Poligraph: Intrusion-Tolerant and Distributed Fake News Detection System

Guohou Shan, Boxin Zhao, James R. Clavin, Haibin Zhang, and Sisi Duan

Abstract—We present Poligraph, an intrusion-tolerant and decentralized fake news detection system. Poligraph aims to address architectural, system, technical, and social challenges of building a practical, long-term fake news detection platform.

We first conduct a case study for fake news detection at authors' institute, showing that machine learning-based reviews are less accurate but timely, while human reviews, in particular, experts reviews, are more accurate but time-consuming. This justifies the need for combining both approaches.

At the core of Poligraph is two-layer consensus allowing seamlessly combining machine learning techniques and human expert determination. We construct the two-layer consensus using Byzantine fault-tolerant (BFT) and asynchronous threshold common coin protocols. We prove the correctness of our system in terms of conventional definitions of security in distributed systems (agreement, total order, and liveness) as well as new review validity (capturing the accuracy of news reviews). We also provide theoretical foundations on parameter selection for our system.

We implement Poligraph and evaluate its performance on Amazon EC2 using a variety of news from online publications and social media. We demonstrate Poligraph achieves throughput of more than 5,000 transactions per second and latency as low as 0.05 second. The throughput of Poligraph is only marginally (4%–7%) slower than that of an unreplicated, single-server implementation. In addition, we conduct a real-world case study for the review of fake and real news among both experts and non-experts, which validates the practicality of our approach.

Index Terms—reliability, fault tolerance, machine learning

1 Introduction

The use of words and semantics to share information in news media and political discourse has considerable power in influencing people's beliefs and opinions [1]. Fake news is defined as intentionally fabricated information that mimics news media content in form but not in organizational process or intent, lacking the news media's editorial norms and processes for ensuring the accuracy and credibility of information [2]. The extensive dissemination of fake news has the potential to significantly impact both the individual and society. It is misleading and potentially harmful to readers when news is disconnected from its original source and context [3]. Unfortunately, a person's ability to distinguish truth from deception has just 54% accuracy on average, or slightly better than random guessing [4]. The majority of U.S. adults (62%) get their news from social media [5], which is the main source of fake news. Currently, fake news about COVID-19 spreads fast and becomes dangerous on social media platforms. Being able to detect fake news has become more important than ever before.

Existing studies in fake news detection can be classified into four categories: experimental studies, research on fake news modeling, machine learning (ML), and deep learning (DL) based detection. In experimental studies, researchers recruit participants to verify the authenticity of the news and assess whether their opinions or beliefs will be influenced by fake news [6]–[8]. Fake news modeling focuses on constructing fake news detection models based on the genesis, evolution, and propagation of a fake news article, by cataloging and tracing its characteristics over time [9], [10]. By extending and utilizing various news-related features, researchers have built and trained different machine learning models to predict the probability that a news article is fake, through a variety of classical data science techniques, i.e., classification, and regression [11], [12]. Besides conventional ML approaches, deep learning-based approaches have recently also been studied for fake news detection [13]–[23]. ML based models leverage algorithms such as logistic regression (LR), support vector machine (SVM), and XGBoost to classify news based on news textual and/or visual features while DL based models leverage neural networks like CNN, LSTM, and adversarial network to identify fake news based on news textual and/or visual information. The majority of the works, however, are ML-based, partly because ML approaches usually achieve lower latency (i.e., and therefore higher throughput from system perspective). Furthermore, several companies are also looking for systematic and publicly available solutions for fake news detection. For instance, Facebook is developing its own cryptocurrency solutions with the potential to eliminate fake news and

- *G. Shan is with Department of Management Information System, Temple University. Co-first author.*
  E-mail: guohou.shan@temple.edu
- *B. Zhao is with Institute for Advanced Study, Tsinghua University. Co-first author.*
- *J. Clavin is with Department of Information Systems, University of Maryland, Baltimore County.*
- *H. Zhang is with Shandong Institute of Blockchain. Corr esponding author.*
  Email: bchainzhang@aliyun.com
- *S. Duan is with Institute for Advanced Study and Beijing National Research Center for Information Science and Technology, Tsinghua University. Corresponding author.*
  Email: duansisi@mail.tsinghua.edu.cn

bots [24].

Despite an impressive amount of work on fake news detection *techniques* as described above, and given that some non-dedicated services that are run by a single provider do exist (e.g., Amazon Mechanical Turk [25]), to the best of our knowledge, there is no public and intrusion-tolerant system for fake news detection ensuring reliability, security, and accuracy.

Besides the basic requirements of being decentralized and defending against arbitrary (Byzantine) failures and malicious attacks, we identify the challenges and requirements of building an intrusion-tolerant fake news detection system.

- **The need for ML and human review integration.** In our work, we focus on ML based approaches. Compared to DL approaches, ML approaches have lower latency and thus incur a higher throughput in the system we build. Furthermore, most existing approaches for fake news detection are ML based ones. The fake news modeling and ML techniques have limitations that are both general to data science and specific to this domain. As is the case with any ML technique, the results depend on the quality of the training and holdout data, and the overall size of the data set being fed into the ML algorithm. Even for those ML fake news algorithms which have been built upon a robust data set, there remains a challenge: any ML result must be reviewed by a human, preferably a specialist in the field, to validate the results. For instance, social media such as Facebook has a long history of using ML to detect fake news and spam. The ML approaches, however, are shown to have high inaccuracy that can only be addressed by human moderators [30].
- **Reproducibility.** It is desirable to guarantee that news prediction is deterministic and can be verified by anyone over time. Many existing ML systems use a proprietary data set and therefore the prediction is not reproducible. Even if the data set is open-source, one would have to trust the algorithms and the system running the algorithms. Therefore, building a reliable and secure system for an ML system can enhance the reproducibility of the ML model and the data set.
- **On-line vs. off-line fake news detection.** Providing real time, on-line prediction is needed for many applications, but the results may be less accurate. An off-line system, however, may provide results that are less responsive but more accurate. While it is desirable to provide efficient trade-offs, no such mechanism exists.
- **Practical state transfer for ML.** While we know how to maintain and transfer state in replicated systems, it is not clear how to deal with state transfer for ML efficiently. The ML state is complex, and a lightweight solution is needed.
- **Toward an evolving, long-term platform.** The accuracy of a fake news detection system may be improved as labeled data grows. In addition, the ML needs to be re-trained periodically for better accuracy. This poses additional requirements and challenges of building such a system.

Our approach and our contribution. We have conducted a news review of both real and fake university news among a group of students and news staff at UMBC. We show that machine learning based reviews are less accurate but timely; we also show that with certain expertise in a certain field, collective reviews among a group of human are able to achieve a high enough accuracy to justify the authenticity of the news. The case study bolsters the intuitive viewpoint that one should combine both machine learning techniques and expert review determination to achieve the best of both worlds.

We design and implement Poligraph, a Byzantine fault-tolerant (BFT) distributed system for fake news detection satisfying *all* system requirements mentioned above. Poligraph ensures integrity and availability, defending against malicious (Byzantine) news feed providers, servers, and human reviewers. Poligraph has a novel system architecture based on a new consensus primitive, *two-layer consensus*, which can combine the news authenticity results from ML models and human reviewers. We show that two-layer consensus can be efficiently built from BFT and asynchronous threshold PRF protocols [31]. We prove the correctness of our protocol and show how to tune parameters to allow parallel reviews securely and efficiently.

A comparison with existing approaches is shown in Table 1. In particular, as the data set maintained by our system grows, its anthropological, forensic, analytic, and data science value increases. In other words, the motivation of our work is not to study the fake news problem itself or enhance the ML approach, but to provide a secure and distributed approach, which can be used as a platform to maintain a growing list of data set. The data set over time will enhance the ML performance.

We evaluate the performance of Poligraph on Amazon EC2 using both political and entertainment news from news websites (e.g., Washington Post) and social media platforms (e.g., Twitter) with news labels from PolitiFact.com and GossipCop.com. Via extensive evaluation of Poligraph, we show it achieves throughput of more than 5,000 transactions per second and has latency as low as 0.05 second. The throughput of Poligraph is only marginally (4%–7%) slower than that of an unreplicated, single-server implementation.

