A Two-Stage Encrypted Cryptomining Traffic Detection Mechanism in Campus Network

***Abstract*—Cryptomining behaviours pose severe security threats to campus network. However, existing blacklist and DPI-based techniques suffer from delayed blacklist updates and inability to identify encrypted cryptomining traffic. Furthermore, existing encrypted cryptomining traffic detection schemes usually fail to provide detailed information about cryptomining behaviours and do not have a solution to deal with false positives caused by detection models. To meet the needs of campus networks and solve the problems of existing work, this paper proposes an effective and practical encrypted cryptomining traffic detection mechanism in campus network. It consists of a two-stage detection framework, which can effectively provide fine-grained detection results by machine learning and reduce false positives from classifiers through active probing. Based on our collected dataset and extracted time series features, our classifiers detect mining traffic with an 0.99 F1 score and identify the cryptocurrency being mined with 99.39% correct recognition rate. Unlike existing schemes, we perform active probing after the traffic classification to reduce false positives. Futhermore, we have extensively evaluated the active probing scheme to verify its effectiveness for different mining pools.**

***Keywords*—Encrypted Cryptomining Traffic Detection, Campus Network, Machine Learning, Active Probing.**

I. Introduction

In the early days of Bitcoin [1], individuals could earn cryptocurrency from cryptomining on personal computers. However, the rapid growth of the cryptomining industry has significantly increased the overall network hashrate. By November 2023, Bitcoin's total hashrate had reached 459 EH/s [2], making it almost impossible for solo miners to profit independently. Consequently, most miners have joined pools to ensure a steady income from mining [3].

Additionally, some individuals engage in malicious activities, such as embedding scripts in websites [4] or installing malware on victims' devices [5]. Victims' computing power is used by attackers for profit and even incorporated into botnets for cryptomining [6]. Such behaviours not only compromise cyberspace security but also potentially hinder economic and social progress, as well as efforts in energy conservation and emissions reduction.

A. Motivation and Problem Statement

In campus networks, two main risks prevail: insider misuse and external cyber-attacks [7]. Insiders might exploit equipment and electricity for active mining. We categorize active mining into two types: solo mining and pool mining. Solo mining involves individuals or small groups connecting directly to a blockchain's full node using specialized software [8]. However, hashrate of individuals is relatively negligible to the entire network's, which means it is nearly unprofitable for miners. Thus, mining pools offer a collaborative platform for miners to combine their hashrate, enhancing the chances of solving the blockchain puzzles. Miners usually need to obtain authorization from mining pools and then mine through the pool. Communication between miners and the mining pool occurs in three primary ways [9]: direct connection to the pool, using a proxy server, or employing an anonymous network service like Tor or VPN. The mining pool allocates revenue based on each miner's hashrate contribution. When a blockchain puzzle is successfully solved, the pool distributes rewards proportionally to the miners based on their individual contribution.

Meanwhile, external attackers often target these devices in campus, potentially commandeering them for cryptojacking operations [10]. Cryptojacking is a malicious activity involving the unauthorized use of victims' device for cryptomining [11]. It manifests in two forms: browser-based cryptojacking and binary-based cryptojacking. Browser-based cryptojacking involved loading JavaScript or WebAssembly scripts in browsers, enabling mining by miners. Despite Coinhive, a dominant browser-based mining service provider, shutting down in March 2019 [12], browser-based cryptojacking persists [13]. Attackers embed mining scripts in websites, coercing visitors' devices into joining botnets for cryptomining. Binary-based cryptojacking is more covert. Binary malware clandestinely embeds itself in the victim's system [5], automatically manipulating the device into executing cryptomining tasks. Futhermore, these binary malwares aim to launch at system startup [9], masquerading as legitimate background processes to mine continuously without detection.

To safeguard their network, campus often rely on simple and convenient methods, such as configuring firewall rules to block access to blacklisted URLs of mining pools and using Deep Packet Inspection (DPI) technology to detect and impede non-encrypted Stratum connections. However, this blacklist-based approach demands continuous maintenance and updates due to the ever-changing landscape of mining pool URLs. Maintaining an effective blacklist requires a stable and reliable source of threat intelligence, which might involve gathering public intelligence or deriving mining pool service URLs information from DPI detection results.

Several challenges arise from these methods. Firstly, the increasing use of proxy and private mining pools complicates the acquisition of blacklists [14]. Additionally, for security reasons, mining pools often change the URLs listed in blacklists for safety. Furthermore, DPI-based detection methods struggle to identify encrypted traffic, which poses a significant problem as most mining pools now offer SSL/TLS encryption services. Last but not least, supervisors often need more information about cryptomining behaviours to apply different governance

measures. It means that campus needs a robust detection scheme that can detect encrypted cryptomining traffic and provide with more information related to mining behaviours.

B. Related Work

Current cryptomining behavior detection schemes primarily base on host behavior [15], [16], browser behavior [17]-[19] and machine learning applications [20]-[23]. Host behavior-based detection schemes try to spot cryptomining by looking for unusual host behaviors, e.g., CPU or memory usage. However, latest mining softwares can config a modest hashrate used for cryptomining by setting "throttle value". Also, these methods often need to install special detection tools on each device, which is expensive and impractical. Browser-based detection methods focus on different indicators, like WebAssembly (Wasm) or JavaScript (JS) file characteristics and their compilation times. These methods need access to Wasm, JS, and other files created during web browsing. They are good at finding cryptomining behaviors done through browsers but they can't detect cryptomining that happens outside of browsers.

Machine learning-based schemes are particularly relevant for most cryptomining detection scenarios today, especially given the widespread use of SSL/TLS encryption by mining pools and mining softwares, which poses a challenge for traditional DPI-based detection methods. Several researchers have contributed to this field. In [20], they analyzed traffic from IoT devices running malicious mining software in smart home networks, applying classification models like SVM and GNB for cryptomining traffic detection. In [21], they utilized NetFlow/IPFIX flow measurement techniques, using features like outgoing and incoming packet flow rates to differentiate cryptomining traffic from the normal. As for [22], the authors employed 51 features extracted by the Tstat tool to train a classification model and evaluate it in realistic scenarios. In [23], the authors proposed and proved the robustness of *Crypto-Aegis*, a machine learning framework adept at detecting unauthorized cryptomining behaviors related to the sponge-attack.

However, existing encrypted cryptomining traffic detection schemes still have some shortcomings. Primarily, most current schemes only provide binary classification methods for distinguishing encrypted traffic as mining or non-mining, which makes it difficult for supervisors to obtain fine-grained information about cryptomining behaviours, such as the crypto-currencies being mined. Furthermore, existing schemes for cryptomining traffic detection tend to have a high false positive rate, but they usually lack solutions to solve this problem. This will undoubtedly reduce the usability of these schemes in real-world scenarios.

C. Contributions

To address the identified challenges in cryptomining traffic detection, this paper makes the following major contributions:

- A two-stage encrypted cryptomining traffic detection mechanism based on machine learning and active probing. Compared to existing detection schemes, the mechanism is able to detect encrypted cryptomining traffic while combining active probing to effectively reduce the model's false positive rate, which is more practical in real-world scenarios.
- A fine-grained cryptomining traffic detection scheme based on time series features and machine learning. By extracting optimal time series features for training, our classifier can detect mining traffic with an 0.99 F1 score on the collected dataset and identify the cryptocurrency being mined with 99.39% correct recognition rate.
- An active probing scheme for identifying mining pools. It involves constructing specific requests and parsing responses. Futhermore, we have conducted extensive evaluations of the scheme to prove its effectiveness.

The rest of the paper is organized as follows: Section II describes the threat model and we will complete following experiments in that scenario. In section III, we describe our methodology in detail, including our two-stage detection mechanism, as well as a fine-grained cryptomining traffic detection scheme based on machine learning. Section IV describes our active probing scheme and the evaluation of it. Section V finalizes the paper with the discussion and conclusion.

II. Threat Model

A Cisco's report says that college campuses are the second biggest miners of cryptocurrencies behind the energy and utilities sector [30]. This is probably due to the fact that there are many high-performance devices suitable for cryptomining on campus, such as personal computers (PCs) and servers. On the one hand, students can use these devices as well as free electricity to mine for benefits, and on the other hand, attackers may use malware or malicious websites to control these devices for mining. Therefore, we constructed our threat model within a controlled lab in the campus network. Firstly, the model is based on the following assumptions:

- Each device connects to the external world through an internal gateway.
- All miners focus on mining profitable cryptocurrencies. It means that these cryptocurrencies are more prominent in the field of cryptomining.
- All cryptomining traffic uses SSL/TLS encryption as we focus on the detection of encrypted traffic.
- There are no mining pools within the LAN. All mining pools that devices connect to are external.

Meanwhile, as shown in Fig. 1, miners are categorized into active miners and cryptojacked devices. Active miners use the lab's resources (e.g., PCs, and servers) to connect to mining pools for cryptomining. Solo mining is not included because it is almost unprofitable. Cryptojacked Devices are those that have been compromised by attackers to perform cryptomining. As previously mentioned, attackers may perform browser-based and binary-based cryptojacking on victim devices. Additionally, external mining pools are classified into three categories [14]: public pools, proxy pools, and private pools. Thus, both active miners and cryptojacking devices are connected to public or private mining pools, either directly or indirectly through proxy mining pools. Based on the above, this model can adequately summarise the various mining behaviours under the campus network. We aim to detect the encrypted cryptomining traffic generated by these cryptomining behaviours in this scenario.

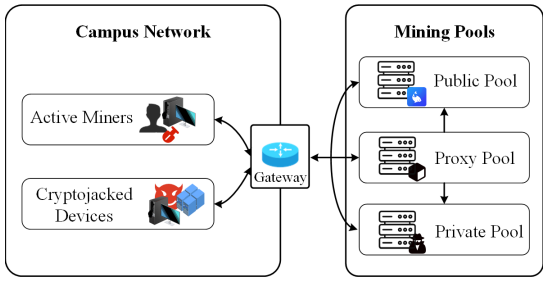

Fig. 1. Different cryptomining behaviours in campus network.

III. Methodology

In this section, we describes our methodology for encrypted cryptomining traffic detection for campus network. We propose a two-stage encrypted cryptomining traffic detection mechanism in subsection *A*. Then, we provide a detailed description of the fine-grained machine learning-based encrypted cryptomining traffic detection scheme. First, subsection *B* describes the collection of the dataset. Further, subsection *C* describes the extraction and selection of time series features. After that, the model selection and training results are described in subsection *D*. Finally, subsection *E* describes a fine-grained detection scheme for detecting cryptocurrency being mined with training results.

A. Two-stage Detection Mechanism

Encrypted cryptomining traffic breaks through the limitations of DPI-based detection techniques. Identifying this type of traffic requires a specific technique or a mechanism that integrates multiple detection methods. Thus, we propose a two-stage mechanism with fine-grained machine learning-based encrypted cryptomining traffic detection and active probing. This section focuses on the former and the corresponding experiments and evaluation results.

For traffic that passes through the campus gateway, the campus network usually uses blacklists, DPI-based or other detection schemes to quickly classify the traffic. They are identified as cryptomining traffic if they contain suspicious destination addresses or mining-related message content. However, there is no certainty that "normal" encrypted traffic is innocent. Therefore, we can use our two-stage mechanism for further detection. As shown in Fig. 2, we extract features from "normal" encrypted traffic and classify it with a trained classifier. The classifier is able to indicate whether the traffic is cryptomining traffic or not, and further gives information about the coins being mined. From the result of traffic classification, we can get a list of suspicious addresses which are the destination addresses and ports obtained from the traffic classified as cryptomining traffic. We use these suspicious addresses as inputs to the active probing module in the second step (2). The module will probe these addresses based on request construction and response prasing. Eventually, based on the probing results, we can efficiently identify encrypted cryptomining traffic and update the blacklists at the same time.

