DFedForest: Decentralized Federated Forest

Lucas Airam C. de Souza, Gabriel Antonio F. Rebello, Gustavo F. Camilo,
Lucas C. B. Guimarães, Otto Carlos M. B. Duarte

Grupo de Teleinformática e Automação (GTA) - COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, Brasil

***Abstract*—The effectiveness of machine learning systems depends heavily on the relevance of the training data. Usually, the collected data is sensitive and private because it comes from devices and sensors used in people's daily lives. The General Data Protection Regulation (GDPR) in Europe, the California Consumer Privacy Act (CCPA) in California, and China's Cybersecurity Law put the current approach at risk, as it prohibits centralized remote processing of sensitive data collected in a distributed manner. This paper proposes a distributed machine learning system based on local random forest algorithms created with shared decision trees through the blockchain. The results show that the proposed approach equals or exceeds the results obtained with the use of random forests with only local data. Furthermore, the proposal increases the detection of new attacks when the domains have different threat distributions.**

Keywords: Federated-Learning; Network-Security; Blockchain; Data-Privacy;

I. Introduction

Machine learning systems are highly dependent on the quantity and the quality of the data used to train a machine learning model. Therefore, traditional machine learning environments centralize the storage of data from different sources in one domain to train and execute algorithms. Nevertheless, centralized storage presents privacy problems, as it requires "data owners" to trust in the storage provider that centralizes the data and their ability to prevent leaks of private information. Besides, centralization requires high computational power to process the high volume of data generated by data collectors, as the number of connected devices and the amount of information are growing every day. An efficient way to guarantee privacy is to encrypt data. Traditional encryption, however, does not allow data to be processed, which prevents more effective global learning through encrypted data. Recently, homomorphic cryptography was proposed, which provides sum and product operations on encrypted data. Nevertheless, homomorphic encryption is sophisticated and requires a high amount of data for the encrypted data and the cryptographic key [1].

People are increasing their privacy concerns. Moreover, many countries are establishing restricting Laws over private data such as the General Data Protection Regulation (GDPR), in Europe, the California Consumer Privacy Act (CCPA), in the USA, the Cybersecurity Law, in China, and the LGPD, in Brazil. Federated learning proposes to share machine learning models created locally from private samples instead of sending these samples to a centralized authority [2], [3], [4]. The proposal adopts a client-server architecture, in which clients use local data to update a global model and send the result of their computation to the server. Therefore, domains share only the result of local computing, which is less sensitive than private data.

Distributed solutions can solve centralization and privacy concerns, but introduces a challenge: the data trustworthiness. In a distributed environment a local data owners may behave maliciously, sharing incorrect results to spoil the resulting model. Thus, monitoring the behavior of employees is essential to identify malicious participants, and thereby create an accurate final model.

This paper proposes the DFedForest, a federated learning system for the creation of random forests in a distributed way, with the following characteristics: i) creation of simple machine learning models, based on an ensemble of decision trees, more collaboratively, and accurately than a local model; ii) non-transmission of local data, avoiding loss of privacy; iii) transmission of de-identified models, instead of data, to guarantee a certain level of security and avoid complex homomorphic encryption and iv) use blockchain technology to ensure mutual trust and to register the references of local model addresses in a distributed way, preventing malicious participants from harming the accuracy of the model. We evaluate the performance of the distributed machine learning proposal through an intrusion detection prototype. The prototype addresses the problem of detecting attacks through devices infected with botnets. We develop the prototype using the Hyperledger Fabric platform and the scikit-learning library. The experiments use the CTU-13 [5] dataset, which has samples of normal and malicious traffic generated by botnets. The results of the performance evaluation show that the proposed-system accuracy of a random forest model overcomes the results obtained when the learning model is only local and does not consider data from collaborative models. As machine learning algorithms incorporate a large number of data samples in a compact model, our proposed system effectively limits the volume of exchanged information when compared with centralized models.

We organize the rest of the paper as follows. Section II shows the state-of-the-art machine learning systems and techniques for preserving data privacy. Section III presents the proposed architecture, analyzing the security and the computational complexity of the system. Section IV details the prototype development, in addition to an analysis of the obtained results. Finally, Section V concludes the article and presents future directions.

II. Related Works

The collection of all kinds of data from different places is crucial for the success of several modern applications related to smart cities, smart buildings, electronic health, among others. This data is collected, sent to large data centers, and processed by machine learning algorithms. Nevertheless, growing privacy concerns and frequent data leakage scandals are threatening this thriving practice. Recently, several works investigate distributed machine learning and solutions that keep data at its source instead of sending it to large centralized data centers. Giacomelli *et al.* propose the creation of random forests in a distributed way [6]. Vaidya *et al.* [7] study how to create and share models based on random decision trees [8]. In both papers, the authors guarantee the privacy and security of the resulting models using encryption techniques, which are complex solutions and introduces latency. The use of homomorphic [6] or differential [9] encryption ensures data privacy in distributed environments while allowing simple sum and product operations. Nevertheless, homomorphic cryptography generates large cryptographic keys [1] and computational overhead. Besides, differential privacy makes it difficult to create accurate models due to the introduction of noise in the data. The Federated Forest [10] is a federated learning algorithm that centralizes the management of the model building and the classification of samples on a server. Gossip Learning is a decentralized federated-learning proposal through gossip with known collaborators [11]. The proposal presents better classification results than those obtained by Google [2]. The proposal, however, is susceptible to the Sybil attack [12], as it allows an attacker to create several false profiles to share incorrect results and compromise the final model.