2 Related Work

Fake News Detection. In the past two decades, fake news detection has been investigated extensively [**?**], [6], [9], [11], [18]–[23], [32]–[34]. We classify the past studies into four categories: experimental studies, fake news modeling, ML, and DL based detection. (1) *Experimental studies* utilize behavioral experiments to reveal the cognitive aspects of people towards fake news detection. For example, Pennycook et al. adopted Amazon's Mechanical Turk (AMT) [25] to conduct their experiments on whether prior fake news exposure will affect people's perceived accuracy of fake news. In their research, they confirmed that prior exposure to fake news can increase subsequent perceptions. Specifically, people are more likely to believe fake news articles if they saw those articles before elsewhere, and were even more likely to believe if they saw the article on multiple occasions [6]. (2) *Fake news modeling* approaches focus on building fake news detection models from news characteristics. For instance, Jin et al. constructed a hierarchical propagation model, consisting of

fake news detection system	decentralized	Byzantine failure	human review	ML	reproducibility	growing data set
ML-based online detection [26], [27]	○	○	○	●	○	○
ML detection models [13], [14], [17]	○	○	○	●	○	○
Human determination [28], [29]	○	○	●	○	○	○
Amazon Mechanical Turk [25]	○	○	●	○	○	○
Poligraph (this work)	●	●	●	●	●	●

Table 1: Comparison of fake news detection systems. ○Not supported ●Supported.

event, sub-event, and message networks, to verify the news credibility on two data sets from Sina Weibo, a popular micro-blogging site. After linking the semantic and social associations of a news event, the model achieves higher accuracy and F1-score [35] than baseline methods [34]. (3) *Machine learning based methods* apply different models to enhance the accuracy of fake news detection. Several ML models combine the features from previous studies and propose new features to improve the performance of fake news detection [**?**], [1], [36]. For instance, Jin et al. proposed five new image visual features and seven new image statistical features extracted from news articles to detect fake news. In their experiments, they used SVM, Logistic Regression, KStar, and Random Forest models, utilizing their newly identified features with baseline features from previous studies. They applied their models on the Sina Weibo data set and found their selected features improve fake news detection performance [**?**]. (4) Recently, deep learning based approaches have also been studied for fake news detection [13]–[23]. Specifically, various deep learning models such as CNN, LSTM, Bi-LSTM, adversarial neural network, and even transfer learning are used to identify fake news. For example, Wang et al. designed an event adversarial neural networks by considering both news textual and visual information, where both text-CNN and image-CNN (e.g., VGG-19) are used to extract the textual and visual feature representation [18].

In our work, we integrate ML-based methods and experimental studies (human reviews to be specific) and provide a generic system architecture to build such a system. Compared to DL-based approaches, we study ML based ones in our work, mainly because ML-based approaches have lower latency and the majority of the works are ML-based ones. Our system architecture, however, is generic so one could simply replace ML-based approaches with DL-based ones.

Byzantine Fault Tolerance (BFT). The goal of BFT consensus is that the correct replicas reach a consensus on the order of client requests. Beginning with PBFT [37], an impressive number of efficient BFT protocols are proposed (e.g., [38]–[47]). Numerous efforts have been made to improve the performance of BFT using different approaches [41], [48]–[56]. While most of them work in partially synchronous environments to guarantee liveness, there exist efficient protocols working in completely asynchronous environments [57], [58]. Our two-layer consensus is general and can be based on any of these BFT protocols.

Blockchains. Blockchains can be categorized into two types: permissionless blockchains and permissioned blockchains. A permissionless blockchain allows dynamic membership using Sybil attack resistant mechanisms [59]–[61]. In contrast, permissioned blockchains require that servers know the identities of each other but do not have to trust each other [62]–[65]. BFT is deemed as *the* model for permissioned blockchains [65] and can also be used to improve permissionless blockchains (hybrid blockchains).

Our two-layer consensus divides the roles of nodes into consensus nodes (replicas) and reviewer nodes. The idea of separating roles of nodes originates from early consensus works such as Lamport's Paxos crash fault-tolerant consensus [66]. Several previous efforts also separate the roles in the BFT for various purposes [54], [67]–[71].

Group	Min	Max	Median	Mean	Std
Group 1	0	10	7	6.30	2.53
Group 2	10	10	10	10	0
ML	5	9	6	6.27	1.19

Table 2: Review scores for both human groups and ML.

3 The Need for Combining Machine Learning and Expert Reviews: A Case Study

Intuitively, one would need machine learning for quick reviews and need human experts for more accurate and detailed new reviews. To justify the need for combining ML techniques and human expert reviews, we have conducted a news review at the authors' institute (UMBC). We provide interesting results, bolstering the viewpoint that a combination of ML and expert reviews achieve the best of both worlds.

Specifically, we crawled 50 university news articles from the UMBC's news website. We then use Grover [26] to generate 50 fake news. The 100 news forms a pool of university news.

We conduct news reviews among two groups of people. Group 1 includes only graduate students. Group 2 includes staff from the communications team at UMBC and some of them may have reviewed or edited some news in our news pool. We intentionally chose the two groups of reviewers for our study. For group 1, we aim to validate the conclusion where for news that is not subjective (e.g., university news is usually either fake or authentic), experts can well validate the authenticity of the news. Second, for group 2, we aim to study whether, with some background knowledge, a non-expert can justify the authenticity of the news with decent accuracy. The study for group 2 can also be used to validate our proposed architecture as discussed in great detail later.

For each participant, we randomly select 10 news out of the 100 news pool to review. We have collected 30 valid sets of results. In addition, we also train our LR model using the training data set from PolitiFact.com and use it to predict the authenticity of the 100 news. For the news word embedding, we use doc2vec google news corpus [72] to extract the news features. We include detailed discussion about the implementation of ML model in Sec. 8. Specifically, each time we randomly select 10 samples to obtain the ML results

Approach	Accuracy	Precision	Recall	F1-score	Latency
Non-expert review (Group 1)	0.625	0.663	0.529	0.551	14 min
Expert review (Group 2)	1	1	1	1	3.3 min
Human review (combined)	0.652	0.709	0.580	0.601	7min
ML review	0.637	0.688	0.784	0.732	6.5s
T-stat	2.756**	3.534***	-4.891***	-0.931	NA

Note: *< 0.05; **< 0.01; ***< 0.001.

Table 3: Review results of human reviewers and ML reviews.

and we run the same test 30 times. This matches the test we have done for human reviews. We have also conducted a t-test for comparing the average performance between human and ML reviews. The T-stat is calculated as follows:

$$T - stat = \frac{\bar{X_1} - \bar{X_2}}{\sqrt{1/N({S_1}^2 + {S_2}^2)}} \quad (1)$$

where $\bar{X_1}$, $\bar{X_2}$, ${S_1}^2$, and ${S_2}^2$ represent the average performance (e.g., accuracy) and standard deviations among two samples (human and machine learning), respectively. N reveals the sample size, which is 30 in our situation.

We first report the *score* for the human review for both groups and doc2vec reviews in Table 2. The score refers to the number of correct answers in each experiment. The results for group 2 are highly accurate, in part because the communications team staff are very familiar with the news of the institute. In comparison, the results for students in group 1 are less accurate. The median and mean, however, are well above average. Although none of the student participants is majoring in media or news, the results have shown that, with some familiarity of the news content, people's justification is well above the 54% random guessing for a general audience [4]. In comparison, non-expert reviews are slightly more accurate than ML reviews and less accurate than expert reviews.

We also compare accuracy, precision, recall, f1-score, and evaluation time for both human reviews and ML results, where the first four metrics as commonly used automatic fake news detection performance metrics [11], [12], [36]. Specifically, accuracy is the percentage of correctly detected fake and true news among the total number of news examined; precision is defined as the percentage of total news detected as fake that is indeed fake ones; recall is the ratio of fake news that is correctly identified; and F1-score is a harmonic mean of precision and recall.

As shown in Table 3, human reviews in general achieve much higher performance for all the criteria we have used. The major drawback is the long review latency. Indeed, humans take on average 7 minutes to review each article, which is significantly higher than a few seconds for ML reviews. This is unavoidable in practice since all the data sets used in supervised ML rely on humans to review them. Among the human reviews, the results by experts (group 2) are consistently higher than that by non-experts (group 1), both of which are consistently better than ML.

To conclude, our results have validated the need for a combination of ML and expert reviews. Specifically, the ML results are less accurate but more timely. Human reviews, in particular, expert reviews, are more accurate but might be time-consuming. With little to median expertise, human results can be used as labels to finalize the results of news.

4 System and Threat Model

We consider a generalized distributed review system that is not restricted to fake news detection. The system consists of *clients*, n *servers* (*replicas*), and N *reviewers*. Clients submit *requests* to the system and expect *results* for their requests. The result for a request can be a *temporary result* or a *final result*. A temporary result is an initial assessment of a request by replicas according to the current system state, st. A final result combines the temporary result with reviews collected from reviewers. A client may first receive a temporary result and later receive a final result, or receive a final result directly if the request has already been reviewed by reviewers. In our work, the duration between the temporary result and the final result is determined by the time for human reviews.

Each time a new request that has not been reviewed is submitted, a set of I reviewers will be selected to contribute their reviews for the request. Each reviewer provides a binary result with certain data payload. The selected set is called a *reviewer set*. Let R denote the number of required matching reviews. If R matching reviews are received by servers, the servers combine the temporary server-generated result with the human reviews to form a final result for that request. Client requests and reviews are collectively called *transactions*. Clients, servers, and reviewers are collectively called *nodes*.

Nodes can fail arbitrarily (*Byzantine failures*). We consider a strong adversary which can coordinate Byzantine nodes. We assume f out of n servers may be Byzantine and we require $n > 3f$. We assume F out of N reviewers may be faulty. For simplicity and correctness, we require $N > 2F$ in this paper. Namely, among the N reviewers, the majority of them (correct reviewers) will return the same binary review result (denoting if a request is authentic) for the same request. The corresponding review is referred to as a *correct review*. Faulty reviewers will either be non-responsive or generate biased (incorrect) reviews.

We assume *asynchronous* environments, making no timing assumptions on message processing or transmission delays. In some cases, the system works under *partial synchrony* [73], where synchrony holds after some unknown global stabilization time, but the bounds on communication and processing delays are themselves unknown.

Goals. Our system has the following goals.

- **Agreement**. If any correct replica delivers a transaction m, then every correct replica delivers m.
- **Total Order**. If a correct replica has delivered transactions $m_1, m_2, \cdots, m_s$ and another correct replica has delivered $m'_1, m'_2, \cdots, m'_{s'}$, then $m_i = m'_i$ for $1 \leq i \leq min(s, s')$.
- **Liveness-1**. If a transaction m is submitted to $n-f$ correct replicas, then all correct replicas will eventually deliver

m.