B. Dataset Collection

In this experiment, we focused on two datasets: normal traffic and cryptomining traffic. The latter is divided into active mining traffic and cryptojacking traffic. The former originates from direct connections to mining pools. Based on findings from [11] and [18], which highlight Monero as a frequent target in cryptojacking, our cryptojacking traffic dataset centers on Monero mining via browser and binary malware.

The experiment was conducted in a controlled campus lab network. Firstly, we deployed multiple virtual machines in bridged mode using VMware in each PC and server. Second, we deployed different mining software in each VM to perform cryptomining of different cryptocurrencies. Eventually, we run multiple VMs to perform cryptomining in parallel and efficiently capture the mining traffic at the gateway using Wireshark software. Due to the low volume of traffic generated by cryptomining, all cryptomining behaviors lasted at least 6 hours. This approach ensured a thorough dataset for our analysis. It is worth mentioning that before the mining process started, we had reported to the administration and obtained their permission to minimise the impact on the campus network.

Eventually, we gathered cryptomining traffic for 7 different cryptocurrencies (shown in Table I). For active mining, we selected 7 categories of cryptocurrencies from the top 20 as of January 2023, according to [32]. These cryptocurrencies are popular among miners for their profitability. In contrast, our cryptojacking mining traffic dataset includes two main sources: browser-based and binary-based cryptojacking. We collect JavaScript code from existing browser-based mining services and incorporate it into our private websites. We then simulate browser-based cryptojacking behaviours by browsing these pages. For binary-based cryptojacking, we used public dataset from [20], which used MinerGate to replicate malware intrusion scenarios. This dataset captures the entire traffic flow from the

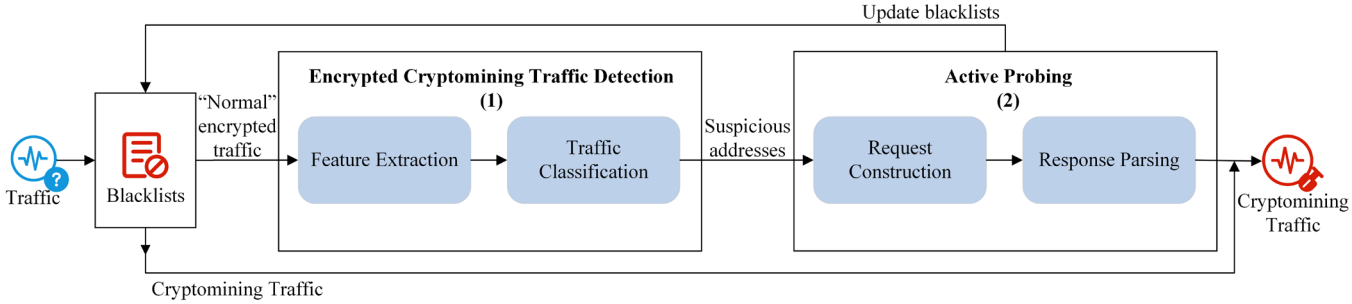

Fig. 2. The two-stage mechanism of encrypted cryptomining traffic detection.

TABLE I. DATASET OF CRYPTOMINING TRAFFIC

Type	Crypto-currency	Cryptomining service provider	Total Time (minutes)	Packet Count
Active mining	BTC	F2pool	296	288,440
	XMR	C3pool	1,597	65,152
	ETC	F2pool	3,174	55,296
	ETHW	JNpool	2,361	84,736
	ETF	JNpool	1,237	63,744
	CFX	JNpool	838	50,240
	RVN	JNpool	6,133	81,024
Crypto-jacking	XMR	Webminepool (browser-based)	2,391	27,384
		Webmine (browser-based)	1,879	27,874
		MinerGate (binary-based)	984	22,111
Total				**766,001**

TABLE II. DATASET OF NORMAL TRAFFIC

Type	Name	Total Time (minutes)	Packet Count
Normal online behavior	Online Chatting	37	231,768
	Online Gaming	12	213,306
	Online Music	15	201,527
	Website Browsing	58	211,504
Total			**858,105**

beginning to the end of cryptomining behaviors on victim devices.

For the normal traffic dataset, we captured encrypted traffic data from 4 typical online behaviors within the campus network: online chatting, web browsing, listening to online music, and online gaming. These collection details are presented in Table II. Due to the voluminous nature of normal traffic, we restricted each behavior to a maximum duration of 1 hour.

For the collected dataset, we preprocess the dataset before extracting features, includes filtering the traffic data based on five-tuple and normalizing data. Since the main objective of this experiment is to detect the cryptomining traffic, we categorize the cryptomining traffic as positive samples and the normal traffic as negative samples. To ensure unbiased model training, we maintain a similar count of positive and negative samples. Lastly, for the experiment's subsequent phases, we split these samples into training and testing sets at a 4:1 ratio.

C. Feature Extraction

The use of SSL/TLS by mining softwares and pools enhances communication security and impedes threat intelligence-based detection methods. However, distinct patterns emerge upon analyzing normal traffic and encrypted cryptomining traffic. As evidenced in [24] and confirmed in our collected dataset, about 40% of mining packets range from 36 to 80 bytes, and over 50% fall between 105 to 110 bytes. This contrasts sharply with normal traffic, where less than 10% of packets between 105 to 110 bytes. Beyond packet length, cryptomining traffic differs significantly from normal traffic in terms of time series characteristics, e.g., flow rate, peak, and the mean of packet lengths. Consequently, extracting time series features from traffic is a viable and effective approach to differentiate normal traffic and cryptomining traffic.