Most distributed learning proposals neglect security and assume that all participants are honest. There is, however, a possibility of attacks by malicious collaborators that intend to damage the final model. One way to mitigate this type of attack is by using a blockchain [13]. The blockchain technology guarantees mutual trust in distributed environments, as proven by several works, such as in smart cities [14], IoT access control [15], [16], [17], databases [18], education [19], network slices [20], smart grids [21], and smart vehicles [22].

Li *et al.* propose using a blockchain to implement distributed federated learning [23]. The blockchain stores the global model updated by a set of domains. Although the learning takes place in a distributed way, the network nodes hold a global model, so a malicious participant can exploit the model to find loopholes and attack other nodes. Besides, the proposal requires several consensus rounds to agree on a global model, increasing the creation time of the final model [2]. FLChain [24] is a federated learning framework that uses of blockchain to create machine learning models. The blockchain audits and tracks the malicious behavior of the participants, stores the model, and pays honest participants for their contribution through a smart contract. Bao *et al.*, however, use only gradient-based algorithms in the proposal, and there are multiple rounds of communication to establish the final model.

Intrusion detection is a research area that studies threat mitigation [25]. Mitigating network threats in IoT environments is crucial, as these environments are usually composed of devices and sensors with limited computational resources. Due to the low processing power, an attacker can easily compromise devices and sensors to be part of a botnet [26], [27]. Thus, a massive botnet can initiate a Distributed Denial of Service (DDoS) [28] attacks.

Kaygusuz *et al.* [29] and Guimarães *et al.* [30] propose two platforms for intrusion detection and classification of data. However, few proposals adopt privacy-preserving measures of the data used for training models capable of detecting attacks [31].

Unlike other works, DFedForest allows the safe sharing of complete non-encrypted decision tree models for local random forest creation. The system protects the user's privacy by de-identifying the local training samples used in the models without introducing latency due to message exchanges for classification or use of encryption. Also, to detect intrusion threats, participants test the decision tree models before aggregating in the local forest, which allows them to identify malicious behavior and incorrect models.

III. The DFedForest System

DFedForest[1] is a distributed system that creates random forests in a federated manner across domains. The main goal of the system is to build an efficient forest model while keeping data private to avoid sensitive information leakage. Thus, we ensure that each domain independently builds tree models using only its private data. We leverage the parallel nature of the random forest algorithm [32] to create a decentralized forest in which domains share trees and receive trees from other domains. By sharing tree models instead of data, the system can achieve high efficiency while preserving data privacy.

A. The Proposed Architecture

The domains in our system use the bagging[2] method [33] to create the decision tree models and classify samples. The bagging method comprises two phases: bootstrap and aggregating. The bootstrap phase consists of generating equally-sized datasets from the original training dataset through random

[1] The implementation is available at https://github.com/GTA-UFRJ-team/DFedForest

[2] Bagging is an acronym for **b**ootstrap **agg**regat**ing**.

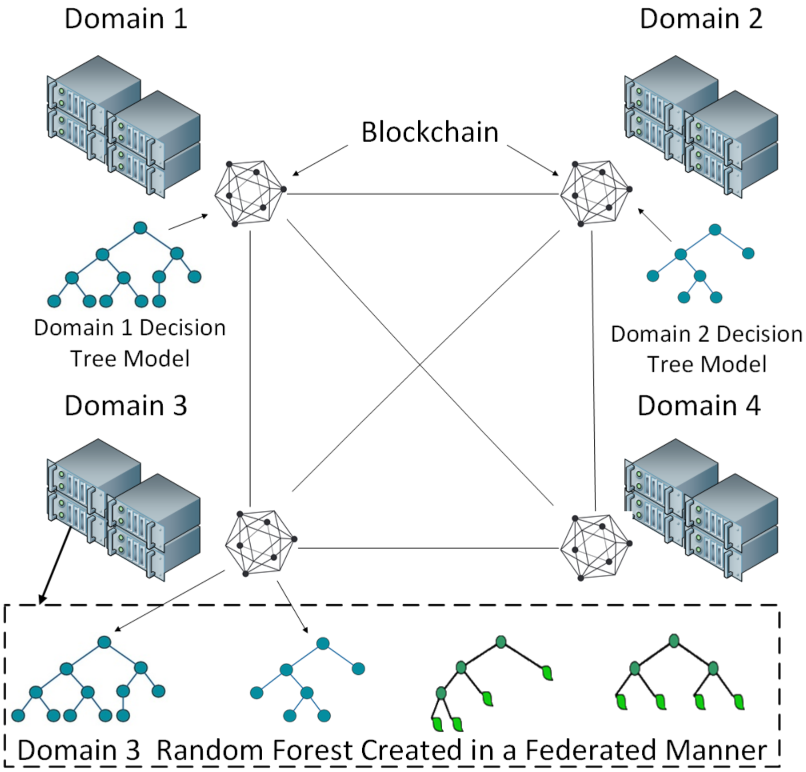
Domain 1
Domain 2
Blockchain
Domain 1 Decision Tree Model
Domain 2 Decision Tree Model
Domain 3
Domain 4
Domain 3 Random Forest Created in a Federated Manner