- **Liveness-2: Temporary result**. For each new request that is sent by a client and has never been processed and finalized by replicas, the client will eventually receive a temporary result from replicas.
- **Liveness-3: Final result**. The client will eventually receive a final result from replicas for a request.
- **Validity**. The final result is guaranteed to be correct in spite of faulty (non-responsive and biased) reviewers.

4.1 Fake News Detection

We now restrict the review system to the case of fake news detection. In this system, clients submit news to the servers. Servers run ML models to generate a temporary assessment of whether the news is fake or authentic. Reviewers are experts such as credentialed journalists.

Machine learning models. In this work we use supervised machine learning models. Compared to deep learning models, machine learning models are faster and have low latency in detecting fake news. Among the machine learning models, LR, SVM, random forest, naive bayes, and XGBoost are commonly used. Among them, LR and SVM perform the best with the lowest latency in general, as pointed out by prior works [74]. Thus, we focus on LR as the machine learning model. Each model can be initiated with a number of parameters $a_1, a_2, \cdots, a_k$. For example, for a LR model, the input parameters be penalty (the norm used in the penalization), tol (tolerance for stopping criteria), etc.

The parameters determine the ML state. Each machine learning model consists of two phases: training and prediction. The training phase uses a *labeled data set* to train the ML model and forms a ML state. The prediction phase utilizes trained ML state to make predictions (results) for unlabeled data (news verification requests from clients).

5 Building Blocks

BFT. We use BFT state machine replication (SMR) protocols, where f out of n replicas may fail arbitrarily (Byzantine failures) and a computationally bounded adversary can coordinate faulty replicas. A replica *delivers transactions* submitted by a client. A client can compute a final result to its submitted transaction from the responses it receives from replicas. Correctness of a secure BFT protocol is specified as follows.

- **Agreement**: If any correct replica delivers a transaction m, then every correct replica delivers m.
- **Total Order**: If a correct replica has delivered $m_1, m_2, \cdots, m_s$ and another correct replica has delivered $m'_1, m'_2, \cdots, m'_{s'}$, then $m_i = m'_i$ for $1 \leq i \leq min(s, s')$.
- **Liveness**: If a transaction m is submitted to $n - f$ correct replicas, then all correct replicas will eventually deliver m.

BFT is a key building block for our system. Both efficient asynchronous and partially synchronous BFT protocols exist. If the underlying BFT protocol is asynchronous (resp., partially asynchronous), our protocols are asynchronous (resp., partially asynchronous).

Threshold PRF. We review threshold PRF (e.g., [31]), where a public key is associated with the system and a PRF key is shared among all the servers. A (t, n) threshold PRF scheme Π for a function F consists of the following algorithms (FGen, Eva, Vrf, FCom).

- A probabilistic key generation algorithm FGen takes as input a security parameter l, the number n of total servers, and threshold parameter t, and outputs (pk, vk, sk), where pk is the public key, vk is the verification key, and $sk = (sk_1, \cdots, sk_n)$ is a list of private keys.
- A PRF share evaluation algorithm Eva takes a public key pk, a PRF input m, and a private key sk_i, and outputs a PRF share y_i.
- A deterministic share verification algorithm Vrf takes as input the verification key vk, a PRF input m, and a PRF share y_i, and outputs $b \in \{0, 1\}$.
- A deterministic combining algorithm FCom takes as input the verification key vk, a PRF input m, and a set of t valid PRF shares, and outputs a PRF value y.

We require the threshold PRF value to be unpredictable against an adversary that controls up to $t - 1$ servers. We also require the threshold PRF to be robust in the sense the combined PRF value for m is equal to $\mathsf{F}(m)$.

We use the CKS threshold PRF [31], as described in Figure 1. The protocol is non-interactive, working in asynchronous environments. The CKS threshold PRF is secure under the computational Diffie-Hellman (CDH) assumption in the random oracle model.

Setup. Let G be a group of prime order q with generator g. Let $\mathbb{Z}_q$ be the additive group, integers modulo a prime q. Let l be a security parameter. Let $m \in \{0,1\}^*$. Define the following three hash functions: $H\colon \{0,1\}^* \to G$, $H'\colon G^6 \to \mathbb{Z}_q$, $H''\colon G \to \{0,1\}^l$.

• $\mathsf{FGen}(1^{|q|})$: Choose random points $d_0, \cdots, d_{t-1} \xleftarrow{\$} \mathbb{Z}_q$ and define a polynomial $F(X) = \sum_{j=0}^{t-1} d_j X^j \in \mathbb{Z}_q[X]$. For $i \in [1..n]$, set $x_i = F(i) \in \mathbb{Z}_q$ and $h_i = g^{x_i}$. Set $x = F(0)$ and $h = h_0 = g^x$. Set $pk = (G, g, h)$, $vk = (pk, h_1, \cdots, h_n)$, and $sk_i = (pk, x_i)$ for $i \in [1..n]$. Return (pk, vk, sk).

• $\mathsf{Eva}(pk, m, sk_i)$: Compute $\hat{h} = H(m)$, $\hat{h}_i = \hat{h}^{x_i}$. Choose at random $s \xleftarrow{\$} \mathbb{Z}_q$, compute $d = g^s$, $\hat{d} = \hat{h}^s$, $c = H'(g, h_i, d, \hat{h}, \hat{h}_i, \hat{d})$, and $z = s + x_i c$. Return $y_i = (\hat{h}_i, c, z)$.

• $\mathsf{Vrf}(vk, m, y_i)$: Given $y_i = (\hat{h}_i, c, z)$, check if $c = H'(g, h_i, d, \hat{h}, \hat{h}_i, \hat{d})$, where $d = g^z / h_i^c$ and $\hat{d} = \hat{h}^z / \hat{h}_i^{\,c}$. If the test holds, return $b = 1$. Otherwise, return $b = 0$.

• $\mathsf{FCom}(vk, m, \{y_j\})$: Given t valid PRF shares, run the Lagrange Interpolation in the Exponent on $\{y_j\}$ to obtain $y' = H(m)^x$. Return $y = H''(y')$.

Figure 1: CKS threshold PRF (FGen, Eva, Vrf, FCom) for a function $\mathsf{F}\colon \{0,1\}^* \to \{0,1\}^l$.

6 Poligraph: Intrusion-Tolerant Fake News Detection

6.1 Motivation and Pitfalls

We build a fake news detection system, Poligraph, combining ML servers and human expert determination. A client may submit news to the system and expect a result representing whether the news is fake. Servers run supervised ML models to generate a temporary assessment of whether the news is fake, along with a confidence score. Meanwhile, servers also invite reviewers, such as credentialed journalists, to review the news. If matching reviews from a majority of reviewers are received, the result for the news is finalized and will be sent to the client. The final result is also viewed as a label of the news item. Thus, the labeled data set maintained by the system can grow over time and can be used to re-train ML models and enhance the accuracy of the ML results in the long run.

Collecting reviews from reviewers, however, is tricky.

- First, different from computer nodes, reviewers can be (arbitrarily) slow: reviewers may not be responsive and the review process itself may take time. It is difficult to set up appropriate timers or rely on any timing assumptions. A system working in completely asynchronous environments is desirable.
- Second, one cannot ask all reviewers to provide reviews for every news feed. This is prohibitively expensive and slow. We can, however, allow selecting subsets of reviewers to perform parallel reviews for individual news feeds. This again poses various challenges: How can we select subsets of reviewers securely and fairly? What if a subset of reviewers happens to have a malicious majority?

Attempts. Our first attempt is to directly use BFT (e.g., PBFT [37]) or blockchain smart contracts (e.g., Chaincode in Hyperledger Fabric [75]). In this approach, we use a group of servers to run a BFT or blockchain system and treat both clients and reviewers as BFT or blockchain clients.

One may use a specific server, e.g., the leader of the BFT or blockchain system, to select subsets of reviewers. However, this approach does not guarantee the accuracy, because using the leader to select reviewers easily leads to biased results if a faulty leader colludes with certain reviewers. Ideally, reviewers for a news feed should be uniformly chosen at random among all reviewers with specific expertise to ensure fairness and accuracy of review results. Second, the approach impedes liveness. For instance, a faulty leader may simply choose not to send requests to reviewers.

To solve the issue, we use an asynchronous common coin protocol to unbiasedly select reviewer sets. In particular, all replicas jointly and asynchronously generate an uniformly random number corresponding to a random reviewer set. Moreover, each reviewer set should be *bound* to a specific news feed.

Existing blockchain systems, however, cannot be directly used for our purpose because they cannot efficiently support the data structures we need. Poligraph requires an integration of ML and BFT consensus, involving multiple data formats, e.g., database tables, in-memory hash maps, ML model initialization parameters. These formats are not well supported by existing smart contract and blockchain platforms. Additionally, Poligraph uses threshold PRF to instantiate an asynchronous common coin protocol. Existing blockchains do not support deploying such distributed cryptographic protocols using smart contracts. Therefore, a new system needs to be built.