Approximately, we can view traffic packets as time series that are continuously distributed over timestamps. Thus, we extract time series features from collected dataset based on three important pieces of information about traffic：

- **Timestamps**: Timestamps indicate packet order and timing, crucial in identifying cryptomining traffic, which often follows specific temporal patterns such as persistent high traffic or periodicity, unlike the temporal characteristics of nomal traffic.
- **Five-tuple**: Five-tuple includes source and destination IP addresses, source and destination ports, and protocol type. This information is key in recognizing network connections. Miners' frequent communication with specific servers, often identifiable by unique destination IPs and ports. Furthermore, the protocol can further clarify the traffic, as different cryptomining behaviors tend to use varied protocols.
- **Packet Length**: Packet length is an important input for calculating time series features along with timestamps. Cryptomining traffic often involves intensive data exchange, leading to unique packet length distributions. The temporal aspect of packet length can elucidate changing traffic patterns over time, potentially revealing differences in periodicity, burstiness, or continuity when compared with normal traffic. burstiness, or continuity when compared with normal traffic.

In our analysis, we divide the traffic dataset into different streams according to the five-tuple. Then, each stream was segmented into subgroups of 10 consecutive packets $P = \{ pkt_1, pkt_2, \ldots, pkt_{10}\}$. Utilizing the tsfresh [26], we performed a relevance analysis on all the features and selected statistically significant features with smaller P-value (less than 0.01) as optimal features. Finally, we selected 231 optimal features (partly shown in Table III).

The selected features are important to uncover the distinct characteristics of cryptomining traffic from various angles.

TABLE III. A PART OF OPTIMAL FEATURES AND THEIR DESCRIPTIONS.

Feature Name	Description
Sum-values	Total length of packets within the observation window.
Number-cwt-peaks	Number of different peaks in the packet length sequence.
Binned-entropy-max-bins-10	Entropy value of the packet length distribution divided into 10 bins.
Index-mass-quantile-q-0.1	The mass index corresponding to the 10% quantile of the packet length distribution.

These features aim to capture unique patterns inherent in cryptomining behaviors. For instance, the total length of packets (*Sum-values*) provides insights into the data-intensive nature of cryptomining traffic. The *Number-cwt-peak*, derived from Continuous Wavelet Transform (CWT) , highlights the periodic and bursty patterns of cryptomining traffic. It also reflect the synchronization between computational and communication processes. Additionally, the entropy value of packet length distribution (*Binned-entropy-max-bins-10*) and the quartile index quality (*Index-mass-quantile-q-0.1*) shed light on the regularity or concentration of packet length distribution in cryptomining traffic.

Overall, these features clearly show how cryptomining traffic is different from normal traffic. They help us see the unique patterns in packet lengths, how often packets are sent, the number of peaks, and the way packet lengths vary. These differences are key to telling cryptomining traffic apart from normal traffic. By focusing on these time series features, we can better identify cryptomining traffic. This makes our detection both more accurate and faster.

D. Machine Learning Models

In our study, we focused on evaluating three advanced Machine Learning models: Gradient Boosting Machine (GBM) , Random Forest (RF) and eXtreme Gradient Boosting (XGBoost) [27]. For all models, we utilized the default settings provided by scikit-learn or dedicated library for XGBoost.

To evaluate the effectiveness of these models, we use 5-fold cross-validation. For the binary classification task in crypto-mining traffic detection, we selected several evaluation metrics: Accuracy, Precision, Recall, F1 Score, and AUC. Among these, Accuracy is particularly relevant in our study with a balanced dataset, as it reflects the proportion of samples the model correctly classified. Precision assesses the proportion of true positive samples correctly identified. Recall measures how well the model identifies all true positive samples. The F1 Score, as the harmonic mean of Precision and Recall, summarizes the ability of the model to identify positive samples. In addition, AUC provides insights into the model's ability to distinguish between positive and negative samples without being tied to a specific classification threshold.

Table IV shows calculated accuracy metrics illustrating model's performance against this dataset. It shows that XGBoost performs better than the other models across all metrics. Specifically, XGBoost achieves a precision, recall, and F1 Score of 0.99, with an AUC nearing 0.998. In contrast, GBM and Random Forest exhibit relatively lower performance in this experimental context, highlighting significant advantages of XGBoost. Unlike Random Forest and GBM, which are also ensemble learning models, XGBoost benefits from a more intricate model structure. This complexity enables XGBoost to adapt more effectively to varying data characteristics, especially in scenarios involving incomplete or noisy data. Therefore, XGBoost's robustness and adaptability make it more suitable for cryptomining traffic detection.

TABLE IV. OVERALL ACCURACY METRICS AGAINST DATASET

Model	Accuracy	Precision	Recall	F1 Score	AUC
GBM	0.97	0.97	0.97	0.97	0.988
RF	0.98	0.99	0.98	0.99	0.998
XGB	**0.99**	**0.99**	**0.99**	**0.99**	**0.999**

TABLE V. HYPERPARAMETERS OF XGBOOST AFTER BAYESIAN OPTIMISATION

Parameter Name	Value
gamma	0.01
max_depth	4
subsample	0.819
colsample_bytree	0.514
min_child_weight	5
learning_rate	0.409

E. Detection of Different Cryptocurrency

Different cryptocurrencies have some differences in mining algorithms and protocols, which obviously affect traffic features. For example, Bitcoin uses SHA-256 as its mining algorithm while Monero uses RandomX to allow ordinary hardware (e.g., CPUs) to participate in mining. We will analyze the differences of different cryptocurrencies in mining protocols in Section IV. There is no doubt that mining traffic will differ for these reasons.

Thus, we extend our scheme to include cryptomining traffic data for seven different cryptocurrencies, as outlined in subsection A. Similarly, we extract 182 optimal time series features for the cryptomining traffic using the feature extraction method in subsection B. Then, we employ the XGBoost model for a multi-classification task to discern specific crypto-currencies from cryptomining traffic.

The performance of the XGBoost model significantly depends on the choice of its hyperparameters. Therefore, tuning these hyperparameters is crucial for optimizing the model. Considering the size of the hyperparameter space for this experiment, we apply a Bayesian optimization algorithm, which offers a notable advantage over traditional methods like random or grid search. Its ability to prune the search space and use historical data to construct a Gaussian process model enables it to identify the globally optimal solution with fewer iterations.

We use softmax as the loss function for XGBoost. To validate the model's effectiveness and mitigate the risk of overfitting, we also implement five-fold cross-validation during the training and validation stages. This approach allows us to calculate mlogloss, as in:

$$mlogloss = \frac{-1}{N}\sum_{i\in N}\sum_{j\in M} y_{ij}\, log(\, p_{ij}\,) \qquad (1)$$

In (1), N represents the total number of samples in the dataset, and M indicates the number of categories, which is seven for this study. The term y_{ij} is a binary indicator, assigned 1 if sample i belongs to category j, and 0 otherwise. The term p_{ij} represents the model's predicted probability that sample i belongs to category j. We continuously adjust the range of hyperparameters and the number of Bayesian optimization iterations. Through repeated training and multiple experimental sets, we select the

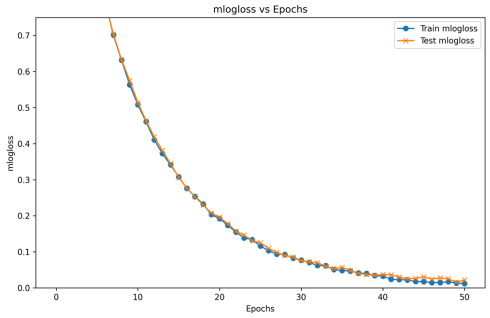

Fig. 3. Variation of mlogloss of XGBoost model with epochs on training and test sets.

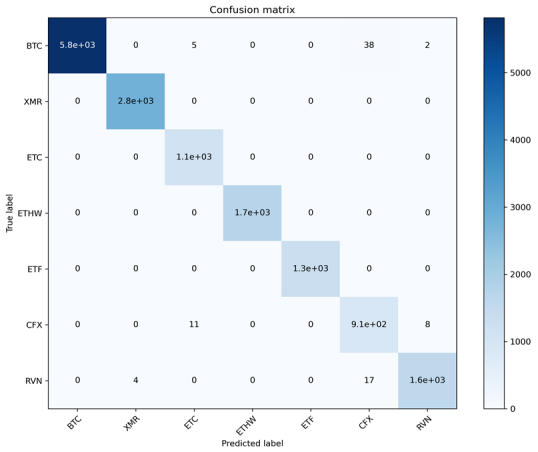

Fig. 4. Confusion matrix for the test set of the cryptocurrency detection experiment.

optimal results for model construction. We carried out 10 separate experiments on the dataset and calculated the average of the individual hyperparameters obtained. The best parameter combination is presented in Table V.

Using the optimal hyperparameter combinations, Fig. 3 shows how the mlogloss of the XGBoost model varies with the number of training epochs for both training and test sets. Notably, the model reaches a low mlogloss of 0.079 for the training set and 0.083 for the test set after 30 epochs. This result demonstrates the model's strong capability to both fit and generalize well across the training and test data. Additionally, the model achieves a 99.39% correct recognition rate after 50 epochs, and the confusion matrix is shown in Fig. 4. This result further validates the model's accuracy and effectiveness for different mining cryptocurrencies.

IV. Active Probing

In Section III, we demonstrate schemes to classify encrypted network traffic with machine learning models. These models effectively distinguish between normal and cryptomining traffic and identifies specific cryptocurrencies involved in crypto-mining. However, it already has a certain False Positive Rate (FPR), where normal traffic is incorrectly classified as cryptomining behavior. To mitigate this issue in this section, we propose an active probing scheme based on request construction and response parsing to detect mining pool service.

A. Analysis of Mining Protocol

To construct suitable request messages, it is essential to understand the details of the communication between miners and mining pools. This involves grasping the message format and definitions as outlined by the mining protocol. Thus, We conducted a comprehensive analysis of mining protocols. Our approach is grounded in a thorough investigation of related papers (e.g., [9] and [25]), source code from [28]-[30], and collected cryptomining traffic datasets. We found that Stratum protocol [25] is widely adopted in cryptomining for cryptocurrencies like Bitcoin, Monero, and Ethereum. Stratum protocol is based on the TCP protocol and enhanced by JSON-RPC2.0. This combination addresses key communication challenges in the cryptomining sector, notably improving the interaction and efficiency of cryptomining.

The protocol is mainly structured around four message formats: miner subscription, miner authentication, mining job notification, and share submission [25]. These formats facilitate a standardized collaboration process between mining pools and miners, allowing for the efficient distribution and completion of mining jobs. Our active probing method focuses on different specific implementations of the first three messages. Before cryptomining, miners usually need to send subscription and authentication messages to establish a connection with the mining pool. Then, the mining pool will return a response or mining job after possible authentication. The standardized communication is utilized in both active mining and cryptojacking scenarios. While the implementation of the Stratum protocol may vary slightly among different crypto-currencies and mining pools, the fundamental pattern of the protocol remains consistent. For example, Monero have adopted a refined version of the Stratum protocol, further optimizing the handshaking process for more efficient cryptomining operations (shown in Fig. 5).

We focus on the Stratum protocols used by three major cryptocurrencies: Bitcoin, Monero, and Ethereum. Thus, we analyze their messages about miner subscription with each corresponding success and error responses, as illustrated in Table VI. By the way, there are two versions of the Stratum protocol, StratumV1 and StratumV2. However, StratumV2 is an unpractised protocol, it is only used by a few pools such as [33]. Consequently, most mining pools and miners continue to use StratumV1. Therefore, we only focus on the StratumV1 protocol.

B. Active Probing Method

Our active probing method is based on subscription message construction and response parsing. For each target URL, we construct subscription messages for Bitcoin (Stratum-BTC), Monero (including Stratum-XMR and Stratum-Webmine-XMR), and Ethereum (Stratum-ETH) as shown in Table VI.

Before sending the message, we try to establish a TCP connection with the URL to confirming its availability. If it is confirmed to be alive, we continue our probing method. Then, we send our constructed subscription messages by following methods: i) **HTTP** for traditional plaintext transmission using

TABLE VI. DIFFERENT SUBSCRIPTION MESSAGES IN THE STRATUM PROTOCOL, EACH WITH ITS CORRESPONDING SUCCESS AND ERROR RESPONSES

Protocol	Subscription message	Success response	Error response
Stratum-BTC	{"id": 1,"method":"mining.subscribe", "params": []}	{**"id"**: 1, **"result"**:[["mining.set_difficulty", "<Target difficulty >"],["mining.notify", "<Wallet address>"]],**"error"**: null}	{**"id"**:1,**"result"**:false, **"error"**:[20,"Not supported",null]}
Stratum-XMR	{"id":1,"jsonrpc":"2.0","method": "login","params":{"login":"<Wallet address>","pass":"x",}}	{**"id"**:1,"jsonrpc":"2.0","**result**":{**"id"**:"<Miner ID>", **"job"**:{**"algo"**:"rx/0","**blob**":"<Blob>", **"target"**:"<Target difficulty>"}, **"status"**: "OK"},**"error"**:null}	{ **"id"**: 1, "jsonrpc": "2.0","**error**": {"code": -1, **"message**": "Invalid address" } }
Stratum-ETH	{"id": 1,"jsonrpc":"2.0","method": "eth_submitLogin", "params":[<Wallet address>]}	{ **"id"**: 1, "jsonrpc": "2.0", **"result**": true }	{ **"id"**: 1,"jsonrpc": "2.0","**result**": null,**"error"**: {**code**: -1, **message**: "Invalid login"}}
Stratum-Webmine-XMR	{"id":"start","m":"start","p":{"token":"< Miner token>","subscribe":[]}}	{**"r"**:{"subscribed":[]},**"id"**:"start"}	{**"e"**:"noRights","**id**":"start"}

JSON-RPC2.0; ii) **HTTPS** for encrypted cryptomining traffic via SSL/TLS protocol; iii) **WebSocket** for communication in browser/server mode and commonly used for browser-based mining. All of these are TCP-based protocols, which is why we need to check if we can establish a TCP connection with the target URL first. We send these messages using HTTP and HTTPS for Stratum-BTC, Stratum-XMR, and Stratum-ETH. Websocket was reserved exclusively for Stratum-Webmine-XMR, as it is predominantly used in browser-based mining, with Monero being a principal cryptocurrency.

Our analysis focused on specific fields in the success and error responses. We find that standard Stratum protocol responses typically include key fields (e.g., *id*, *result*, and *error*). Utilizing this pattern, we match fields within the responses to check for these common elements. The presence of these fields allows us to determine that the probed URL is offering a mining pool service.

C. Evaluation

To evaluate the effectiveness of our active probing method, we collect a list of mining pool service URLs from various types of pools, including public, proxy and private pools. The dataset for these URLs are detailed as follows:

Public Mining Pools: These pools provide services to miners via publicly disclosed service URLs. We refer to mining pool intelligence from [32] as of November 2023 to identify the top 10 mining pools for Bitcoin, Monero, and Ethereum series (Ethereum Classic, Ethereum PoW), based on their hashrates. We found that the total hashrate of the top 10 pools for each cryptocurrency represents more than 92% of their respective blockchain networks. Finally, we registered with these pools and collected a total of 124 URLs for probing, which differed in different cryptocurrencies, different service regions, whether or not they used SSL/TLS encryption, and so on.

Proxy Mining Pools: To assess our active probing scheme, we also focus on proxy mining pools. These pools offer traffic forwarding services for miners to address security issues linked to direct pool connections. We gathered proxy pools' service URLs using two approaches: self-establishing proxy mining pools and extracting C&C server URLs from available browser-based mining scripts.

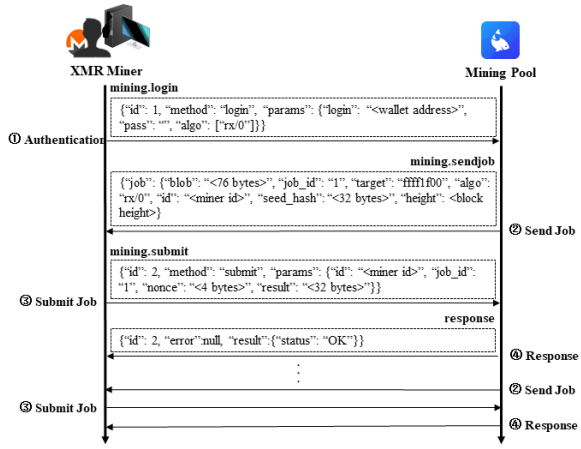

Fig. 5. Monero's Stratum Protocol (ignore some unimportant fields).

We collected proxy mining pool softwares through various public channels. Firstly, we discovered that some public pools, which we had previously identified, independently develop proxy pool software for miner. From these, we collected proxy softwares from three public pools: [38]-[40]. Secondly, we investigated public code repositories such as Github [40], disregarding those with less than 50 stars or with high similarity. Finally, we collected 7 representative mining pool proxy softwares. Among these, [45]-[49] support BTC and ETC, while [25] and [50] are exclusive to XMR. All softwares support Stratum traffic encryption and traffic forwarding. We deployed all the collected softwares on our VPS servers. For those proxies supporting multiple cryptocurrency types simultaneously, we configured different ports to offer distinct proxy services for each cryptocurrency type.

Additionally, for browser-based mining services, we gathered publicly available JS script samples still offering browser-based mining services and extracted C&C server URLs. This approach yielded three proxy mining pool service URLs from [42], [43], and [44].

Eventually, we collected 16 URLs from 13 different proxy mining pools for active probing.

Private Mining Pools: We collected about 1,200 code repositories associated with mining pool software from Github. We ingore those repositories with i) less than 100 stars; ii) high similarity to top-ranked softwares; iii) source code published by public mining pools; iv) no updates for over a year; and v) lack of support for cryptocurrencies in the Bitcoin, Monero, and Ethereum series. Ultimately, we identified and deployed [35], [36], and [37] that fully met our criteria. We opened different ports to support diverse mining services, including different cryptocurrencies and whether to use SSL/TLS. As a result, we established 9 unique private mining pool service URLs for probing.

Results and findings: Totally, we collected 149 mining pool service URLs. We probed this dataset using our active probing method based on request construction and response parsing. Ultimately, our probe successfully received success or error responses from 147 different mining pool service URLs.

Futhermore, we have also made some new findings following our detailed analysis of the mining pool software and the results of active probing. First, the pools which were not successfully probed all belonged to browser-based mining pool services. It seems that these pools check the API token or site key information in the subscription message to verify the legitimacy of the request, and they will not respond to any messages with unregistered API token or site key. This means that detecting browser-based mining pool services require constructing request messages specific to different pools. In addition, for the public and private pools we collected, they do not take "no response" measures to respond to incorrect requests but at least return an error response to inform miners. However, we have found from the source code of some private mining pools [35] that they can avoid this active probing by setting a blacklist or maximum number of connections. By this way, the pool does not respond to blacklisted miners, and only sets a maximum number of connections that is just enough to reject unanticipated illegal probing requests. However, by default, the blacklist is empty and the maximum number of connections is large enough for the private and proxy pools we deployed.

V. Discussion and Conclusion

As described in Section III, we propose a two-stage encrypted cryptomining traffic detection mechanism in campus networks. The mechanism not only effectively solves the problem of detecting encrypted cryptomining traffic for the existing schemes in campus network, but also reduces the false positives caused by the existing detection schemes. After that, we also propose a fine-grained cryptomining traffic detection scheme. We investigate the coins that are still profitable and collect corresponding active mining and cryptojacking traffic in a controlled environment. By using optimal time series features to train the classifiers, as demonstrated in Table IV and Fig. 4, our XGBoost classifiers achieves an F1 score of 0.99 in the mining traffic detection scenario and achieve a 99.39% correct recognition rate in multi-class classification scenarios for different cryptocurrencies. Furthermore, unlike existing schemes, we propose and evaluate an active probing scheme to reduce the FPR of the classifiers in Section IV. We analyse the different implementation of Stratum protocol to construct request messages and prase responses. Eventually, the results show that our method can detect 98.66% of the mining pool URLs in the collected dataset except for two browser-based mining pool services.

While our study advances the detection of encrypted cryptomining traffic, it is important to acknowledge some limitations. The collected traffic dataset does not cover all the mining behaviours associated with different cryptocurrencies. It could potentially affect the model's generalizability to diverse real-world scenarios. In addition, while relying on time series features is effective, these features may be confounded by traffic obfuscation techniques, which may be used by tricky miners. Such techniques could mask cryptomining behaviors, thereby challenging the accuracy and effectiveness of existing detection schemes.

References

[1] Nakamoto Satoshi, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized business review, 2008.

[2] Coinwarz, “Bitcoin hashrate,” 2023. [Online]. Available: https://www.coinwarz.com/mining/-bitcoin/hashrate-chart#

[3] M. Jabłczyńska, K. Kosc, P. Ryś, P. Sakowski, R. Ślepaczuk, and G. Zakrzewski, “Energy and cost efficiency of Bitcoin mining endeavor,” *PloS one*, vol. 18, no. 3, 2023.

[4] P . Papadopoulos, P . Ilia, and E. Markatos, “Truth in web mining: Measuring the profitability and the imposed overheads of crypto-jacking,” in *International Conference on Information Security (ISC)*, Springer, pp. 277–296, 2019.

[5] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, et al., “How You Get Shot in the Back: A Systematical Study about Cryptojacking in the Real World,” in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 13, 2018.

[6] A. Zareh, H. R. Shahriari, “Botcointrap: detection of bitcoin miner botnet using host based approach,” in *2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, IEEE, pp. 1-6, 2018.

[7] Y. Cheng, Z. Jin, and W. Ding, “Analysis of the Propagation of Miner Botnet,” in *2022 6th International Conference on Cryptography, Security and Privacy (CSP)*, pp. 96-101, 2022.

[8] xmrig, “XMRig for Monero mining,” 2023. [Online]. Available: https://github.com/xmrig/xmrig

[9] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "SoK: cryptojacking malware," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 120-139, 2021.

[10] P . H. Meland, B. H. Johansen, and G. Sindre, “An experimental analysis of crypto-jacking attacks,” in *Nordic Conference on Secure IT Systems*. Springer, pp. 155–170, 2019.

[11] S. Eskandari, A. Leoutsarakos, T. Mursch and J. Clark, “A First Look at Browser-Based Cryptojacking,” in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, pp. 58-66, 2018.

[12] CryptoGlobe, “Crypto Mining Service Coinhive to Shut down over Market Downturn, XMR Updates,” 2019. [Online]. Available: https://www.cryptoglobe.com/latest/2019/02/crypto-mining-service-coinhive-to-shut-down-over-market-downturn-xmr-updates/

[13] S. Varlioglu, B. Gonen, M. Ozer and M. Bastug, “Is cryptojacking dead after coinhive shutdown?,” in *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, IEEE, pp. 385-389, 2020.

[14] Z. Zhang, G. Hong , X. Li, Z. Fu, J. Zhang, M. Liu, et al. “Under the dark: a systematical study of stealthy mining pools (ab) use in the wild,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 326-340, 2023.

[15] D. Tanana, “Behavior-based detection of cryptojacking malware,” in *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, IEEE, pp. 543-545, 2020.

[16] M. Conti, A. Gangwal, G. Lain, and S. G. Piazzetta, "Detecting covert cryptomining using hpc," in *International Conference on Cryptology and Network Security (CANS)*, pp. 344-364, 2020.

[17] R. K. Konoth, E. Vineti, V . Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, and G. Vigna, "Minesweeper: An in-depth look into driveby cryptocurrency mining and its defense," *in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1714–1730, 2018.

[18] A. Kharraz, et al., "Outguard: Detecting in-browser covert cryptocurrency mining in the wild," *in The World Wide Web Conference (WWW)*, pp. 840-852, 2019.

[19] F. N. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac, "MINOS: A Lightweight Real-Time Cryptojacking Detection System," in *the Network and Distributed System Security Symposium (NDSS)*, pp. 1-15, 2021.

[20] E. Tekiner, A. Abbas, and A. S. Uluagac, "A lightweight IoT cryptojacking detection mechanism in heterogeneous smart home networks," *in Network and Distributed System Security Symposium (NDSS)*, 2022.

[21] J. Z. i Muñoz , J. Suárez-Varela, and P. Barlet-Ros, "Detecting cryptocurrency miners with NetFlow/IPFIX network measurements," in *International Symposium on Measurements and Networking (M&N)*, IEEE, pp. 1-6, 2019.

[22] A. Pastor et al., "Detection of Encrypted Cryptomining Malware Connections With Machine and Deep Learning," in IEEE Access, vol. 8, pp. 158036-158055, 2020.

[23] M. Caprolu, S. Raponi, G. Oligeri, and R. Di Pietro, "Cryptomining makes noise: Detecting cryptojacking via machine learning," *Computer Communications*, pp. 126-139, 2021.

[24] V. Veselý, M. Žádník, "How to detect cryptocurrency miners? By traffic forensics!," *Digital Investigation*, vol. 31: 100884, 2019.

[25] Braiins, "StratumV1 Protocal," 2023. [Online]. Available: https://zh.braiins.com/stratum-v1/docs

[26] blue-yonder, "Tsfresh tools," 2023. [Online]. Available: https://github.com/blue-yonder/tsfresh

[27] T. Chen, and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785-794, 2016.

[28] edsonayllon, "Stratum implementation for Pantheon," 2020. [Online]. Available: https://github.com/edsonayllon/Stratum-Implementation-For-Pantheon#4-stratum-implem-entation

[29] sammy007, "Stratum Mining Protocol in open-ethereum-pool," 2017. [Online]. Available: https://github.com/sammy007/open-ethereum-pool/blob/master/docs-/STRATUM.md

[30] pooler, "CPUminer," 2021. [Online]. Available: https://github.com/-pooler/cpuminer

[31] xmrig, "XMRig-proxy," 2023. [Online]. Available: https://github.com-/xmrig/xmrig-proxy

[32] Miningpoolstats, "Mining pool information," 2023. [Online]. Available: https://miningpoolstats.str-eam/

[33] Braiins, "braiins pool," 2023. [Online]. Available: https://zh.braiins.com/pool

[34] PCMag, "PCMag's news," 2019. [Online]. Available: https://www.pcmag.com/news/college-kids-are-using-campus-electricity-to-mine-crypto

[35] sammy007, "Open Ethereum pool," 2017. [Online]. Available: https://github.com/sammy007/open-ethereum-pool

[36] jtgrassie, "Monero pool," 2023. [Online]. Available: https://github.com/-jtgrassie/monero-pool

[37] Miningcore, "Miningcore mining pool," 2023. [Online]. Available: https://github.com/oliverw/miningcore

[38] btccom, "BTC Smart Agent," 2022. [Online]. Available: https://github.com/btccom/btcagent

[39] Braiins, "Braiins Pool Proxy," 2023. [Online]. Available: https://zh.braiins.com/farm-proxy

[40] Viabtc, "Viabtc's Pool Proxy," 2021. [Online]. Available: https://github.com/viabtc/mineragent

[41] Github, "Public code repository," 2023. [Online]. Available: https://github.com/

[42] Webminepool, " browser-based mining service provider," 2023. [Online]. Available: https://www.webminepool.com/page/documentation

[43] Webmine, "browser-based mining service provider," 2023. [Online]. Available: http://webmine.cz/

[44] Browsermine, " browser-based mining service provider," 2023. [Online]. Available: https://cp.browsermine.com/

[45] EvilGenius-dot, "RustMinerSystem" 2023. [Online]. Available: https://github.com/EvilGenius-dot/RustMinerSystem

[46] Zaxblog, "Original MinerProxy," 2023. [Online]. Available: https://github.com/Zaxblog/MinerProxy

[47] Char1esOrz, "MinerProxy," 2022. [Online]. Available: https://github.com/Char1esOrz/minerProxy

[48] MinerProxyBTC, "GoMinerTool," 2022. [Online]. Available: https://github.com/MinerProxyBTC/GoMinerTool

[49] nicococococ, "MinerProxy303," 2022. [Online]. Available: https://github.com/nicococococ/MinerProxy303

[50] deepwn, "DeepMiner" 2019. [Online]. Available: https://github.com/deepwn/deepMiner