Fig. 1: Proposed system architecture. Domains 1 and 2 create models of decision trees and share references through the blockchain with publication transactions. Domain 3 issues a query transaction for the blockchain, receives the created models, selects, and evolves the local forest.

sampling. Then, the method trains a machine learning model from each sampled dataset. The goal is to build learning models with different structures that present different views when classifying new samples. In the aggregating phase, the method uses all different model structures of each local model to discover the correct class of a new sample. Each machine learning model classifies the sample, and the final result is the statistical mode of all classifications. This way, the method can generalize the behavior of new samples while minimizing variance.

The proposed architecture of the system includes four main components: the domains, the decision tree models, the blockchain, and the local random forest created in a federated manner. The domains create the tree models through bootstrap, share them through the blockchain, and query new decision trees to improve the local random forest. Before adding trees to the local forest, the domain uses part of its dataset to validate the new models. If the validation result is higher than a predefined threshold, the domain adds the new model to the local forest. The final federated random forest in each domain includes decision trees created locally and by other domains.

Figure 1 shows the proposed architecture in a scenario with four domains. Domains 1 and 2 capture traffic from their networks to create decision trees. After creating the trees and storing them in a server, the domains share a reference, the Uniform Resource Locator (URL), to the model into the blockchain. Domain 3 queries the blockchain for new decision tree model references and downloads the models from the storage server. After selecting the best models, Domain 3 adds the trees of Domains 1 and 2 to its local forest, improving its efficiency.

Figure 2 displays the UML sequence of the system. The left part before the storage server in Figure 2 covers the steps for the publication of new models. A domain captures traffic from its network and extracts its features to creates a dataset. We assume that domains have similar traffic distributions, and all domains extract the same features from network data. The proposed architecture is agnostic to feature extraction methods and data labeling. The domain uses the dataset to create decision tree models, and it shares the reference of the models by publishing transactions in the blockchain. The size of the decision tree models depends on the total number of nodes in the tree. Thus, the system limits the depth of the trees created to reduce the size of the stored models and to avoid overfitting the training dataset.

The right part after the storage server in Figure 2 covers the process of improving the local forest with trees created in other domains. Domains query the blockchain to acquire models from other domains. The blockchain only stores the

URL reference of the decision tree models to avoid overloading of the blockchain size. The domain issues Hypertext Transfer Protocol (HTTP) requests to the referenced storage server after obtaining the storage location of the new models. The server responds to the request with the requested tree models. Upon receiving the response from the storage server, the domain server validates the models obtained to measure accuracy and precision in part of its dataset. The domain administrator defines the criteria for stopping the growth of the local random forest and the thresholds of metrics in the validation.

Storing decision tree references in the blockchain provides auditability to all publishing operations in the proposed system. Thus, if one domain publishes incorrect models, the other domains can easily identify which domain acted maliciously and punish its behavior. Equation 1 presents the structure of a transaction that publishes a model reference into the blockchain:

$$TX_{pub} = [TX_{ID}|A_{ref}|Org_{src}], \tag{1}$$

where TX_{ID} is the transaction identifier, A_{ref} is the storage-location address reference of the decision-tree model, and Org_{src} is the signature of the domain that published the model reference. The system sequentially appends blocks that contain all transactions published in the network. The block header contains information such as the hash of the block content and the hash of the previous block, which provides integrity and total ordering of transactions.

B. Security Analysis

The proposal admits the following malicious domain actions: (i) share incorrect models, (ii) estimate information on forests in other domains, (iii) explore shared models in the blockchain, and (iv) issue empty transactions to produce a denial of service in the system.

The domains test the decision tree models obtained through the blockchain with local data before adding to the random forest. Since the data used for the test is unknown to trained models, this method is similar to the out-of-bag method [34]. Thereby, the system efficiently detects incorrect models, avoiding damage to the final random forest models by an attacker. The domains can identify models, which present low classification metrics, and track the malicious participant due to the immutability and non-repudiation properties of the blockchain. Since the blockchain is permissioned, the credentials of a malicious participant can be revoked by honest members of the network, if most domains agree.

Domain administrators decide which decision tree models of the blockchain will be added to the forest to increase performance, without disclosing the final local result. Therefore, the attacker is unaware of the federated random forest of each domain. Even if a domain queries a tree, the model can fail in the local test for aggregation to the forest. As the test data is kept confidential, it is not easy to estimate which trees were selected. Also, the attacker is unaware of the size of the forests, and the more trees shared in the blockchain, the more difficult it is to discover a domain's federated random forest.

Network participants can access the complete decision-tree models. Therefore, the decision-tree structure excludes private information that can identify a user of the network. Thus, numerical features define the tree thresholds division for each node.

Another malicious behavior is the denial of service through the blockchain network. A malicious domain can issue several valid transactions to flood honest network nodes. The attack, however, is easily identifiable, since the domains sign all transactions and, the blockchain is permissioned. Therefore, the system revokes the attacker credentials and removes him/her from the network.

C. System Complexity Analysis

The training time complexity of the system includes the complexity of three main phases: i) creating decision trees inside domains, ii) storing the models in a server and publishing model references into the blockchain, and iii) reading the references from the blockchain and querying the storage server for the models.

The complexity of creating a single decision tree in our system is the same as in the CART algorithm[3] [35], which is $O(m.n_t.log(n_t))$, where n_t is the number of training samples and m is the number of features in the dataset. Nevertheless, we can simplify the tree creation complexity to $O(n_t.log(n_t))$ because m is limited to four features in this work. The complexity for creating h trees sequentially is $O(h.n_t.log(n_t))$. It can, however, be improved to $O(n_t.log(n_t))$ because each domain can concurrently create its trees if it uses a multiprocessor cluster [32].

Uploading the decision tree model to a storage server requires $O(b)$ complexity, where b is the model size in bytes. The size of the decision tree models in our system, however, is limited to avoid overfitting in the training phase and to save storage space. Thus, we may simplify the complexity to $O(1)$. We also assume each domain can upload its tree concurrently. Publishing a tree requires issuing a transaction that contains a reference to the tree model in the storage server. Hence, the model size does not impact the computational complexity, and we may safely assume that publishing h trees incurs writing $O(h)$ transactions into the blockchain. Nonetheless, we cannot assume concurrency for publishing trees because transactions in the blockchain must be ordered sequentially.

We bound the complexity of querying models in two phases: querying the blockchain for model references and downloading the models from the storage server. In the first phase, querying the blockchain for h trees presents $O(h)$ complexity if the blockchain platform stores transactions in a global hash table, or $O(h.t)$ if we need to search across all t transactions in the blockchain. In the second phase, downloading the model from the server presents $O(1)$ complexity, as in

[3] Classification and Regression Trees (CART) is a state-of-the-art algorithm for building decision trees.

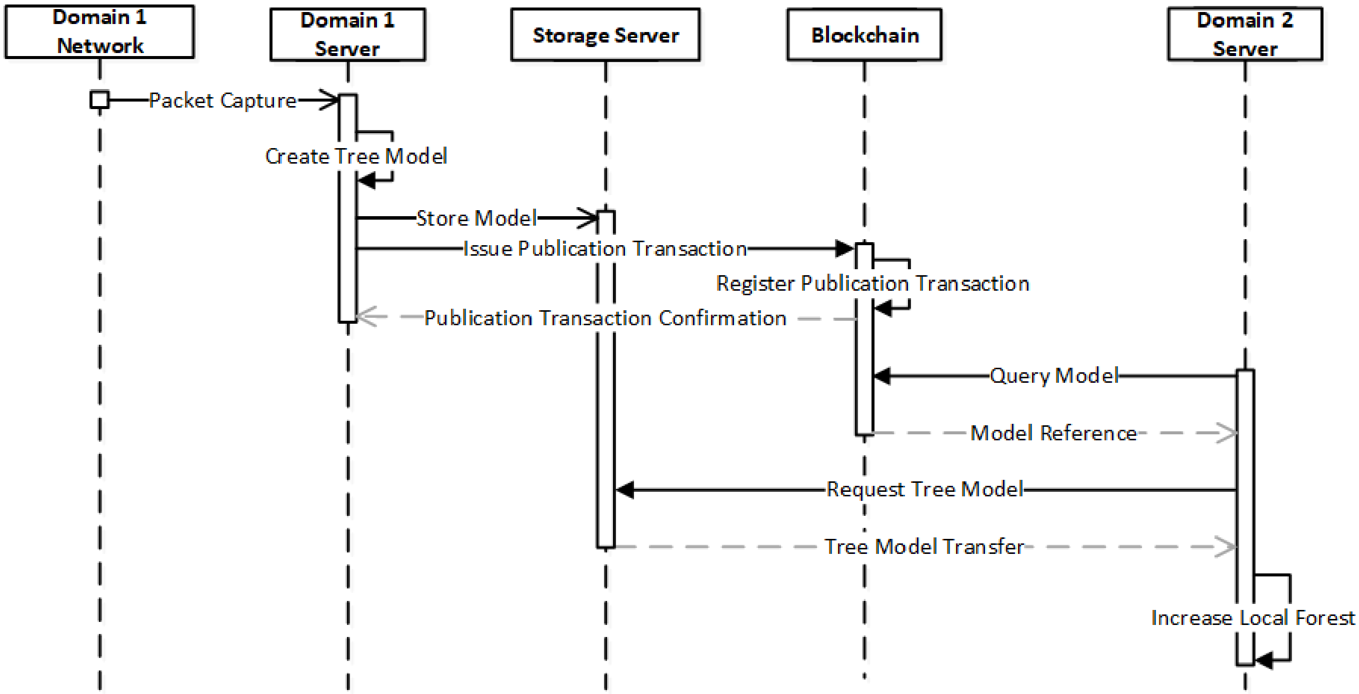

Fig. 2: UML sequence diagram of publishing and querying decision tree models.

the uploading phase. Thus, the time complexity of receiving h decision tree models is $O(h)$ in the best-case scenario or $O(h.t)$ in the worst-case scenario.

Assuming the number of trees h is always less than the number of training samples n_t, the overall best-case time complexity of building a decentralized forest is $O(n_t.log(n) + h + h) = O(n_t.log(n_t))$. Conversely, the worst-case scenario complexity is $O(n_t.log(n_t) + h + h.t) = O(max(n_t.log(n_t), h.t))$. Since most blockchain-based platforms implement hash tables to improve efficiency [36], the results show that despite building a forest, our system incurs no overhead in training time complexity in comparison with CART.

Once the building of the decentralized forest is complete, the forest in each domain classifies new samples like random forest. In a random forest with H trees, the predicted class $\hat{y}$ of a sample x is given by [34]:

$$\hat{y} = f(x) = arg \max_{y \in Y} \sum_{j=1}^{H} I(y = h_j(x)), \qquad (2)$$

where $h_j(x)$ returns the predicted class of x by tree h_j. The term $I(.)$ is the indicator function. The set Y represents the existing classes, which in our work are binary: 0 for normal flows and 1 for malicious flows. If there exists a tie between the predictions of the two classes, each domain applies its tiebreaker rule. The j-th decision tree h_j classifies the sample with $O(log(d))$ time complexity, where d is the depth of the tree. Therefore, the classification complexity of the federated forest is the same as the traditional random forest, given by $O(H.n_c.log(d))$, where H is the number of trees in the forest and n_c o number of samples to be classified.

The overall time complexity of the system is the sum of the training complexity and classification complexity. Nevertheless, the training phase occurs only once and the classification phase occurs for as long as new samples reach the system. Hence, we can assume that eventually $n_c > n_t$ and the overall time complexity is bound by the classification complexity $O(H.n_c.log(d))$.

IV. Prototype Development and Results

The distributed environment was developed with the open-source platform Hyperledger Fabric 2.0 [36] for the blockchain, with virtualized nodes through Docker v19.03.5 containers[4]. The blockchain uses the Raft consensus protocol, and each block has 100 transactions, as used in previous work on the performance evaluation of the Hyperledger Fabric [37] platform. Hyperledger Fabric is a free platform, easily programmable, and without any currency or cost associated with the implementation of permissioned blockchains for organizations. The organizational aspect of Hyperledger meets the scenario of the multi-domain proposal, in which companies share decision tree models. The blockchain has two types of nodes: orderers and peers. Nodes are associated with network organizations, which represent domains. Peers are responsible for interacting with the blockchain and issuing, validating, and verifying transactions. The orderers verify the signatures of the peers, establish the order of transactions within blocks,

[4] Available at https://docs.docker.com /

and publishing blocks to the network. The system considers transactions with unknown or revoked signatures to be invalid. A prototype of the system was developed in Python v2.7.13 using the scikit-learn v0.20.4 library[5] for the maintenance of partial decision tree models. We perform the experiments on an Intel i7-8700 CPU 3.20 GHz server with six processing cores and 32 GB of RAM, and present the average values with 98% confidence intervals.

The prototype uses the CTU-13 [5] dataset, which contains samples of normal and malicious traffic generated with machines infected by botnets. The choice of this dataset allows the assessment of system performance using a current problem that affects thousands of vulnerable devices on computer networks. The CTU-13 dataset has 13 scenarios described in Table I, with 14 features extracted from the packet header. Eight of these identify flows, and two others, ToS destination and ToS origin, are null in the entire dataset. The domains exclude the features that associate a flow with a user[6] to de-identify the data owner. Then, the tree structure contains only information about the flow distributions. This deidentification replaces the complex encryptions because the proposed system shares the decision tree models. Therefore, for training and tests, four features are used: number of packets, number of bytes sent by the source, total bytes of the flow, and its duration.

Mantovani *et al.* claim that hyperparameter optimization does not improve too much decision-tree performance when compared to default values [39]. These algorithms also converge quickly. Thus, we perform four grid searches to adjust the hyperparameters of the number of trees and the maximum depth of the random forest. The other hyperparameters remain at their default values. The first search used the values 10, 100, 500 for the number of trees and 5, 30, 100, 1000 for their depth, with the best results being 100 trees with depth 30. The results of the first search show that the metrics values present a low variance after 100 trees, as verified in other previous work [40]. Thus, we fix the number of trees equal to 100, and we restrict the grid search to the tree depth, searching for smaller models with the same levels of accuracy and precision. We divide the search space into values close to the best result of each previous search, with 23 being the best value found after tuning. The following experiments, both for the random forest and DFedForest, use the hyperparameters found. Experiment 1 aims to verify and compare the performance of the DFedForest system and the random forest with local data on the network. Experiment 1 combines all 13 scenarios in the Table I to produce a balanced dataset, balancing the number of normal and attack flows. Then, this complete dataset with the 13 scenarios is divided into distinct and balanced subsets of the complete dataset using random samples to obtain the subset of each domain. We consider two cases for the traditional approach: (i) domains can access the complete dataset to create the random forest model, without maintaining data privacy, and (ii) domains create the random forest with their data. The forest in case (iii) is our proposal, the DFedForest system, in which each domain creates the same number of trees. To compare the results, the number of trees is equal to 128. The results shown in Figure 3 correspond to a Domain dataset 1 division of 70 to train and 30% to test, for a fair comparison.

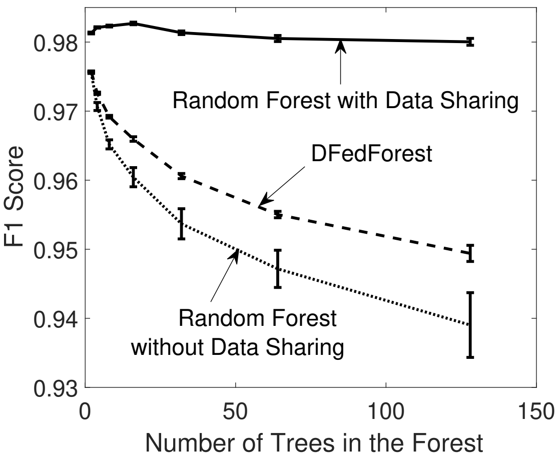

Fig. 3: Comparison of F1-measure for different quantities of domains with equal parts of the dataset. The lines illustrate the results of the classification approaches: (i) random forest with data sharing of the domains for training the models; (ii) random forest trained with data only from the domain; and (iii) random forest created with the DFedForest system.

The results of Experiment 1 show that case (i) produces the best result in all scenarios when the random forest has access to the domain-training sets. However, as this solution is not feasible to maintain data privacy, domains opt for measures that guarantee data privacy, as in cases (ii) and (iii). DFedForest is superior to the random forest without data sharing in most cases, and the more the data is scattered among the domains, the higher the advantage obtained. The results obtained are compatible with those found by Giacomelli *et al.* [6].

Experiment 2 aims to evaluate the effect of adding trees from two different domains to detect new threats. To obtain different types of threat in the two domains we accept the recommendations of the authors of the dataset [5]: Domain 1 combines scenarios 1, 2, 6, 8, and 9 and Domain 2 combines scenarios 3, 4, 5, 7, 10, 11, 12, and 13. We divide each dataset in 30% of data for testing and 70% for training. Figure 4 shows the accuracy results in the classification of Domain 1 in relation to the test data of Domain 2. Domain 1 has 64 trees created with its dataset, represented in Figure 4 by the solid line, and adds trees from Domain 2, represented by the

[5] Scikit-learn [38] is a well-documented open-source library for creating machine learning models, which has a large number of developers. Available at https://scikit-learn.org/

[6] Source IP, Source port, Destination IP, Destination port, application layer protocol, capture timestamp, flow direction, and flow state.

Scenario	Normal Flows	Malicious Flows	Botnet	Infected Nodes
1	30387	4959	Neris	1
2	20941	9120	Neris	1
3	116887	26822	Rbot	1
4	25268	1768	Rbot	1
5	4679	901	Virut	1
6	7494	4630	Menti	1
7	1677	63	Sogou	1
8	72822	6126	Murio	1
9	43340	184979	Neris	10
10	15847	106352	Rbot	10
11	2718	8164	Rbot	3
12	7628	2168	NSIS.ay	3
13	31939	39993	Virut	1

TABLE I: Description of samples and types of botnets of the scenarios present in the CTU-13 dataset.

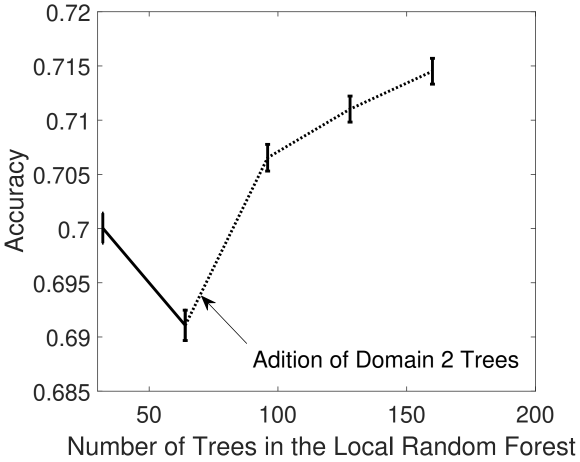

Fig. 4: Accuracy impact of adding Domain 2 trees on Domain 1 random forest when classifying the Domain 2 test set. The solid line represents the DFedForest classification result on Domain 1 with trees created in the domain itself. The dotted line displays the results after adding trees from Domain 2.

dotted line. The addition of models created in Domain 2 to the forest of Domain 1 has a positive impact of up to 3% increase in accuracy when classifying unknown threats present in the Domain 2 test set, a result represented by the dotted line in Figure 4. As the tuning of hyperparameters shows that the size of the forest does not impact the classification metrics after 100 trees, it can be said that the gain in accuracy was due to the DFedForest system. Model sharing improves the detection of threats unknown by Domain 1.

Experiment 3 aims to verify the access time to the blockchain for sharing the references of the decision tree models. Figure 5 displays the time required for issuing different amounts of publishing transactions for a domain. As discussed in Section III-C, the time to issue t models varies linearly with the number of shared models, behavior also observed concerning new models. After building the random forest according to the domain's policies, there is no exchange of messages for the classification of new samples. This feature is crucial for the system's application to real-time classification. Other works that require message exchange or the use of encrypted-data result in high latency, requiring minutes to obtain the class of new samples [6], [10], [8].

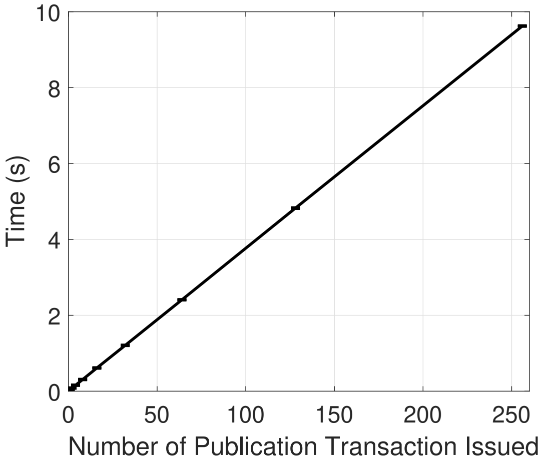

Fig. 5: Asymptotic behavior for issuing publishing transactions that reference decision tree models stored outside the blockchain.

V. Conclusion

The proposed DFedForest system classifies network traffic without requiring overhead and processing and exchanged messages when compared to the federated learning model that uses homomorphic encryption and requires forward of new samples for classification.

The system has low computational complexity, asymptotically described by $O(H.n_c.log(d))$, the same as the traditional random forest with H trees with depth d to classify n_c samples. DFedForest performs better than the approach with local data from the same domain. This improvement is due to the sharing of local models between all domains. We proposed a system that uses a blockchain to share the reference to obtain in a distributed way local models. The immutability and transparency properties allow participants to audit malicious domains that intend to harm the final models. The blockchain does not affect the performance of the system, as the prototype developed requires half a second to share a model and varies linearly with the number of shared models. For future work, we intend to integrate DFedForest with a distributed file system, such as the InterPlanetary File System (IPFS), and to evaluate the proposal with different datasets.

VI. Acknowledgment

This work was financed by CNPq, CAPES, FAPERJ and FAPESP (18/23292-0, 2015/24485-9, and 2014/50937-1).

References

[1] P. S. Pisa, M. Abdalla, and O. C. M. B. Duarte, "Somewhat homomorphic encryption scheme for arithmetic operations on large integers," in *IEEE GIIS*, 2012, pp. 1–8.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[3] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Transactions on Mobile Computing*, 2020.

[4] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, 2020.

[5] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[6] I. Giacomelli, S. Jha, R. Kleiman, D. Page, and K. Yoon, "Privacy-preserving collaborative prediction using random forests," *AMIA Summits on Translational Science Proceedings*, vol. 2019, p. 248, 2019.

[7] J. Vaidya, B. Shafiq, W. Fan, D. Mehmood, and D. Lorenzi, "A random decision tree framework for privacy-preserving data mining," *IEEE TDSC*, vol. 11, no. 5, pp. 399–411, 2013.

[8] W. Fan, H. Wang, P. S. Yu, and S. Ma, "Is random model better? On its accuracy and efficiency," in *Third IEEE International Conference on Data Mining*, 2003, pp. 51–58.

[9] S. Rana, S. K. Gupta, and S. Venkatesh, "Differentially private random forest with high utility," in *2015 IEEE International Conference on Data Mining*, 2015, pp. 955–960.

[10] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated Forest," *IEEE Transactions on Big Data*, pp. 1–1, 2020.

[11] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP DAIS*. Springer, 2019, pp. 74–90.

[12] J. R. Douceur, "The sybil attack," in *IPTPS*. Springer, 2002, pp. 251–260.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, disponível em https://bitcoin.org/bitcoin.pdf. Last access: 15 July 2020.

[14] R. A. Michelin, A. Dorri, R. C. Lunardi, M. Steger, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "SpeedyChain: A framework for decoupling data from blockchain for smart cities," in *MobiQuitous*, 2018, pp. 145–154.

[15] M. A. Islam and S. Madria, "A permissioned blockchain based access control system for IoT," in *IEEE Blockchain*, 2019, pp. 469–476.

[16] T. H. Kim and J. Lampkins, "SSP: Self-Sovereign Privacy for Internet of Things using blockchain and MPC," in *IEEE Blockchain*, 2019, pp. 411–418.

[17] G. F. Camilo, G. A. F. Rebello, L. A. C. de Souza, and O. C. M. B. Duarte, "AutAvailChain: Automatic and Secure Data Availability through Blockchain," *IEEE GLOBECOM*, 2020.

[18] S. Linoy, H. Mahdikhani, S. Ray, R. Lu, N. Stakhanova, and A. Ghorbani, "Scalable privacy-preserving query processing over Ethereum blockchain," in *IEEE Blockchain*, 2019, pp. 398–404.

[19] L. Liu, M. Han, Y. Zhou, R. M. Parizi, and M. Korayem, "Blockchain-based certification for education, employment, and skill with incentive mechanism," in *Blockchain Cybersecurity, Trust and Privacy*. Springer, 2020, pp. 269–290.

[20] G. A. F. Rebello *et al.*, "Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology," in *IEEE HPSR*, 2019, pp. 1–6.

[21] A. Aderibole *et al.*, "Blockchain Technology for Smart Grids: Decentralized NIST Conceptual Model," *IEEE Access*, vol. 8, pp. 43 177–43 190, 2020.

[22] D. Gabay, K. Akkaya, and M. Cebe, "Privacy-preserving authentication scheme for connected electric vehicles using blockchain and zero knowledge proofs," *IEEE Transactions on Vehicular Technology*, 2020.

[23] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *arXiv preprint arXiv:2004.00773*, 2020.

[24] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLChain: A blockchain for auditable federated learning with trust and incentive," in *BIGCOM*, 2019, pp. 151–159.

[25] M. A. Lopez, D. M. F. Mattos, and O. C. M. B. Duarte, "An elastic intrusion detection system for software networks," *Annals of Telecommunications*, vol. 71, no. 11-12, pp. 595–605, 2016.

[26] V. Bezerra, V. T. da Costa, R. Martins, S. Barbon, R. Miani, and B. B. Zarpelão, "Providing IoT host-based datasets for intrusion detection research," in *SBSeg*, 10 2018, pp. 15–28.

[27] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "Botchase: Graph-based bot detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 15–29, 2020.

[28] M. Pelloso, A. Vergutz, A. Santos, and M. Nogueira, "A self-adaptable system for DDoS attack prediction based on the metastability theory," in *IEEE GLOBECOM*, 2018, pp. 1–6.

[29] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac, "Detection of compromised smart grid devices with machine learning and convolution techniques," in *IEEE ICC*, 2018, pp. 1–6.

[30] L. C. de Brito Guimarães, G. A. F. Rebello, F. S. Fernandes, G. F. Camilo, L. A. C. de Souza, D. C. dos Santos, L. G. C. M. de Oliveira, and O. C. M. B. Duarte, "TeMIA-NT: ThrEat Monitoring and Intelligent data Analytics of Network Traffic," 2020.

[31] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *IEEE ICDCS*, 2019, pp. 756–767.

[32] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for Big Data in a Spark cloud computing environment," *IEEE TPDS*, vol. 28, no. 4, pp. 919–933, 2016.

[33] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[34] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[35] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees. belmont, ca: Wadsworth," *International Group*, vol. 432, pp. 151–166, 1984.

[36] Androulaki *et al.*, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the 13th EuroSys Conference*, 2018, p. 30.

[37] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second," in *IEEE ICBC*, 2019, pp. 455–463.

[38] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. de Carvalho, "Hyper-parameter tuning of a decision tree induction algorithm," in *2016 5th BRACIS*, 2016, pp. 37–42.

[40] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" in *MLDM*. Springer, 2012, pp. 154–168.