There is another pitfall. Even if we have a way of fairly selecting subsets of reviewers and even if we can guarantee a majority of reviewers in a subset is correct, there are still potential inconsistencies among the servers. Suppose we have an intuitive protocol requiring servers to take majority votes on reviews. It is possible that replicas agree on the decision (whether or not news is fake) but fail to agree on the human reviews: some correct replicas receive reviews from a subset of reviewers, while some other correct replicas have received reviews from a different subset of reviewers. As these human reviews are used to run ML prediction and further used to perform periodical ML re-training, inconsistencies would occur among correct replicas.

Our approach at a high level. We present the concept of two-layer consensus, a new primitive for BFT consisting of a BFT server layer and a review layer. The BFT layer maintains the availability of the service and runs ML models to generate a temporary assessment of the news. The BFT layer also interacts with the review layer and integrates the results. Two-layer consensus provides a two-step validation for the data in client requests, integrates machine review results and expert review results, and enables online vs. offline response trade-offs. We use BFT and asynchronous common coin protocols to realize two-layer consensus and support parallel reviews. We further tune the parameters between the performance and accuracy of the final result. To solve the inconsistency problem, we treat reviews as transactions and explicitly *order* review transactions. In this way, the exact same reviews should be included to ensure that replicas have the same state and replicas have the same final result integrated from the reviews.

6.2 Poligraph: A Two-Layer Consensus Protocol for Fake News Detection

We present Poligraph, a distributed system that relies on two-layer consensus to provide an intrusion-tolerant, reliable, and tamper-proof service for fake news detection.

We believe that abstractly, the two-layer consensus is of separate interest and can be applied to other domains as well. In the rest of this section, we present the workflow of two-layer consensus for fake news detection. The workflow, however, is generic enough to be easily used in other domains.

Replica state. In our two-layer consensus protocol, there are three types of data: transactions (client requests and reviews), temporary results (client requests with pending finalized review results), and final results. The three types of data can be mapped to three tables: a *transaction table*, a *temporary state table*, and a *final state table*. The transaction table and final state table are append-only, and new entries will be appended to the end of the tables. The temporary results are put in the temporary state table, the entries of which will later be moved to the final state table.

Figure 2 depicts the workflow of the protocol. The BFT layer runs a BFT protocol and assigns sequence numbers to

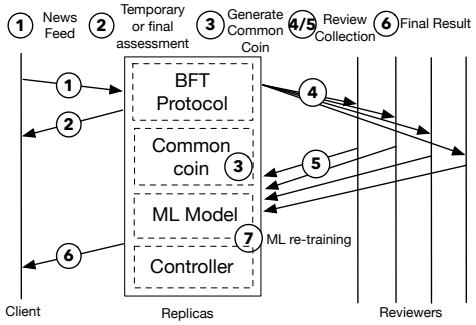


Figure 2: The Poligraph workflow.

both client requests and reviews. Poligraph can be built on top of any BFT protocol. If the BFT protocol is asynchronous (resp., partially asynchronous), our two-layer consensus protocol is asynchronous (resp., partially asynchronous). For instance, if we use HoneyBadgerBFT [57], BEAT [58], our protocol is asynchronous; if we use PBFT [37], our protocol works in partially synchronous environments.

System setup. We set up an $(f+1, n)$ threshold PRF scheme, Π = (FGen, Eva, Vrf, FCom), where a public key pk and verification keys vk are associated with the system, while a secret key is shared among all servers, with a server p_i having a key sk_i for $i \in [1..n]$. The keys can be generated either by a trusted dealer or running a distributed key generation algorithm [76], [77]. Let $h(\cdot)$ be a hash function.

Step 1: Client submits a request m to the replicas. The client sends a request of the form $\langle \text{REQUEST}, cid, ts, o \rangle$ to the servers, where cid is the client id, ts is the timestamp, and o is the data payload of a news feed.

Step 2: Replicas run the BFT protocol to assign a sequence number seq to the request m. After a replica delivers the request, the request is appended to the transaction table.

The replicas look up the final state table to identify if $h(o)$ exists, i.e., whether o has been stored in the final state table.

- If the data payload o already exists in the final state table, replicas directly send a final result to the client. In this case, while the request is new, the data payload for this request has been previously reviewed and finalized. The final result can be directly obtained from the final state table and sent to the client. Replicas complete the request.
- If the data payload o has not been reviewed before, each replica runs its ML model to obtain a temporary result of the news in the format of $\langle \text{TEMP}, ev, cl \rangle$. The value ev is a boolean result representing if the news is authentic. The value cl is the confidence level, where $0 < cl < 1$. After running the ML model, the replica sends the temporary result to the client and then continues to proceed to the reviewer selection procedures (step 3).

Step 3: Replicas agree on a random reviewer set.

- Each replica p_i runs Eva on (m, seq) to generate a threshold PRF share y_i and broadcasts $\langle \text{BEB}, seq, y_i \rangle$ to all other replicas.
- Upon receiving $f + 1$ *valid* PRF shares of the form (m, seq, y_j) from replica p_j, replica p_i runs FCom on $(vk, (m, seq), \{y_j\})$ and obtains a random output denoting a reviewer set of I invited reviewers.

Step 4/5: Replicas collect reviews from the reviewers.

- Each replica p_i sends the review request $\langle \text{REVIEW-REQUEST}, seq, o \rangle$ to the reviewers in the reviewer set. *We remove the identity information cid from the request m and reviewer set I and only forward the data payload.* This maintains decoupling between clients and reviewers in time and space, providing scalability and (weak) anonymity.
- Upon receiving $f + 1$ matching replica review requests with the same (seq, o), a reviewer reviews the payload request and sends replicas a review $\langle \text{REVIEW}, seq, b, rev \rangle$, where b denotes a binary review result, and rev is a detailed review.

Step 6: Replicas integrate reviews and send a final result to the client. A review of the form $\langle \text{REVIEW}, seq, b, rev \rangle$ is ordered as a transaction by replicas and appended to the transaction table. A replica begins integrating reviews after collecting *exactly* R reviews with the same b, where R is a system threshold parameter that will be specified shortly in Sec. 6.3. The entry in the temporary state table is removed. A new entry with the integrated final result is appended to the final state table. After collecting R reviews, any other reviews for m will not be stored or processed. The corresponding entry is then removed from the temporary state table, and the integrated final result is then sent to the client. Correspondingly, the client accepts the result after receiving $f + 1$ matching replies.

Now it becomes clear why we need to order reviews. In our protocol, the integrated final result for a client request takes as input R reviews of the form (b, rev). If we do not order reviews, it is possible that replicas agree on the same indicator bit but fail to agree on the reviews. Therefore, the exactly same reviews should be included to ensure that replicas have the same state and the same final result integrated from the reviews. The above argument also explains why we need to choose exactly R reviews for all correct replicas.

Step 7: ML model re-training. Periodically if a replica appends δ news to the final state table, the replica will copy the δ entries from the final state table to the labeled data set to re-train the ML model. The replica will stop processing new transactions while re-training, and once complete, resumes its normal operations. The step improves accuracy of the ML prediction over the time.

A machine learning model, making predictions (temporary results) for news from client requests, can take a few minutes or even longer. Therefore, we separate the training phase from the prediction phase. Specifically, during system setup, we pre-train the ML model using ground truth data set from our pre-collected labeled news. When the system processes client requests, the ML model processes the news by executing just the prediction phase. This will enable online ML model and reduce the time for training from scratch.

The need for threshold common coins. One may ask if we *need* to use threshold common coins: would it be possible to use pseudorandom function (PRF) or even a hash function to select reviewers? The idea of using a publicly known PRF or a hash function to generate random coins for BFT, or more generally, for state machine replication,

dates back to Rodrigues, Castro, and Liskov [78]. The idea can be used in "benign" applications, say, generating coins for randomized (non-deterministic) algorithms; it cannot be used in scenarios where coins may be manipulated by the adversary.

Our goal is to ensure that the reviewers are chosen in an unpredictable manner, without being controlled by the adversary. If the adversary can predict the reviewers chosen, it can feed into the system with adaptively chosen news feeds and maliciously train the ML model in the favor of the adversary.

6.3 Proof of Correctness and Parameter Selection

In this section, we prove the correctness of Poligraph and provide theoretical foundations on parameter selection.

Theorem 1. Two-layer consensus achieves agreement and total order.

Proof 1. Agreement and total order follow from the underlying BFT protocol since both requests and reviews are treated as transactions.

Theorem 2. Two-layer consensus achieves liveness-1 and liveness-2.

Proof 2. Liveness-1 easily follows from the livness of the underlying BFT protocol, as we order both expert reviews and client requests. Liveness-2 is immediately implied by liveness 1, as the temporary result does not involve the review layer.

Lemma 3. Correct replicas agree on the same set of reviewers for any transaction to be reviewed.

Proof 3. We observe that transactions are totally ordered by all correct replicas. All correct replicas will take the same input for the common coin protocol and thus obtain the same common coin. Therefore, all correct replicas will select the same set of reviewers.

Theorem 4. Two-layer consensus achieves liveness-3 under either of the following two conditions: 1) $I > 2B$ and $B + 1 \leq R \leq I - B$, where B is the number of faulty reviewers in the reviewer set selected by the replicas; 2) $N > 2F$ and $F + 1 \leq R \leq N - F$. (In particular, if the underlying BFT protocol is asynchronous, two-layer consensus achieves liveness-3 in asynchronous environments.)

Proof 4. According to Lemma 3, replicas agree on a common set of reviewers. It is straightforward to verify that both conditions guarantee that at least R correct reviews will be received by all replicas. Thus, all correct replicas will make the same review decision on the review. The theorem thus follows from the liveness of the underlying BFT protocol.

