Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier INSERT

An Intelligent Distributed Ledger Construction Algorithm for IoT

CHARLES RAWLINS[1], (Member, IEEE) and S. JAGANNATHAN[1](FELLOW, IEEE)
[1]Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65409 USA

Corresponding author: Charles C. Rawlins (e-mail: crawlins@mst.edu).

ABSTRACT Blockchain is the next generation of secure data management that creates near-immutable decentralized storage. Secure cryptography created a niche for blockchain to provide alternatives to well-known security compromises. However, design bottlenecks with traditional blockchain data structures scale poorly with increased network usage and are extremely computation-intensive. This made the technology difficult to combine with limited devices, like those in Internet of Things networks. In protocols like IOTA, replacement of blockchain's linked-list queue processing with a lightweight dynamic ledger showed remarkable throughput performance. However, current stochastic algorithms for ledger construction suffer distinct trade-offs between efficiency and security. This work proposed a machine-learning approach with a multi-arm bandit that resolved these issues and was designed for auditing on limited devices. This algorithm was tested in a reinforcement-learning environment simulating the IOTA ledger's construction with a decision tree. This study showed through regret analysis and experimentation that this approach was secure against manipulation attacks while remaining energy-efficient. Although the IOTA protocol was a pioneer for lightweight distributed ledgers, it is expected that future blockchain protocols will adopt techniques similar to those presented in this work.

INDEX TERMS Blockchain Security, Distributed Ledger Technology, Internet of Things, Machine Learning, Multi-arm Bandit, Regret Analysis

I. INTRODUCTION

Blockchain is one of the most revolutionary methods for securing network data with adoption that has exploded into a variety of data management communities like government documents [1] and financial products [2]. Recently, Internet of Things (IoT) devices have received attention for data security in industries like supply-chain and consumer households. Data gathered from these devices efficiently tracks inventory and products at distant at third-party servers [3]. Limited hardware and mobility of many devices usually implies reduced security [4]. Distributed data can be used in social engineering or behavior prediction [5], so there is a need for improved trust in the technology.

Blockchain is a potential solution with decentralized data management. However, there are many challenges to implement blockchain on IoT due to traditional blockchain's design decisions. The computationally-intense queue processing of Sybil-prevention mechanisms like Proof-of-Work (PoW) in Bitcoin [6] create bottlenecks for transaction throughput and network growth that directly conflicts with the properties of IoT. Recent works offloaded computation with machine-learning [7, 8] but compromise decentralization. Only efforts that alternate ledger storage formats [9–11] and allow concurrency with lightweight verification can be feasibly implemented on IoT devices. Of these, the IOTA protocol [9] successfully found a balance between security and performance for IoT.

An open problem in the IOTA-related literature is the development of lightweight and secure transaction selection algorithms for ledger construction [12]. The secure selection schemes are required to approve two existing transactions in creation of a new transaction, which decides what data persists as ground-truth and can be vulnerable to ledger-specific attacks. There were published papers that discussed improvements to ledger construction by expanding on basic methods [9, 12]. For instance, G-IOTA's [13] approach proposed possibly approving a third transaction. E-IOTA [14] built on G-IOTA by randomly varying several construction algorithms. The approach Ferraro, King, and Shorten [15] explored used liquid modelling with IOTA's ledger and proposed their own hybrid scheme for high transaction confirmation rates. Chafjiri and Mehdi Esnaashari Esfahani

[16] proposed an algorithm with adaptive parameters for a random walk through the ledger. The IOTA foundation also developed an attack detection scheme using ledger features [17], though it was not used in ledger construction. These approaches were randomized to prevent adversarial gaming, but do not remove a distinct possibility that a transaction will go unconfirmed in the ledger. Use of complex features in transaction selection to prevent this issue, such as using a ledger or network state, were absent in the literature.

This paper proposes a ledger construction algorithm with a multi-arm bandit algorithm to resolve the issue of unconfirmed transactions while remaining secure. The approach also considered limited IoT devices in its design with an auditable decision tree using Q-learning. Based on the author's understanding, this is the first blockchain protocol to actively use machine-learning in decision-making, which is an open problem in general blockchain research [18, 19].

The contributions for this work include secure Reinforcement Learning (RL) analysis for lightweight ledger construction and a robust multi-arm bandit algorithm. Regret analysis and simulation for performance and security guarantees was also included. Section II provided general background related to blockchain and learning mechanisms. Section III discussed the methodology for analyzing the IOTA ledger and its security model. Section IV shows the algorithm's effectiveness and regret analysis.

II. BACKGROUND

Combining IoT with blockchain is crucial for data security, but infeasible with traditional designs. Concurrency with newer protocols' execution presented opportunities for lightweight machine learning in optimization and attack avoidance.

A. IOT SECURITY AND IOTA

Internet of Things (IoT) consists of numerous heterogeneous devices communicating to improve either consumer quality-of-life or manufacturing efficiency. Trivial schemes of collecting IoT data include traditional database systems with network-distant servers [20]. In this scheme, a user must accept consequences of not controlling their own data [21]. A solution requires a secure means of recording data management activity to improve privacy. An advancement toward this was with blockchain networks, which are computation-intense security protocols for storing arbitrary network data in a transaction format via a distributed ledger. There are numerous factors that prevent collaboration of IoT and blockchain, including: network scalability, computational demand, and large ledger storage.

One methodology the blockchain industry used to address these issues was alternative ledger structures. Recent protocols proposed generalized schemes that form a Directed Acyclic Graph (DAG). These ledgers are fundamentally different but accomplish similar tasks to traditional blockchain, so they and blockchain have been termed Distributed Ledger Technology (DLT). A DAG ledger consists of interconnected single transactions that each reference one or more previous transactions. This form factor was adopted by various DLT networks such as Hadera Hashgraph [10], Nano [11], and Obyte [22]. One pioneering DLT network offering protocol flexibility is IOTA. This protocol is an IoT-focused scheme that emphasised direct use on IoT devices with its ledger, called the Tangle. The Tangle's dynamic benefits in ledger construction and data storage make it practical for use in IoT networks. It is important to understand how the IOTA protocol is groundbreaking for allowing direct IoT device usage in DLT networks, as it presents many research opportunities for securing distributed IoT data.

IOTA at the time of writing experienced a massive research effort that removes a centralized coordinator for verifying transactions [12]. An open area of research noted by the IOTA foundation is effective transaction selection algorithms [12]. As transactions are selected by individual nodes and appended to the tip of the DAG, there were no enforced protocols for devices to confirm a certain transaction. The class of algorithms doing this are simply called tip selection algorithms, which are used for ledger construction.

B. REINFORCEMENT LEARNING

The general learning methodology chosen to create an intelligent tip selection algorithm was Reinforcement Learning (RL) with Q-learning [23]. In this scheme, a learner chose an action in the action space $A \in \mathcal{A}$ to take for a given state in the state space $S \in \mathcal{S}$ of a Markov Decision Process (MDP). Each $A \in S$ was associated with a reward value R from a value function V, called Q-values. The optimal policy from these Q-values, Q_* is usually approximated with a machine learner like a Multi-layer Perceptron (MLP).

A design decision considered for directly applying machine learning with blockchain was maintaining auditability. A variety of machine learning algorithms are powerful in approximating unknown nonlinear functions, but are difficult for average humans and researchers to understand. The key strength of blockchain and DLT is that anyone can verify the account balances in the ledger, making strong security. A Decision Tree (DT) from Conservative Q-improvement [24] was chosen as the base learner for this scheme. This DT creation process decided how to grow the tree based on a perceived increase in Q-values.

The Multi-arm Bandit (MAB) is a classic RL problem for determining an optimal selection from a set of discrete actions in the case of uncertain rewards. The contextual bandit was an extension of the MAB that considers an environment with different states of learning for $S \in \mathcal{S}$ [25]. The main decision-making algorithms behind a MAB are called bandit algorithms. Bandits have been analyzed through regret analysis for their performances and security benefits. It was noted that several lightweight bandits were gameable by adversaries, which was accomplished by poisoning rewards [26]. One work showed that bandits like ϵ-greedy and Upper Confidence Bound (UCB), were gameable in $\mathcal{O}(log(n))$ time [27]. One of the simplest robust bandit algorithms was Thompson Sampling (TS) [28]. TS is the first bandit algorithm that successfully balanced exploration and exploitation with use of the beta distribution for posterior distribution sampling. With a beta posterior distribution, TS finds an optimal arm for K actions with shape parameters $A_o = B_o = 1$: $a = \text{argmax}_{k \in K}\{beta(A_k, B_k)\}$. As the learner encounters changes in the environment through Bernoulli rewards $R \in [0, 1]$, the update for each shape parameter associated with the chosen action is

$$\alpha_t, \beta_t = \begin{cases} A_k, B_k & \text{if } a \neq k \\ A_k + R,\ B_k + (1 - R) & \text{if } a = k \end{cases}$$

The security of TS showed robustness to adversarial attacks under arm manipulation [29], but [26] showed an effective scheme to poison rewards with offline data. It is important to understand the the vulnerabilities of TS for creation of a robust version in Section III.

III. METHODOLOGY

After a brief discussion of ledger construction and security, we discuss the threat model and transaction selection before providing the algorithms. For construction of the IOTA ledger, analysis of ideal behavior was necessary for reward function design. Security concerns and known vulnerabilities were then considered for robust decision-making. This helped create a RL value function for ledger construction. Finally, a multi-arm bandit algorithm was proposed for protection against reward attacks. This algorithm uses batch processing with highlighted rewards and filtering to prevent adversary impulse attacks from influencing learning.

A. LEDGER CONSTRUCTION AND SECURITY

The IOTA Tangle ledger is a DAG $G = (V, E)$. Each $v \in V$ contains a transaction that represented an exchange of tokens between multiple parties. