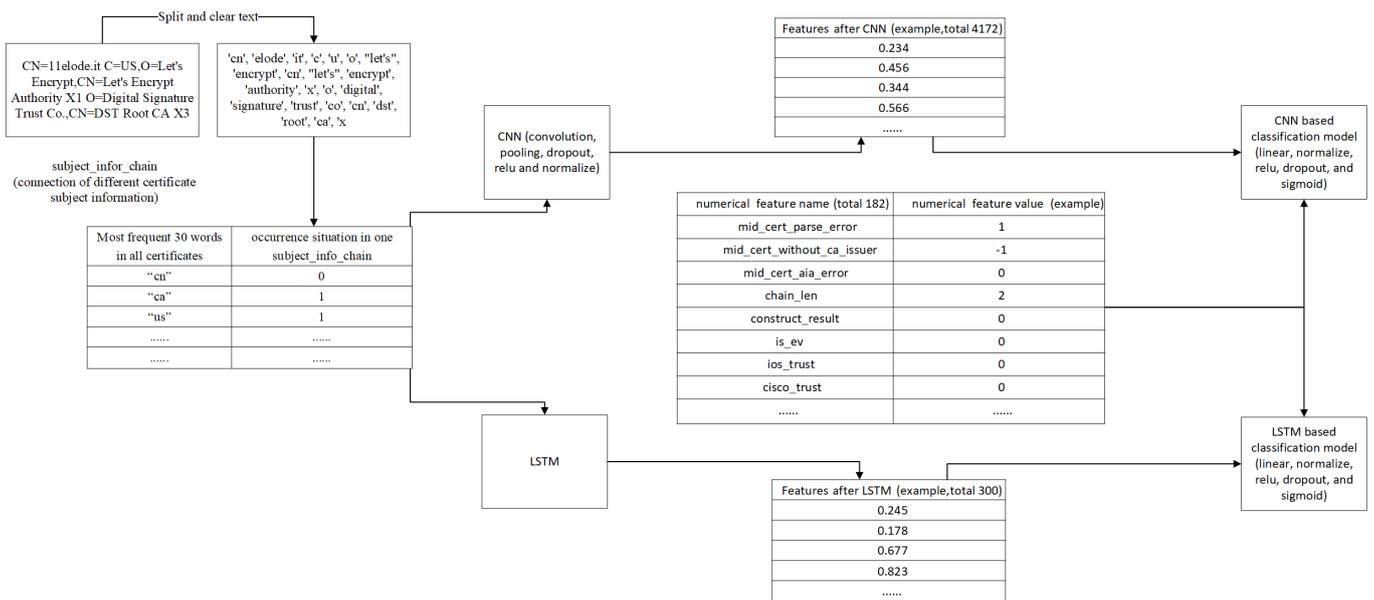


Figure 3. The architecture of networks with convolutional neural network (CNN) and long short-term memory (LSTM).

6. Experiment

6.1. Data Collection

To obtain models with better generalization performance and stability, we feed many data to models, especially deep learning models. The truth is that the number of malicious certificates is far less than the number of benign certificates. Therefore, the principle of use is collecting plentiful malicious certificates and a corresponding number of benign certificates. We acquire malicious certificates from two sources, and one is Uniform Resource Locators (URLs) of phishing websites provided by PhishTank (PhishTank: <https://www.phishtank.com/>). The other is fingerprints of malicious certificates collected by the SSLBL project (SSLBL: <https://sslbl.abuse.ch/>). Connecting URLs with HTTPS in port 443 and obtaining possible certificates sent from servers during the process of ServerHello, we collect 1711 malicious certificates. We apply fingerprints provided by the SSLBL project as searching keys to search at crt.sh (Crt.sh: <https://crt.sh/>) and censys.io (Censys: <https://censys.io/>) to get certificates in PEM format. We finally obtain 611 malicious certificates in this way. The total number of malicious certificates we collect is 2322. Compared with malicious certificates, benign certificates are adequate and easier to obtain. We leverage part domains of the Alexa top 1 million sites (Alexa rank: <https://www.alexa.com/topsites>) as seeds to get their certificates with the help of HTTPS protocol. The number of benign certificates we collect is 11,909, which is imbalanced. To make fair use of malicious certificates, we select 9288 (4×2322) benign certificates whose domain names have a high Alexa rank for different experiments. Considering to train models with great generalization ability, we adopt to re-sampling malicious certificates, which makes data balance in training and validation. To make it easier for reproduction, we release the features of all malicious and benign certificates on our website (Our website: <https://github.com/fight-think/features-extraction>).

6.2. Experimental Design

All the experiments are run on ThinkPad Carbon X1 2019 with 8G RAM hardware, Intel(R) Core(TM) i5-8265U CPU, and 512G SSD. We divide collected malicious and benign certificates into different datasets according to the Alexa rank of domain names or re-sampling malicious certificates. Just as Figure 4 shows, we select 9288 (4×2322) benign certificates with higher Alexa rank and divide them into b1, b2, b3, and b4 four parts, each of which has the same number of certificates with malicious certificates part, m1. Then we combine m1 with b1, b2, b3, and b4 for constituting Dataset1, Dataset2, Dataset3, and Dataset4, respectively. Three more datasets are formed by re-sampling malicious

certificates and combining the same number of benign certificates with them. We apply different machine learning models with seven datasets, including LR, DT, SVM, RF, XGBoost, CatBoost, LightGBM, CNN-based model, and LSTM-based model, to fit each dataset and compare the performance of all models in various datasets.

Benign certificates	b1	b2	b3	b4				
Malicious certificates	m1							
Dataset1	m1	b1						
Dataset2	m1	b2						
Dataset3	m1	b3						
Dataset4	m1	b4						
Dataset5	m1	m1	b1	b2				
Dataset6	m1	m1	m1	b1	b2	b3		
Dataset7	m1	m1	m1	m1	b1	b2	b3	b4

Figure 4. The combination of different datasets.

While training a specific model on one dataset, we use eight-fold cross-validation to find the best model. There are two reasons for using eight-fold cross-validation. The first is that the number of certificates in the experiment is relatively small. If dividing the data into ten pieces, the number of certificates used for testing is relatively small, which will affect the computation of the evaluation metric. The other is that if dividing the data into five pieces, the number of certificates used for training will decrease, which will affect the performance of the model. The experiments with five-fold cross-validation and ten-fold cross-validation indicate them. Therefore, we use eight-fold cross-validation.

During the training process, we split 15% data as testing data and apply eight-fold cross-validation to the remaining data for finding the best model, which means seven folds of data for training and one fold for validation. The model with the highest score of evaluation metric on validation data is regarded as the best model. In classical and ensemble machine learning models, GridSearchCV (GridSearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) is utilized for searching for the best model. In deep learning models, we obtain the best model during updating parameters and altering data. After getting the best model, we test the best model’s performance in testing data and calculate the time cost from training to testing.

We select accuracy as the evaluation metric to measure different models’ performance. Accuracy is associated with true positives, true negatives, false positives, and false negatives. In our experiments, benign certificates are considered positive items, and malicious certificates are negative. There are two main reasons for choosing accuracy as the evaluation metric. To begin with, finding benign certificates is equally important to detecting malicious certificates in our experiments. Comparing with precision, recall, or F1 score, accuracy is suitable for balancing these two abilities. What is more, many previous researchers

evaluate their models with accuracy. For example, Dong et al. [30], Torroledo et al. [5], Fasllija et al. [6], and so on. To make it easier to compare with them, we select accuracy as the evaluation metric.

6.3. Results

Training several models with each dataset, we record the best model's testing accuracy in eight-fold cross-validation and the validation accuracy of eight models trained with corresponding training data during this process. In addition, the time cost from eight-fold cross-validation to testing the best model is calculated. What is more, we pay attention to the importance value of features in DT, RF, XGBoost, CatBoost, and LightGBM models.

6.3.1. Accuracy

During the process of eight-fold cross-validation, eight models are trained with seven folds data. The validation accuracy of eight models on remaining fold data is recorded. Among these eight models, the one with the highest validation accuracy is selected as the best model. Then we obtain the testing accuracy by testing the best model with testing data. Table 9 shows validation accuracy of eight-fold cross-validation with different models on Dataset6. The last column of this table is the standard deviation(std) of validation accuracy values, which show the corresponding model's stability. The mean and standard deviation of eight models expose that SVM, XGBoost, and LightGBM have higher validation accuracy and lower std than other models. In addition, comparing with classical and deep learning models, ensemble models have higher average validation accuracy and lower average std. Therefore, ensemble learning models are the most efficient and stable models.

Table 9. Validation accuracy of eight-fold cross-validation on Dataset6.

Accuracy	model1	model2	model3	model4	model5	model6	model7	model8	mean	std
LR	0.938	0.926	0.937	0.935	0.928	0.930	0.930	0.940	0.933	0.00494
DT	0.943	0.950	0.951	0.946	0.949	0.947	0.935	0.932	0.944	0.00652
SVM	0.969	0.970	0.980	0.978	0.978	0.973	0.972	0.978	0.975	0.00388
RF	0.952	0.949	0.957	0.955	0.955	0.945	0.960	0.960	0.954	0.00493
XGBoost	0.982	0.974	0.974	0.971	0.976	0.976	0.970	0.975	0.975	0.00330
LightGBM	0.981	0.975	0.976	0.974	0.977	0.976	0.975	0.976	0.976	0.00207
CatBoost	0.959	0.956	0.957	0.958	0.960	0.954	0.963	0.955	0.958	0.00256
CNN	0.850	0.853	0.854	0.862	0.863	0.856	0.872	0.864	0.859	0.00679
LSTM	0.860	0.857	0.861	0.864	0.865	0.868	0.870	0.873	0.865	0.00504

In addition to eight-fold cross-validation, we adopt five-fold cross-validation and ten-fold cross-validation to find the best model. Table 10 shows validation accuracy of five-fold cross-validation with different models on Dataset6. Table 11 shows validation accuracy of ten-fold cross-validation with different models on Dataset6. We compare the mean validation accuracy and the standard deviation (std) of different models. The results show validation accuracy of eight-fold cross-validation is higher than five-fold cross-validation and partially higher than ten-fold cross-validation. The std of eight-fold cross-validation is lower than ten-fold cross-validation and not too higher than five-fold cross-validation. That shows eight-fold cross-validation is more suitable.

Table 10. Validation accuracy of five-fold cross-validation on Dataset6.

Accuracy	model1	model2	model3	model4	model5	mean	std
LR	0.928	0.934	0.933	0.930	0.936	0.932	0.00297
DT	0.950	0.928	0.950	0.939	0.948	0.943	0.00856
SVM	0.969	0.975	0.978	0.969	0.974	0.973	0.00349
RF	0.948	0.955	0.954	0.952	0.957	0.953	0.00303
XGBoost	0.973	0.974	0.971	0.976	0.969	0.972	0.00231
LightGBM	0.976	0.974	0.973	0.977	0.972	0.974	0.00208
CatBoost	0.957	0.959	0.955	0.958	0.957	0.957	0.00130
CNN	0.848	0.843	0.850	0.852	0.853	0.849	0.00354
LSTM	0.854	0.852	0.858	0.860	0.862	0.857	0.00370

Table 11. Validation accuracy of ten-fold cross-validation on Dataset6.

Accuracy	m	m	m	m	m	m	m	m	m	m	mean	std
LR	0.938	0.921	0.947	0.923	0.938	0.925	0.938	0.922	0.932	0.941	0.933	0.00868
DT	0.953	0.930	0.938	0.935	0.955	0.930	0.946	0.935	0.948	0.948	0.942	0.00876
SVM	0.968	0.973	0.979	0.975	0.979	0.984	0.976	0.971	0.983	0.978	0.977	0.00484
RF	0.940	0.948	0.961	0.949	0.953	0.955	0.956	0.951	0.955	0.956	0.952	0.00527
XGBoost	0.978	0.972	0.970	0.976	0.969	0.978	0.978	0.980	0.972	0.979	0.975	0.00383
LightGBM	0.981	0.976	0.973	0.975	0.973	0.971	0.971	0.980	0.975	0.975	0.975	0.00316
CatBoost	0.952	0.956	0.963	0.958	0.959	0.956	0.960	0.961	0.959	0.953	0.958	0.00335
CNN	0.852	0.854	0.855	0.863	0.864	0.860	0.874	0.870	0.872	0.873	0.864	0.00789
LSTM	0.862	0.859	0.864	0.867	0.868	0.867	0.875	0.873	0.874	0.878	0.869	0.00583

Finding the best model with eight-fold cross-validation, we feed testing data into the best model for obtaining testing accuracy. The result is displayed in Table 12. To begin with, comparing the results of Dataset1, Dataset2, Dataset3, and Dataset4, we find that feeding benign certificates with a high Alexa rank of subject domain names to models can result in a tiny improvement to testing accuracy. Furthermore, as the number of certificates increases in Dataset5, Dataset6, and Dataset7, the promotion of testing accuracy is various. SVM has the largest improvement, and ensemble learning models have a small boost with more training data. As we can see, the highest testing accuracy is obtained by SVM in Dataset7. What is more, compared with classical and deep learning models, ensemble learning models have higher average testing accuracy, which indicates ensemble models are the most efficient. Finally, the average testing accuracy of different models in seven datasets is 92.7%, and all models can achieve a relatively high testing accuracy, which shows the effectiveness and necessity of features extracted by the VFE.

Table 12. Testing accuracy of the best model.

Accuracy	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Average
LR	0.938	0.930	0.928	0.934	0.933	0.938	0.934	0.934
DT	0.955	0.941	0.954	0.954	0.971	0.966	0.950	0.956
SVM	0.835	0.827	0.812	0.821	0.976	0.978	0.982	0.890
RF	0.960	0.943	0.955	0.953	0.968	0.965	0.954	0.957
XGBoost	0.963	0.947	0.958	0.957	0.968	0.971	0.965	0.961
LightGBM	0.967	0.951	0.953	0.951	0.973	0.974	0.966	0.962
CatBoost	0.957	0.946	0.954	0.955	0.970	0.965	0.954	0.957
CNN	0.872	0.860	0.859	0.862	0.865	0.866	0.872	0.865
LSTM	0.869	0.857	0.860	0.863	0.865	0.868	0.873	0.865
Average	0.924	0.911	0.915	0.917	0.943	0.943	0.939	0.927

What is more, the ANalysis Of Variance (ANOVA) results in Table 13 show different models have a significant influence on testing accuracy with a high F-Statistic score. During calculation, each group includes the best models' testing accuracy with different datasets under one algorithm. The ANOVA calculation detail is illustrated on the website (Calculation of ANOVA: <https://goodcalculators.com/one-way-anova-calculator/>).

Table 13. The ANOVA results of different models.

Source	Degree of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Statistic
Between Groups	8	0.0989	0.0124	15.0821
Within Groups	54	0.0443	0.0008	
Total:	62	0.1432		

6.3.2. Importance Analysis

In addition to the validation and testing accuracy of different models, we analyze the importance values of the features in the best model of DT, RF, XGBoost, LightGBM, and CatBoost. Figure 5 shows the top 10 important features in XGBoost and their scores. The score is the improvement of accuracy brought by a feature to the branches it is on.

Figure 6 shows the top 10 important features in LightGBM and their scores. The score is the relative number of times a particular feature occurs in all splits of the model's trees.

Figure 7 shows the top 10 important features in CatBoost and their scores. Refer to official document (LightGBM document: https://catboost.ai/docs/concepts/fstr.html#fstr_regular-feature-importance), the score is computed with following Formula (1) and (2). c_1, c_2 are the number of objects in each leaf and v_1, v_2 are the formula values in the left and right leaves.

$$feature_importance_F = \sum_{trees, leafs_F} (v_1 - avr)^2 * c_1 + (v_2 - avr)^2 * c_2 \quad (1)$$

$$avr = \frac{v_1 * c_1 + v_2 * c_2}{c_1 + c_2} \quad (2)$$

Figure 8 shows the top 10 important features in DT and their scores. The score is computed as the (normalized) total reduction of the criterion brought by that feature, which is known as the Gini importance (Gini importance: <https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3>).

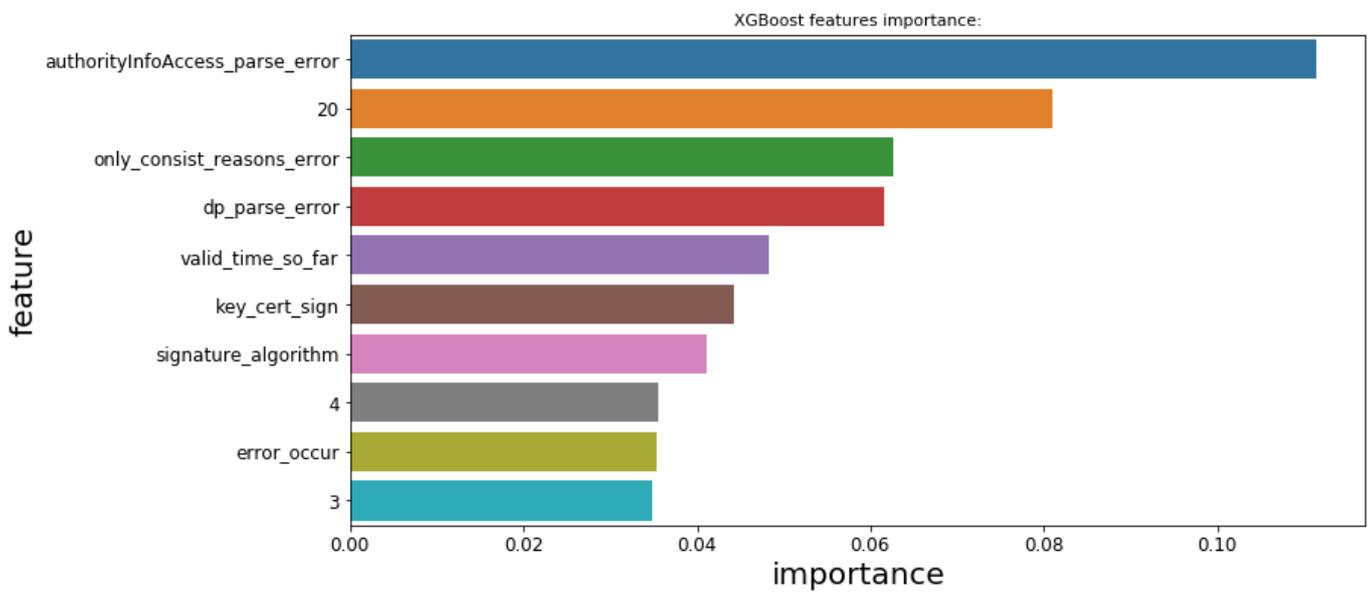


Figure 5. Important features of XGBoost.

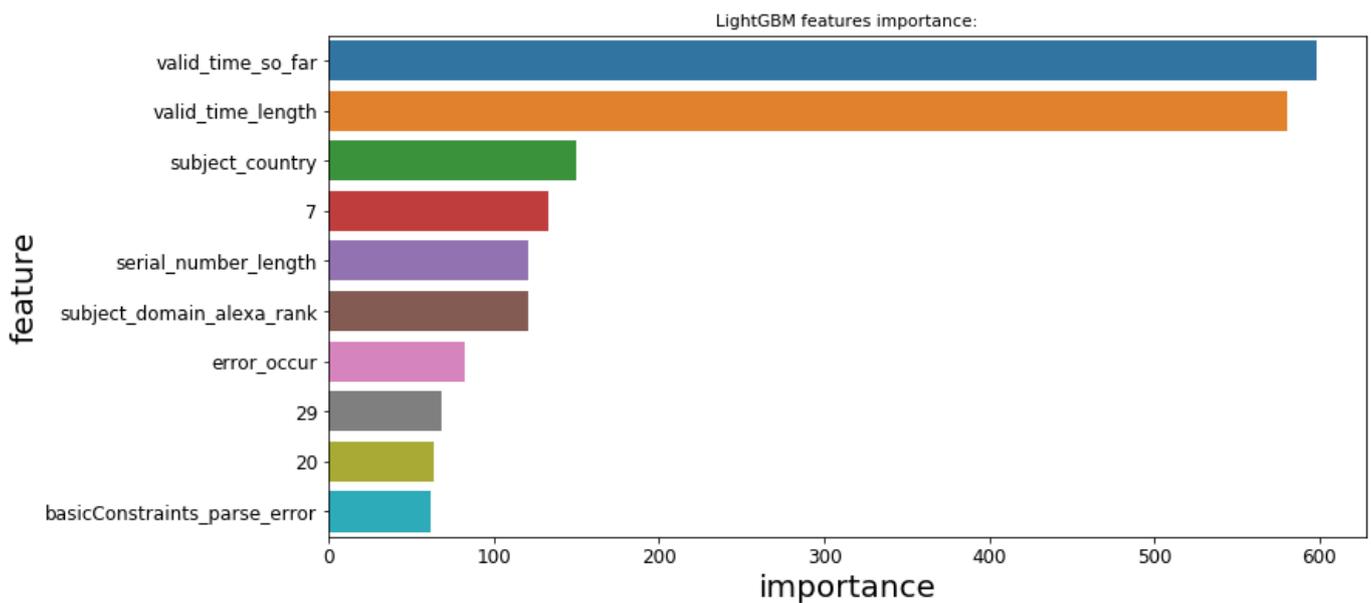


Figure 6. Important features of LightGBM.

Figure 9 shows the top 10 important features in RF and their scores. The score is the average feature importance of trees in the forest. The feature importance of each tree is computed with Gini importance.

As a result that the feature importance property is not provided in every model implementation, we cannot obtain all models' feature importance. Therefore, we analyze the feature importance of XGBoost, LightGBM, CatBoost, DT, and RF. From the top 10 important features in each model, we can see that "valid_time_so_far" is the most important feature in three out of five models. If the certificate is not expired, "valid_time_so_far" is the hours from the issue date to the feature extraction date. If the certificate is expired, "valid_time_so_far" means valid hours when it is not expired. The result exposes that "valid_time_so_far" is the most important feature in our experiments. In addition, we count feature occurrence in the top 10 important features of all five models. Figure 10 shows the results. We can see that there are many common features in the top 10 important features of all five models, which indicates the coincidence of important characteristics

to some extent. What is more, the features in Figure 10 appear in the top 10 important features of all five models more than one time. Therefore, we regard these features as vital features. Among these features, “20” represents the appearance of “ou” in the subject information of certificates, and other features are illustrated on our website (Our website: <https://github.com/fight-think/features-extraction>).

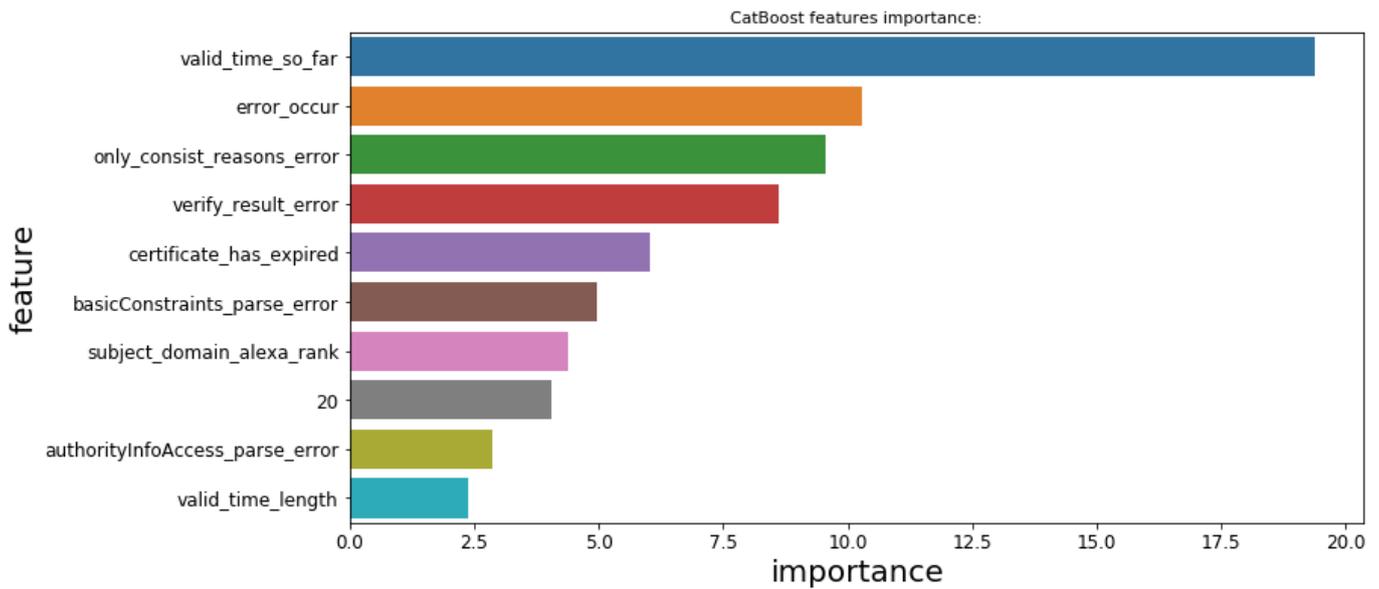


Figure 7. Important features of CatBoost.

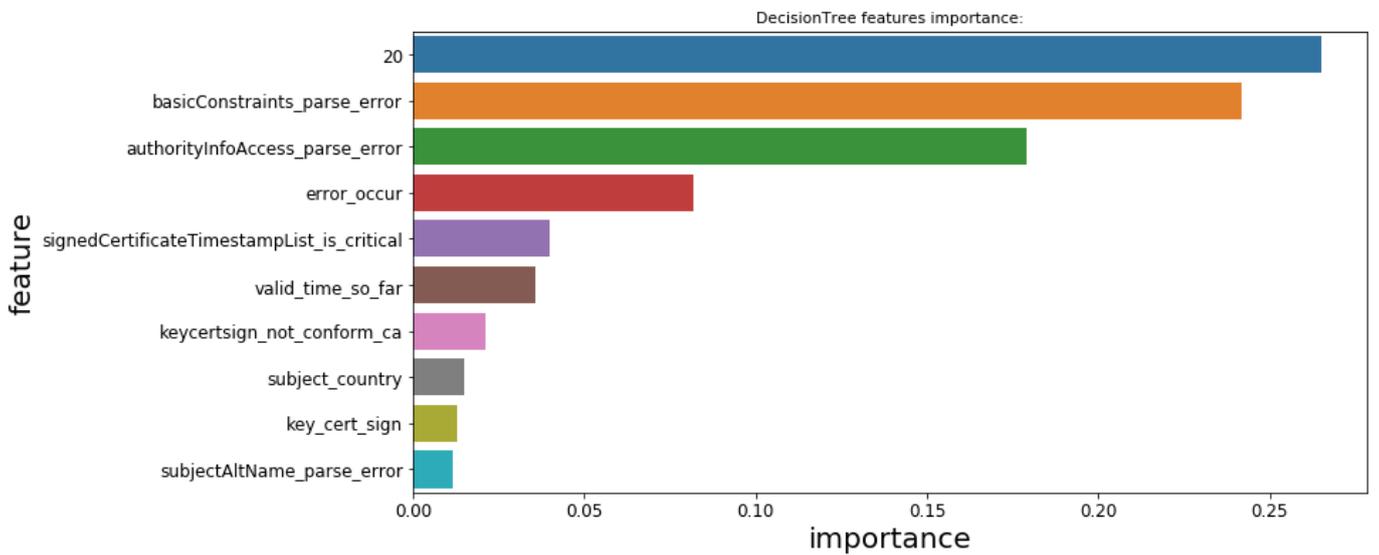


Figure 8. Important features of Decision Tree (DT).

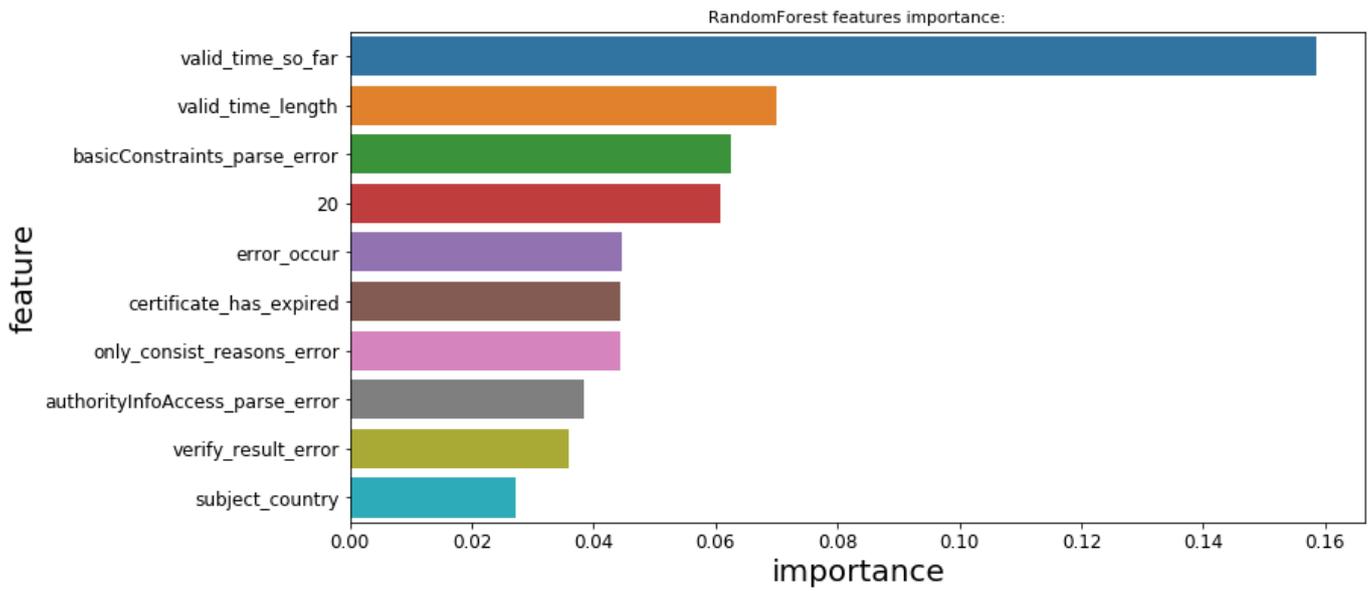


Figure 9. Important features of Random Forest (RF).

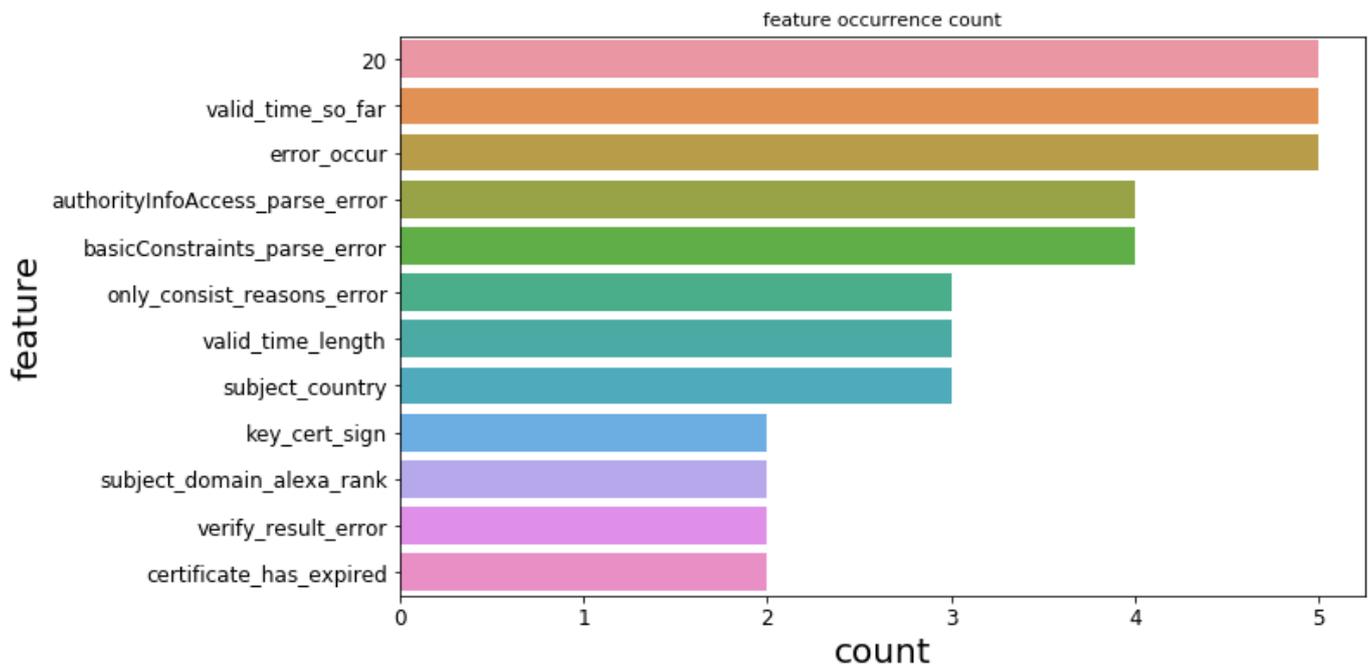


Figure 10. Feature occurrence count in top 10 important features of five models.

6.3.3. Time Consumption

Time consumption is an important metric of models, which reflects the computation resource consumption of training models. We record the time cost of eight-fold cross-validation and test the best model with testing data within one dataset, shown in Formula (3). Among the formula, n is the number of folds in experiments. Time(i) presents the training model’s time cost with n-1 folds data and validation on remaining fold data. Test_time means the time cost of feeding testing data into the best model of cross-validation. In our experiments, we use eight-fold cross-validation, and n is eight.

$$Time_consumption = \sum_{i=1}^{i=n} time(i) + test_time \tag{3}$$

Table 14 shows the time consumption of training Dataset6, reflecting the total time cost. It shows training models with deep neural network spends far more time than classical models and ensemble models. Among different classical models, SVM is the most time-consuming. In addition, in different ensemble learning models, XGBoost takes more time than the other two ensemble models. What is more, the average time consumption of ensemble models is 74.47 s, which is faster than 90.51 s in classical models and 10,502.52 s in deep models. The result shows ensemble models demand less time for training, which means less computation resource consumption.

Table 14. Time consumption of different models.

Dataset6	LR	DT	SVM	RF	XGBoost	LightGBM	CatBoost	CNN	LSTM
Time(s)	53.05	21.32	278.49	9.19	157.52	10.91	54.97	10449.32	10555.80

7. Conclusions

In this paper, we design and implement a system called VFE for obtaining and recording essential characteristics of X.509 certificates. With the help of the VFE, we extract a large number of features for model training. Furthermore, we train different types of models to distinguish between malicious and benign certificates. All the models have a relatively high score of validation accuracy and testing accuracy, which indicates the robustness of the VFE. In addition, the average testing accuracy of different models in all datasets is 92.7%, and the validation accuracy of different models in Dataset6 is 93.8%, which indicates the features extracted by the VFE are essential and crucial. Analyzing the five models' top 10 important features, we find some important common features vital for detecting malicious certificates. The ensemble learning models have higher average testing accuracy and lower average standard deviation of testing accuracy than classical and deep models, which indicate ensemble models are the most stable and efficient models. Furthermore, ensemble models reach an average testing accuracy of 95.9%. What is more, we obtain an SVM-based detection model with a testing accuracy of 98.2%, which is the highest accuracy.

Author Contributions: Conceptualization, Z.Z. and C.G.; methodology, J.L. and Z.Z.; software, J.L.; validation, J.L.; formal analysis, J.L., Z.Z. and C.G.; resources, Z.Z. and C.G.; data curation, J.L.; original draft preparation, J.L.; revise and editing, J.L.; visualization, J.L.; supervision, Z.Z. and C.G.; project administration, C.G.; funding acquisition, Z.Z. and C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (Grant No. 2018YFB0804703).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets used in this study are publicly available on our website (<https://github.com/fight-think/features-extraction>).

Conflicts of Interest: The authors have no conflict of interest concerning this manuscript.

Appendix A. Detail of CNN-Based and LSTM-Based Models

We construct CNN-based and LSTM-based models with the help of Pytorch. The details of the CNN-based model are illustrated in Figure A1. The details of the LSTM-based model are illustrated in Figure A2. In order to make it easier to reproduce this work, we use functions in Pytorch to illustrate the hidden layers of each deep learning model.

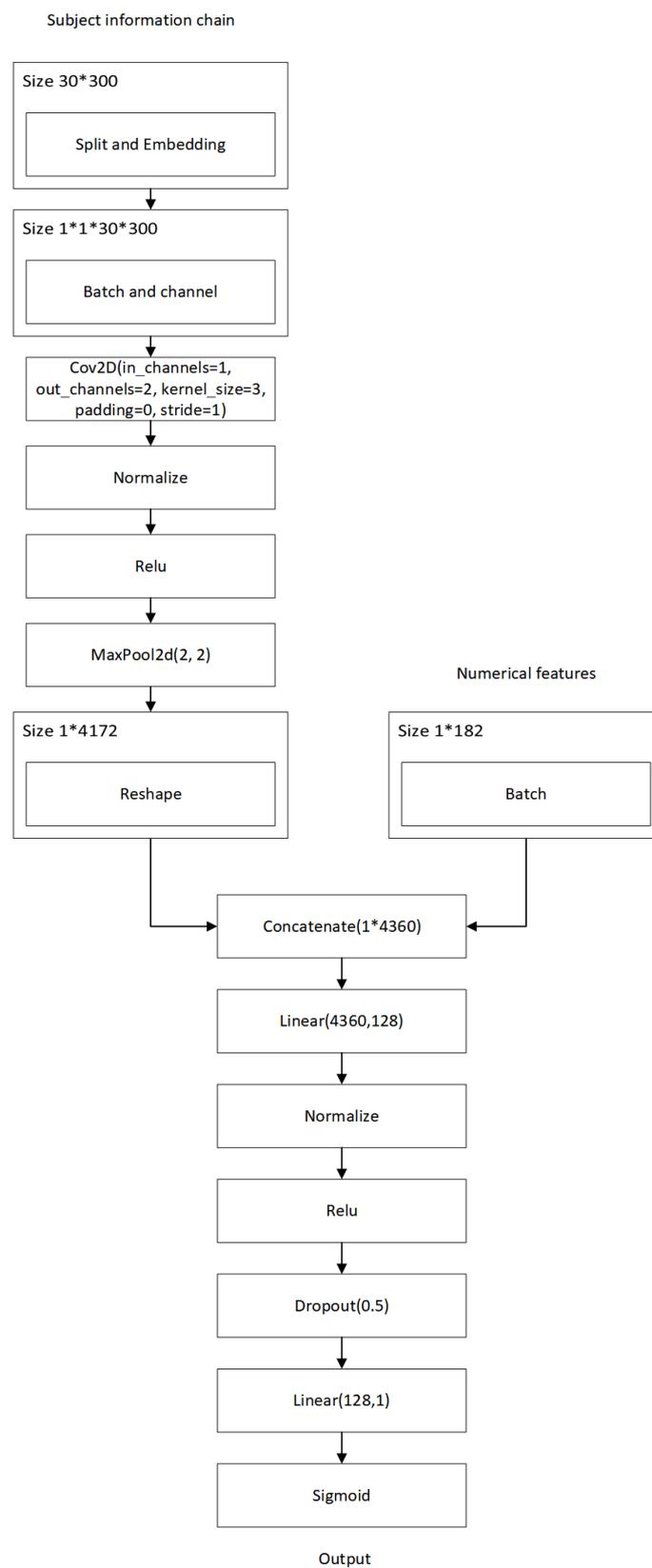


Figure A1. Details of the CNN-based model.

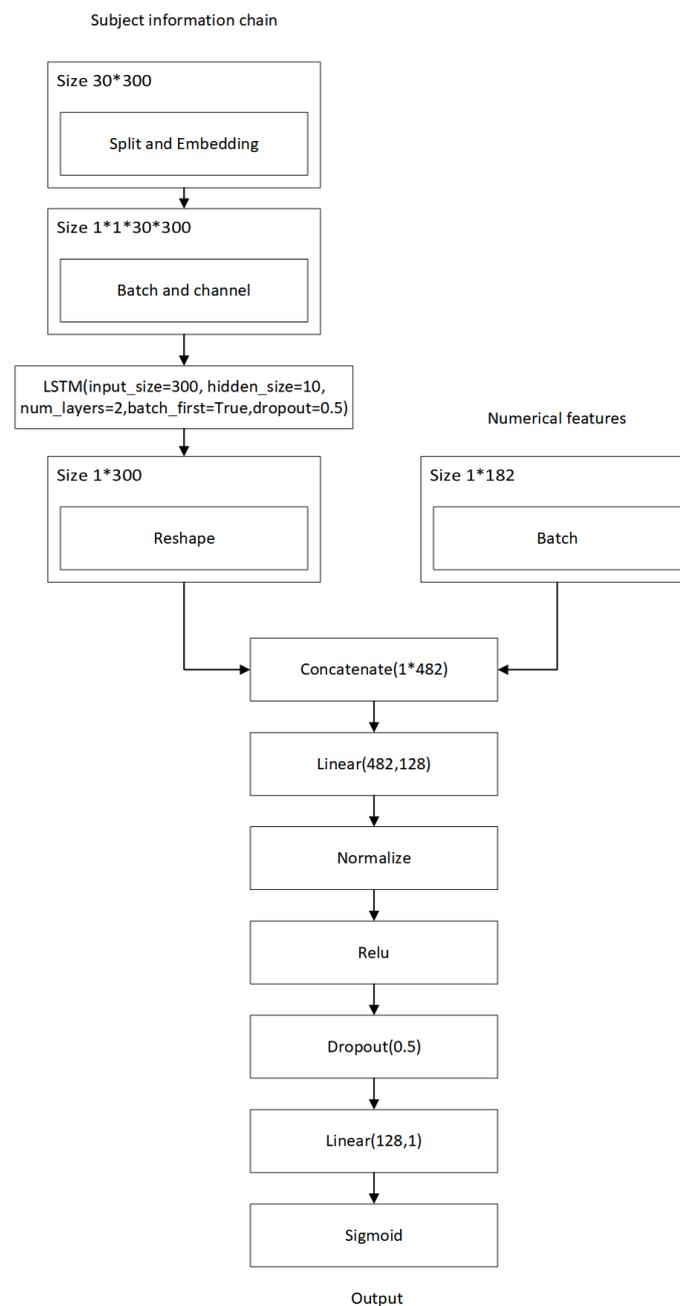


Figure A2. Details of the LSTM-based model.

References

- Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008. Available online: <https://www.rfc-editor.org/info/rfc5280> (accessed on 27 February 2021). [CrossRef]
- Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Han, L.; Raza, U. Malicious SSL certificate detection: A step towards advanced persistent threat defence. In Proceedings of the International Conference on Future Networks and Distributed Systems, Cambridge, UK, 19–20 July 2017.
- Mishari, M.A.; De Cristofaro, E.; Defrawy, K.E.; Tsudik, G. Harvesting SSL certificate data to identify Web-fraud. *arXiv* **2009**, arXiv:0909.3688.
- Dong, Z.; Kane, K.; Camp, L.J. Detection of rogue certificates from trusted certificate authorities using deep neural networks. *ACM Trans. Priv. Secur.* **2016**, *19*, 1–31. [CrossRef]
- Torroledo, I.; Camacho, L.D.; Bahnsen, A.C. Hunting malicious TLS certificates with deep neural networks. In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, Toronto, ON, Canada, 19 October 2018; pp. 64–73.

6. Fasllija, E.; Enişer, H.F.; Prünster, B. Phish-Hook: Detecting Phishing Certificates Using Certificate Transparency Logs. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Orlando, FL, USA, 23–25 October 2019; pp. 320–334.
7. Gu, X.; Gu, X. On the detection of fake certificates via attribute correlation. *Entropy* **2015**, *17*, 3806–3837. [[CrossRef](#)]
8. Dong, Z.; Kapadia, A.; Blythe, J.; Camp, L.J. Beyond the lock icon: Real-time detection of phishing websites using public key certificates. In Proceedings of the 2015 APWG Symposium on Electronic Crime Research (eCrime), Barcelona, Spain, 26–29 May 2015; pp. 1–12.
9. Hutchinson, S.; Zhang, Z.; Liu, Q. Detecting phishing websites with random forest. In Proceedings of the International Conference on Machine Learning and Intelligent Communications, Hangzhou, China, 6–8 July 2018; pp. 470–479.
10. Kulkarni, A.; Brown, L.L. Phishing Websites Detection using Machine Learning. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 6. [[CrossRef](#)]
11. Kumar, D.; Wang, Z.; Hyder, M.; Dickinson, J.; Beck, G.; Adrian, D.; Mason, J.; Durumeric, Z.; Halderman, J.A.; Bailey, M. Tracking certificate misissuance in the wild. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), Francisco, CA, USA, 21–23 May 2018; pp. 785–798.
12. Drury, V.; Meyer, U. Certified phishing: Taking a look at public key certificates of phishing websites. In Proceedings of the 15th Symposium on Usable Privacy and Security (SOUPS'19), Santa Clara, CA, USA, 11–13 August 2019; pp. 211–223.
13. Fadai, T.; Schrittwieser, S.; Kieseberg, P.; Mulazzani, M. Trust me, I'm a Root CA! Analyzing SSL Root CAs in Modern Browsers and Operating Systems. In Proceedings of the 2015 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–28 August 2015; pp. 174–179.
14. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [[CrossRef](#)]
15. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.
16. Severyn, A.; Moschitti, A. Learning to rank short text pairs with convolutional deep neural networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 373–382.
17. Akhawe, D.; Amann, B.; Vallentin, M.; Sommer, R. Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 59–70.
18. Cramer, J.S. The Origins of Logistic Regression. *Tinbergen Inst. Discuss. Pap.* **2002**, *119*, 167–178. [[CrossRef](#)]
19. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv* **2018**, arXiv:1811.03378.
20. Evgeniou, T.; Pontil, M. Support vector machines: Theory and applications. In *Advanced Course on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 249–257.
21. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [[CrossRef](#)]
22. Dietterich, T.G. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 2002; Volume 2, pp. 110–125.
23. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
24. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
25. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3146–3154.
26. Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: Unbiased boosting with categorical features. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 6638–6648.
27. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [[CrossRef](#)]
28. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
30. Dong, Z.; Kapadia, A.; Camp, L.J. Pinning & binning: Real time classification of certificates. In Proceedings of the 29th Annual Computer Security Applications Conference, New Orleans, LA, USA, 9–13 December 2013; poster presentation.