We define *perfect validity* and *probabilistic validity*. Perfect validity guarantees that the final result received by a client reflects a correct review, i.e., it is the same with that returned by the majority of the reviewers. Probabilistic validity guarantees the same result with an overwhelming probability in the parameter I (and parameters R, N, and F).

Lemma 5. It is computationally infeasible for adversaries to distinguish if the I reviewers in the selected reviewer set are uniformly chosen at random from all reviewers.

Proof 5. This is implied by the unpredictability property of the underlying threshold PRF protocol.

Theorem 6. Two-layer consensus achieves perfect validity if $F + 1 \leq R \leq I - F$.

Proof 6. Implied by Theorem 4.

Theorem 7. Two-layer consensus achieves probabilistic accuracy if $B + 1 \leq R \leq I - B$, and N and I are large enough.

Proof 7. Let α be the fraction of faulty nodes in N reviewers, i.e., $\alpha = \frac{F}{N}$. Let X_i be a random variable which outputs value one if the i-th review collected from the reviewer set is correct (where $i \in \{1, \cdots, I\}$). Let $X = \sum_1^I X_i$. Clearly, X follows a binomial distribution.

$$Pr[X \leq R] = \sum_{k=0}^{R} Pr[X = k] = \sum_{k=0}^{R} \binom{I}{k} \alpha^{I-k}(1-\alpha)^k \quad (2)$$

It is known that the probability decreases exponentially as I increases. That is, given a security parameter λ, there exists an I_0 so that $\forall I > I_0$, $Pr[X \leq R] \leq 2^{-\lambda}$.

Examples. For Theorem 7, if we set $\alpha = \frac{1}{3}$, $\lambda = 20$, and $R = \lfloor \frac{I+1}{2} \rfloor$, we have $I_0 = 180$. In words, if 1/3 of the total N reviewers are faulty and if each time we select 180 reviewers and expect 91 matching reviews to finalize the result, then the probability that the review result does not represent a correct review is once every 1 million reviewer set selections.

As another example, we set $\alpha = 0.1$, $\lambda = 10$, and $R = \lfloor \frac{I+1}{2} \rfloor$ and we have $I_0 = 13$. In words, if 10% of the reviewers are faulty and if we select 13 reviewers and expect 7 matching reviews to finalize the result, then the probability that the review result does not represent a correct review is once every thousand reviewer set selections.

Generalizing the acceptance condition and the system assumption. So far we have set the acceptance condition as a simple threshold, i.e., replicas can make a decision after receiving exactly R reviews with the same indicator bit b. One can easily extend the acceptance condition to a more general predicate on the reviews received. It is also easy to make stronger system assumptions regarding the percentage of faulty reviewers.

6.4 Machine Learning State Transfer

BFT nodes may experience (temporary) failures or attacks, causing some "fall-behind" nodes. While several BFT protocols provide schemes for recovering nodes from failures [37], for systems with ML models integrated, it is not clear how we can efficiently transfer ML state. In Poligraph, we provide an efficient machine learning state transfer protocol to bring fall-behind replicas up to date.

In Poligraph, each replica maintains a system state st and a set of ML state parameters $a_1, a_2, \cdots, a_k$. The system state in Poligraph is defined to be the the hash of the transaction history, i.e., the state after processing tx_j is $\mathsf{st} = h(tx_j, h(\mathsf{st}_{j-1}))$. In addition, the ML state parameters are initialized with the parameters for the ML model.

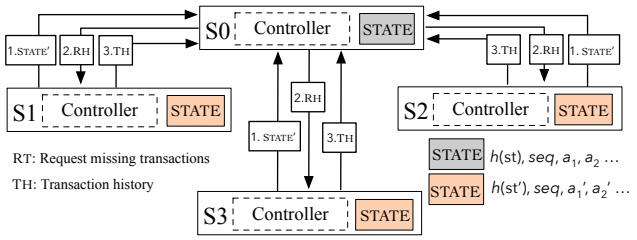


Figure 3: The ML state transfer architecture.

As illustrated in Figure 3, we provide a controller module for each replica. The controller is used for consistency checking and state transfer. In case of a replica inconsistency, the controller will trigger the state transfer protocol and re-initiate the ML model, thereby bringing the replica up to date.

The state transfer protocol in Poligraph works as follows.

Step A: Send ML and replica state. Upon executing a fixed number of transactions, each replica sends a message in the form of $\langle \text{STATE}, h(\mathsf{st}), seq, a_1, a_2, \cdots, a_k \rangle$ to other replicas, where $h(\mathsf{st})$ is the hash of the state st, seq is the largest sequence number in the transaction table, and $a_1, a_2, \cdots, a_k$ are the current ML state parameters.

Step B: Controller module collects $\langle$STATE$\rangle$ messages. For a replica p_i, if it receives at least $2f+1$ matching messages (denoted as $\langle \text{STATE}, h(\mathsf{st}'), seq, a'_1, a'_2, \cdots, a'_k \rangle$), it compares the received state st' and its local state st.

- If st' matches st, the state is considered as a *stable state*.
- If st' does not match st, p_i starts state transfer. Let seq' be the last sequence number where p_i was consistent with other replicas. To begin state transfer, p_i first discards its local state, transactions, and system logs between seq' and seq. For all the incoming transactions, p_i still participates in the consensus. However, it only updates the transaction table and does not process the transactions or reply to the clients. Meanwhile, p_i sends a message $\langle \text{RT}, seq', seq \rangle$ to other replicas and requests the transactions between seq' and seq. If a replica receives a $\langle \text{RT} \rangle$ message and has executed transactions with sequence numbers greater than or equal to seq, a replica will reply with a $\langle \text{TH}, m_{seq'}, \cdots, m_{seq} \rangle$, where $m_{seq'}$ to m_{seq} are the transactions with sequence numbers between seq' and seq. Upon receiving matching $\langle \text{TH} \rangle$ messages from at least $2f+1$ replicas, p_i has all the transactions with sequence numbers greater than seq'. Replica p_i will append these transactions to its transaction table. Also, p_i re-initializes its local ML model using the state parameters $a'_1, \cdots, a'_k$. Then it processes the transactions with sequence numbers greater than seq'.

7 Discussion

The benefits to fake news detection. The purpose of the system is to enhance the quality of the data sets for fake news detection utilizing two-layer BFT consensus. In other words, our goal is not to study or enhance ML algorithms. Instead, due to the growth of the labeled data set (final state table), the accuracy of fake news detection will grow over time.

Human review considerations. Although our paper provides a scheme to select uniformly random reviewers to review the contents and ensure fairness, we cannot guarantee that reviewers are unbiased. Our assumption is that the majority of the reviewers provide the same review results. Indeed, whether the reviews are biased is related to the types of news, which is orthogonal to the focus of this paper.

Considerations about binary labels for news. Data set such as the one from PolitiFact.com has several labels, i.e., true, mostly true, half true, mostly false, false, and pants on fire. We simplify them into a binary result, i.e., true or false, for both ML results and human reviews. This is mainly because, in our work, we consider an open framework where non-experts (humans with certain knowledge) can serve as reviewers. Furthermore, there are a large number of reviewers. It is challenging for non-experts to distinguish *mostly true* from *half true* and similarly, *mostly false* from *false*. Furthermore, differentiating news into multiple categories may create biased results. Therefore, we choose to use binary result in our framework.

Generality of Poligraph. Although we focus on fake news detection in Poligraph, the two-layer consensus protocol can be used in data science in general or even applied to other domains. For instance, consider a medical diagnosis problem. A typical workflow is that doctors diagnose manually based on patient data, e.g, an image. If 3 out of 5 doctors have a matching diagnosis, the corresponding image is labeled. The labeled data set is then used for ML training. This is another ideal application for two-layer consensus where doctors become *reviewers*. Patients get diagnosis results from ML in the first layer and then final results from reviews. The labeled data set grows over time and is stored securely in the system.

Feasibility of Poligraph. Poligraph is an open platform for fake news detection where reviewers are not necessarily experts. As shown in our case study, people with certain background knowledge can justify the authenticity of the news much better than others. Therefore, Poligraph is more suitable for news that is not subjective so that human review results are not biased. Reviewers can be humans with expertise (or at least some familarity) in certain domains. In a complete system where news may fall into different domains, reviewers can be tagged by their expertise. News can also be tagged where they are only sent to reviewers with the same tags. However, the considerations of *who should serve as reviewers* are related to the concrete application if we view Poligraph as a generic architecture.

Considerations of the validity of the labels. In our work, we consider the labels of the news, once reviewed by reviewers, are final, i.e., if the same news is queried again, the result is directly replied to the client. For some news, the authenticity might change over time, e.g., even for some scientific claims, the authenticity may change. We consider this issue a validity problem of the labels. One approach to address this problem is to set up an expiration time for each final label of the data. We can assign the timestamp to each label and an expiration time/data. After the expiration time, the label can be considered invalid. If the news is queried again, it has to go through all the processes and get reviewed by reviewers again.

8 IMPLEMENTATION

We use ECDSA for authentication and use SHA-256 as the hash function. We implement CKS threshold PRF [31] using the Charm Crypto Python library [79]. We use the NIST P-256 elliptic curve to provide 128-bit security.

We use BFT-SMaRt [80] as the underlying BFT engine, as it is "the most advanced and most widely tested implementation of a BFT consensus protocol" [65]. We extend the BFT-SMaRt library and implement a client-server service. Then we connect the Java library with a Python library in which we implement the ML models and the review layer. During performance evaluation, we code *human reviewers* where each reviewer node directly sends a pre-defined review to the replicas. We separate our evaluation of reviewers into two parts. When we evaluate the system performance (scalability, throughput, etc.), we *implement* the reviewers. Specifically, each reviewer script automatically replies with the review result for each news.

We store the data and the state tables in three formats. First, the transactions in the transaction table are written into the database. We use batching to reduce hard disk access and improve the system throughput. We set up a tunable parameter, BatchSize, to provide performance trade-offs. Second, the final state table and temporary state table are stored in memory using hash maps. Last, we store the labeled data set constructed from the final state table in a csv file for the ML model, which is friendly for ML re-training.

We also implement the data preparation and ML models. We first build a data crawler to collect labeled data sets using two python libraries Beautifulsoup [81] and Selenium [82]. We use the labeled data sets to train our ML models to form the initial machine learning state. We then use doc2vec google news corpus [72] to extract the news word embedding as the news features. We select 300 as the feature vector size. Then, we adopt python scikit-learn [83] to implement LR and SVM models. We serialize all the parameters into string format and send them over the network. When a node receives the state, it deserializes the state into the state parameters and initializes the ML model.

For the ML model selection in our system, we use LR. Meanwhile, other machine learning models are available, e.g., random forest and XGBoost [84], [85]. Some prior works also compare different fake news detection models in terms of ML performance. However, depending on the concrete scenarios, the models may achieve different performance. For example, it was shown in [85] that XGBoost achieves the best performance and in [86] that LR achieves the best performance. Therefore, we select the LR model in our implementation because it is a classic ML model, and can achieve low latency and high throughput. To validate the fact that our system can be integrated with any ML model, we also implement SVM model in our system.

Evaluating Poligraph. As shown in Sec. 3, we have conducted experiments for human reviewers to demonstrate the need for Poligraph. Besides, We evaluate the system performance of Poligraph. In particular, we *implement* the behaviors of reviewers and show the results in Sec. 9. The purpose is to evaluate the system performance under extreme conditions, i.e., high concurrency of client requests and a large number of human reviewers. Our evaluation focus on metrics such as latency of each client request (how long does it take for a user of Poligraph to get a ML response to its request), system throughput (how many requests can Poligraph handle at the same time), and ML performance (whether Poligraph can benefit the data science community in enhancing the data quality).

9 EVALUATION

We deploy and test our protocols on Amazon EC2 utilizing up to 35 nodes (VMs). Each node is a general-purpose *t2.medium* type with two virtual CPUs and 4GB memory. Among these nodes, we use up to 10 different nodes to run the replicas, 5 to run the reviewers, and 20 to run up to 1,000 clients. We use each node to simultaneously run up to 50 clients. We compare Poligraph with a single-server, unreplicated implementation (denoted as $n = 1$ in the figures).

9.1 Overview

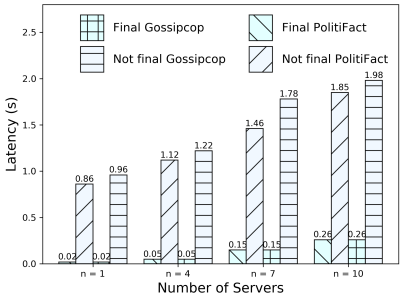
We evaluate the performance of Poligraph from both system and ML perspectives. From the system perspective, we evaluate both latency (the delay between a client request and response) and throughput (the number of transactions processed per second) using different benchmarks. In addition, we show the peak throughput for parameters such as news length (the number of characters) and δ. Recall δ is a parameter, where after every δ news is appended to the final state table, we update the labeled data set and re-train the ML model. From the ML perspective, we show that as the size of labeled data for training grows over time and reviews are integrated, the accuracy of the ML models' temporary feedback score improves.

We build several benchmarks for evaluation. In each benchmark, a fraction of client requests has already been reviewed and finalized. We use the notation $x\%$ *final data set* to represent the case where $x\%$ of all the client requests are already finalized. For all the experiments, we calculate the average throughput/latency for all of the replicas.

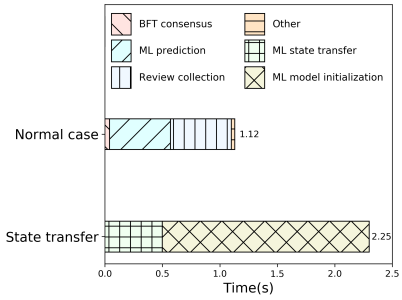
We have evaluated the performance using both political news (from PolitiFact.com) and entertainment news (from GossipCop.com). Our evaluation shows that Poligraph incurs latency of 0.05 second and achieves throughput as high as 5,500 tx/s. In addition, compared with a single-node implementation, the throughput of Poligraph is about 4%-7% lower.

9.2 Latency for Normal Operations and State Transfer

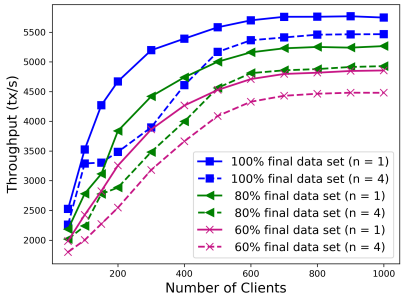
We assess the latency for $f = 1$, 2, and 3, where a single client issues one news verification request to the system. We assess the two cases: 1) the news already exists in the final state table; 2) the news has never been reviewed before. As shown in Figure 4a, the latency of the first case is much lower than that of the second case for both political news and entertainment news. This is because in the first case, the replicas directly reply with a final result, while in the second case, the replicas run the ML model to obtain a temporary result and collect reviews from reviewers. Due to the use of ML optimization, the latency is 1 to 2 seconds in the worst case. The latency for entertainment news is higher than that



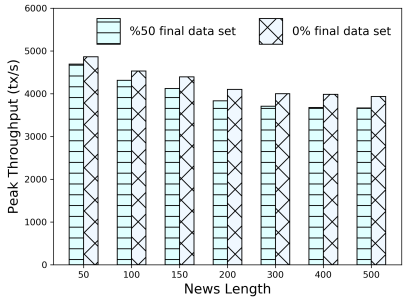
(a) Latency of Poligraph using 4 to 10 servers for the two cases where the news already exists in the final state table (final data set) and the news has never been reviewed before (not in final data set).



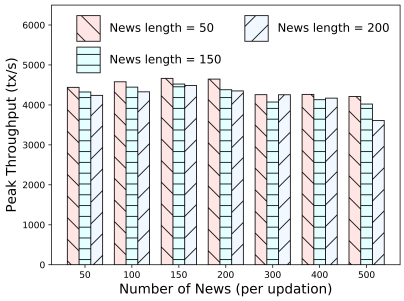
(b) Latency breakdown for normal case and state transfer.



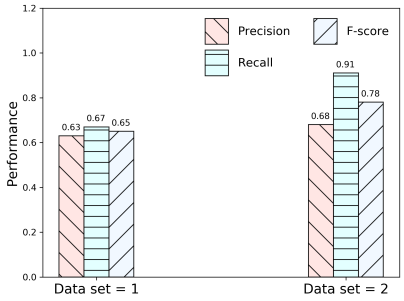
(c) Throughput of the Poligraph using 4 servers using a number of benchmarks, where a fraction of news have already existed in the final state table.



(d) Peak throughput v.s. different the news lengths using 4 servers, 5 reviewers, and 700 clients.



(e) Peak throughput v.s. δ using 4 servers, 5 reviewers, and 700 clients.



(f) ML model performance improvement over time.

Figure 4: Performance of Poligraph.

with political news, mainly due to news length. We also report the latency breakdown of the normal case for political news in Figure 4b, where the client sends a news item that has never been processed before. As shown in the figure, the ML and the review collection processes dominate the latency. In practice, the review process can take much longer and becomes the bottleneck of the system. However, the client can get the temporary result immediately after the ML step and the reviews can be collected asynchronously.

We evaluate the latency for state transfer. We inject one failure and let one node become inconsistent with other nodes. The nodes will then run the state transfer protocol as described in Sec. 6.4. We also measure the latency breakdown in Figure 4b. The overall latency for state transfer is around 2 seconds on average, and the ML state re-initialization dominated the latency.

9.3 Throughput

We use three benchmarks to evaluate the throughput and compare them with an unreplicated implementation of the system. We assess the throughput by gradually increasing the number of clients. For all these experiments, we use news with length 150. We use 4 servers and 5 reviewers, and set up BatchSize to 100. The replicas do not re-train their ML models.

As we can see from Figure 4c, all experiments have a similar trend. As the number of clients increases, the system will be saturated with transactions, and the throughput will reach its peak. In almost all experiments, the throughput reaches its peak when the number of clients is greater than 600. However, the higher the fraction of news that is already in the final state table, the higher throughput the system can achieve. This is expected, for the same reason we observe when assessing latency: if a news item has been recorded in the final state table, the final result is directly returned to the client. In the worst case, where 60% of the news has never been reviewed before, the peak throughput can still be higher than 4,000 tx/s. In the best case, where all the news items have are already in the final state table, the peak throughput is close to 5,500 tx/s. In comparison, the peak throughput of the unreplicated version of Poligraph ($n = 1$) is close to 6,000 tx/s. In other words, Poligraph is only marginally slower than its unreplicated version.

9.4 Impact of News Length

The lengths of news in the client requests will affect the performance of the system in two ways: network bandwidth and ML model prediction speed. Since nodes have to exchange several all-to-all messages in the consensus protocol, longer messages will consume higher network bandwidth which results in downgraded throughput. The ML model execution time increases when it has to predict the veracity of longer news items.

We evaluate the peak throughput by varying news lengths in the client requests from 50 to 500. In all experiments, we use 700 clients and no ML re-training is triggered. As shown in Figure 4d, the throughput indeed decreases as the length of news increases. The curves for the two benchmarks are relatively smooth, which demonstrates that Poligraph is capable of processing long news efficiently. Note that for the 0% final data set benchmark, the replicas need to run the ML model for each transaction. The performance degradation, however, is almost similar to that of the 50% final data set benchmark. Therefore, the news length is

not a bottleneck to the system performance due to the use of ML optimization.

9.5 Impact of ML Re-training

In this experiment, we aim to evaluate the throughput under ML re-training. We use 700 clients, 4 servers, and 5 reviewers. We vary the δ parameter from 50 to 500. We run three benchmarks where news lengths are 50, 150, and 200, separately. As illustrated in Figure 4e, we find that as δ increases, the peak throughput first increases and then decreases. The same trend applies to all three benchmarks. Note that as δ increases, the performance will be affected in two ways. First, the ML model will be re-trained less frequently, during which the protocol will be stalled. Each re-training process, however, will have higher latency. Second, when we update the labeled data set, we simply write all the new data from the final state table to the csv file. The writing time is related to the length of the data. Other operations, such as csv file open and close, are triggered per update. Therefore, as δ increases, csv file operation and ML re-training are triggered less frequently.

As δ first increases from 50 to 200, since re-training is triggered less frequently, the peak throughput will be higher. As the δ further increases, however, the performance degrades. This is because the latency due to csv file update is related to the length of the data. Note that each ML re-training will stall the system performance for a longer period of time. In summary, we find that the optimal performance can be achieved when we update the csv file every 150 to 200 news items.

9.6 Scalability

We assess the peak throughput of Poligraph using n equals 4 to 10. We use 5 reviewers and set up news length as 150. We use 700 clients and run the 50% final data set benchmark.

n	Peak Throughput (tx/s)	Degradation (%)
4	4358.75	—
7	4171.50	(4.29%)
10	3955.41	(5.18%)

Table 4: Scalability of Poligraph

We show the performance degradation as the number of BFT servers n increases in Figure 4. As shown in the table, the performance does not degrade significantly when the number of servers increases. The performance degradation of each experiment, when compared with the previous experiment with a smaller n, is 4-5%. We find a similar result when measuring latency, as shown in Figure 4a. We comment that the scalability of the system is related to the underlying BFT protocol though.

9.7 Machine Learning Performance Improvement

We show that the ML performance improves as a growing labeled data set is generated by Poligraph. We first run the protocol and let clients send requests from data set 1 collected from PolitiFact.com with 14,788 data entries. On the server side, the news in the data set 1 is not stored in the final state table, and the servers will run ML model for every news article. After 4,000 news items have been evaluated by Poligraph and put into the final state table, we re-train the ML model using the 18,788 entries (the data set 2).

We utilize 10 cross-validation methods to run LR and get precision, recall, and f1-score to evaluate the machine learning classification performance.

Figure 4f shows the performance improvement for LR. As expected, we find that the precision, recall, and f1-score are improved. This demonstrates that Poligraph can effectively enhance the accuracy of temporary results from the ML models.

10 Conclusion

We motivate the need for combing machine learning and expert reviews for fake news detection via a real-world case study. We design and implement Poligraph, an intrusion-tolerant fake news detection system, defending against Byzantine failures and malicious attacks. Poligraph combines machine learning techniques and human expert determination based on two-layer consensus and resolves many key challenges in fake news detection (e.g., online and offline trade-offs, secure parallel reviews, practical state transfer). Our extensive evaluation on Amazon EC2 demonstrates that Poligraph achieves latency as low as 0.05 second and throughput of more than 5,000 tx/s, being only marginally (4%-7%) slower than an unreplicated, single-server implementation.

Acknowledgment

Boxin Zhao, Haibin Zhang, and Sisi Duan's work was supported in part by Shandong Key Research and Development Program under grant No. 2020ZLYS09. Boxin Zhao and Sisi Duan's work was supported in part by National Key Research and Development Program of China under grant No. 2018YFA0704701.

References

[1] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, "Truth of varying shades: Analyzing language in fake news and political fact-checking," in *EMNLP*, 2017, pp. 2931–2937.

[2] D. M. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer, M. J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild *et al.*, "The science of fake news," *Science*, vol. 359, no. 6380, pp. 1094–1096, 2018.

[3] V. L. Rubin, Y. Chen, and N. J. Conroy, "Deception detection for news: three types of fakes," in *ASIST*, 2015, p. 83.

[4] V. L. Rubin, N. J. Conroy, and Y. Chen, "Towards news verification: Deception detection methods for news discourse," in *HICSS 2015*.

[5] J. Gottfried and E. Shearer, *News Use Across Social Medial Platforms 2016*. Pew Research Center, 2016.

[6] G. Pennycook, T. Cannon, and D. G. Rand, "Prior exposure increases perceived accuracy of fake news," *Journal of experimental psychology: general*, vol. 147, no. 12, p. 1865, 2018.

[7] S. Tschiatschek, A. Singla, M. Gomez Rodriguez, A. Merchant, and A. Krause, "Fake news detection in social networks via crowd signals," in *The Web Conference*, 2018, pp. 517–524.

[8] E. C. Tandoc Jr, R. Ling, O. Westlund, A. Duffy, D. Goh, and L. Zheng Wei, "Audiences' acts of authentication in the age of fake news: A conceptual framework," *New Media & Society*, 2017.

[9] D. Renzel, K. A. Rashed, and R. Klamma, "Collaborative fake media detection in a trust-aware real-time distribution network," in *SMDT*, vol. 680, 2010.

[10] R. J. Sethi, "Crowdsourcing the verification of fake news and alternative facts," in *HT*, 2017, pp. 315–316.

[11] Z. Jin, J. Cao, Y. Zhang, and J. Luo, "News verification by exploiting conflicting social viewpoints in microblogs," in *AAAI 2016*.

[12] M. D. Vicario, W. Quattrociocchi, A. Scala, and F. Zollo, "Polarization and fake news: Early warning of potential misinformation targets," *TWEB*, vol. 13, no. 2, pp. 1–22, 2019.

[13] J. A. Nasir, O. S. Khan, and I. Varlamis, "Fake news detection: A hybrid cnn-rnn based deep learning approach," *International Journal of Information Management Data Insights*, vol. 1, no. 1, 2021.

[14] N. Aslam, I. Ullah Khan, F. S. Alotaibi, L. A. Aldaej, and A. K. Aldubaikil, "Fake detect: A deep learning ensemble model for fake news detection," *Complexity*, vol. 2021, 2021.

[15] M. S. Mokhtar, Y. Y. Jusoh, N. Admodisastro, N. Pa, and A. Y. Amruddin, "Fakebuster: Fake news detection system using logistic regression technique in machine learning," *IJEAT*, vol. 9, no. 1, pp. 2407–2410, 2019.

[16] M. L. Della Vedova, E. Tacchini, S. Moret, G. Ballarin, M. DiPierro, and L. de Alfaro, "Automatic online fake news detection combining content and social signals," in *2018 22nd Conference of Open Innovations Association (FRUCT)*. IEEE, 2018, pp. 272–279.

[17] Z. Jin, J. Cao, Y. Zhang, J. Zhou, and Q. Tian, "Novel visual and statistical image features for microblogs news verification," *IEEE transactions on multimedia*, vol. 19, no. 3, pp. 598–608, 2016.

[18] Y. Wang, F. Ma, Z. Jin, Y. Yuan, G. Xun, K. Jha, L. Su, and J. Gao, "Eann: Event adversarial neural networks for multi-modal fake news detection," in *SIGKDD*, 2018, pp. 849–857.

[19] D. Khattar, J. S. Goud, M. Gupta, and V. Varma, "Mvae: Multimodal variational autoencoder for fake news detection," in *WWW*, 2019, pp. 2915–2921.

[20] S. Singhal, R. R. Shah, T. Chakraborty, P. Kumaraguru, and S. Satoh, "Spotfake: A multi-modal framework for fake news detection," in *BigMM*. IEEE, 2019, pp. 39–47.

[21] S. Singhal, A. Kabra, M. Sharma, R. R. Shah, T. Chakraborty, and P. Kumaraguru, "Spotfake+: A multimodal framework for fake news detection via transfer learning (student abstract)," in *AAAI*, vol. 34, no. 10, 2020, pp. 13 915–13 916.

[22] L. Cui, S. Wang, and D. Lee, "Same: sentiment-aware multi-modal embedding for detecting fake news," in *ASONAM*, 2019, pp. 41–48.

[23] X. Zhou, J. Wu, and R. Zafarani, "Safe: Similarity-aware multi-modal fake news detection," *PAKDD*, vol. 12085, p. 354, 2020.

[24] L. Mearian, "Facebook's blockchain cryptocurrency could mean big money – and kill 'fake news'," Computer world, 2019.

[25] W. Mason and S. Suri, "Conducting behavioral research on amazon's mechanical turk," *Behavior research methods*, vol. 44, no. 1, pp. 1–23, 2012.

[26] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," in *NIPS*, 2019, pp. 9051–9062.

[27] "Fakebox," https://machinebox.io/docs/fakebox (Dec 2021).

[28] A. D. Holan, "The principles of the truth-o-meter: Politifact's methodology for independent fact-checking," https://www.politifact.com/truth-o-meter/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/, 2018.

[29] "A project of the annenberg public poligy center," https://www.factcheck.org (Dec 2021), Annenberg.

[30] R. Heilweil, "Facebook is flagging some coronavirus news posts as spam," Vox, 2020.

[31] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," *Journal of Cryptology*, vol. 18, no. 3, pp. 219–246, 2005.

[32] T. Dalrymple, "On the detection of fakes," *BMJ*, vol. 334, no. 7599, pp. 905–905, 2007.

[33] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *ASIS&T*, vol. 52, no. 1, pp. 1–4, 2015.

[34] Z. Jin, J. Cao, Y.-G. Jiang, and Y. Zhang, "News credibility evaluation on microblog with a hierarchical propagation model," in *ICDM*. IEEE, 2014, pp. 230–239.

[35] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *AJCAI*, 2006, pp. 1015–1021.

[36] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," *arXiv preprint arXiv:1708.07104*, 2017.

[37] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *TOCS*, vol. 20, no. 4, pp. 398–461, 2002.

[38] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *TOCS*, vol. 32, no. 4, pp. 12:1–12:45, 2015.

[39] J. Hendricks, S. Sinnamohideen, G. R. Ganger, and M. K. Reiter, "Zzyzx: Scalable fault tolerance through byzantine locking," in *DSN*. IEEE, 2010, pp. 363–372.

[40] F. Dettoni, L. C. Lung, M. Correia, and A. F. Luiz, "Byzantine fault-tolerant state machine replication with twin virtual machins," in *ISCC*, 2013, pp. 398–403.

[41] R. Kapitza, J. Behl, C. Cachine, T. Distler, S. Kuhnle, S. V. Mohammadi, W. Schröder-Preikschat, and K. Stengel, "CheapBFT: Resource-efficient Byzantine fault tolerance," in *EuroSys*, 2012, pp. 295–308.

[42] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient byzantine fault tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, 2013.

[43] S. Nikolaou and R. van Renesse, "Turtle consensus: Moving target defense for consensus," in *Middleware*, 2015, pp. 185–196.

[44] J.-P. Bahsoun, R. Guerraoui, and A. Shoker, "Making BFT protocols really adaptive," in *IPDPS*. IEEE, 2015, pp. 904–913.

[45] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolic, "XFT: Practical fault tolerance beyond crashes," in *OSDI*, 2016, pp. 485–500.

[46] J. Li and D. Maziéres, "Beyond one-third faulty replicas in byzantine fault tolerant systems," in *NSDI*, 2007.

[47] S. Duan, S. Peisert, and K. N. Levitt, "hBFT: Speculative byzantine fault tolerance with minimum cost," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 58–70, 2015.

[48] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, "Attested append-only memory: making adversaries stick to their word," in *SOSP*, 2007, pp. 189–204.

[49] S. Duan, K. Levitt, H. Meling, S. Peisert, and H. Zhang, "ByzID: Byzantine fault tolerance from intrusion detection," in *SRDS*. IEEE, 2014, pp. 253–264.

[50] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda, "TrInc: Small trusted hardware for large distributed systems," in *NSDI*, 2009, pp. 1–14.

[51] M. Correia, N. F. Neves, and P. Verissimo, "How to tolerate half less one byzantine nodes in practical distributed systems," in *SRDS*. IEEE, 2004, pp. 174–183.

[52] P. Zielinski, "Optimistically terminating consensus: All asynchronous consensus protocols in one framework," in *ISPDC*. IEEE, 2006, pp. 24–33.

[53] P. J. Marandi, M. Primi, and F. Pedone, "High performance state-machine replication," in *DSN*. IEEE, 2011, pp. 454–465.

[54] R. van Renesse, C. Ho, and N. Schiper, "Byzantine chain replication," in *OPODIS*, 2012, pp. 345–359.

[55] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: Byzantine replication with high throughput and embedded reconfiguration," in *OPODIS*, 2014, pp. 91–106.

[56] R. van Renesse and F. B. Schneider, "Chain replication for supporting high throughput and availability," in *OSDI*, 2004, pp. 91–104.

[57] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *CCS*, 2016, pp. 31–42.

[58] S. Duan, M. K. Reiter, and H. Zhang, "BEAT: Asynchronous BFT made practical," in *CCS*, 2018, pp. 2028–2041.

[59] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology? a systematic review," *PloS one*, vol. 11, no. 10, p. e0163477, 2016.

[60] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin, 2008.

[61] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.

[62] M. Vukolić, "Rethinking permissioned blockchains," in *BCC*. ACM, 2017, pp. 3–7.

[63] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," *Master thesis, The University of Guelph*, 2016.

[64] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *iNetSeC*, 2015, pp. 112–125.

[65] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," in *DISC*, 2017, pp. 1:1–1:16.

[66] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.

[67] E. Androulaki, C. Cachin, D. Dobre, and M. Vukolić, "Erasure-coded byzantine storage with separate metadata," in *OPODIS*, 2014, pp. 76–90.

[68] C. Cachin, D. Dobre, and M. Vukolić, "Separating data and control: Asynchronous bft storage with 2t+ 1 data replicas," in *SSS*, 2014, pp. 1–17.

[69] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin, "Separating agreement from execution for byzantine fault tolerant services," in *SOSP*, 2003.

[70] S. Duan and H. Zhang, "Practical state machine replication with confidentiality," in *SRDS*. IEEE, 2016, pp. 187–196.

[71] Y. Wang, L. Alvisi, and M. Dahlin, "Gnothi: Separating data and metadata for efficient and available storage replication." in *USENIX Annual Technical Conference*, 2012, pp. 413–424.

[72] M. Mihltz, "Pre-trained word2vec google news corpus," Google, 2017.

[73] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *JACM*, vol. 35, no. 2, pp. 288–323, 1988.

[74] D. Crankshaw, *The design and implementation of low-latency prediction serving systems*. University of California, Berkeley, 2019.

[75] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *EuroSys*, 2018, pp. 1–15.

[76] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *J. Cryptology*, 2007.

[77] A. Kate, Y. Huang, and I. Goldberg, "Distributed key generation in the wild." *IACR Cryptology ePrint Archive*, vol. 2012, p. 377, 2012.

[78] M. Castro, R. Rodrigues, and B. Liskov, "Base: Using abstraction to improve fault tolerance," *ACM Trans. Comput. Syst.*, vol. 21, no. 3.

[79] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *JCEN*, vol. 3, no. 2, pp. 111–128, 2013.

[80] J. Sousa, E. Alchieri, and A. Bessani, "State machine replication for the masses with bft-smart," in *DSN*, 2014, pp. 355–362.

[81] L. Richardson, "Beautiful soup documentation," Python, 2007.

[82] H. Percival. " O'Reilly Media, Inc.", 2014.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *JMLR*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[84] B. Bhutani, N. Rastogi, P. Sehgal, and A. Purwar, "Fake news detection using sentiment analysis," in *IC3*, 2019, pp. 1–5.

[85] J. Lin, G. Tremblay-Taylor, G. Mou, D. You, and K. Lee, "Detecting fake news articles," in *Big Data*. IEEE, 2019, pp. 3021–3025.

[86] N. Pinnaparaju, V. Indurthi, and V. Varma, "Identifying fake news spreaders in social media." in *CLEF (Working Notes)*, 2020.

Guohou Shan is a PhD Candidate in the Department of Management Information System of Temple University. His research interests include Fake news, Healthcare IT, blockchain, and online community.

Boxin Zhao received his PhD degree in Cyber Science and Technology from Shandong University in 2020. He is currently a postdoctor at the Institute for Advanced Study, Tsinghua University. His research interests include cryptography and blockchain.

James Clavin is currently pursuing a PhD in information systems with a focus in healthcare informatics from the University of Maryland Baltimore County. He is the chief technology and compliance officer at Hilltop Institute. Jim has an MBA from the University of Baltimore, a BS from UNC-Wilmington, and a BA from UNC-Chapel Hill. His research interests include healthcare informatics and blockchains.

Haibin Zhang received his PhD degree in computer science from University of California, Davis in 2014. He is a research scientist in Shandong Institute of Blockchain. During the academic year 2017-2020, he was an assistant professor in the CSEE Department at University of Maryland, Baltimore County, leading the Distributed Systems and Security (DSS) Lab. His research interests are in the intersection of distributed systems, system security, and applied cryptography, particularly in the design and implementation of Byzantine fault-tolerant (BFT) distributed systems.

Sisi Duan received her Ph.D. in Computer Science from the University of California, Davis in 2014. She is currently a researcher at the Institute for Advanced Study, Tsinghua University. She is also a member of Beijing National Research Center for Information Science and Technology. Prior to joining Tsinghua University, she was an Assistant Professor at the University of Maryland, Baltimore County from 2017 to 2020 and a Weinberg Fellow at Oak Ridge National Laboratory from 2015 to 2017. Dr. Duan's research interests include security, blockchain, distributed systems, and applied cryptography. She is a member of IEEE.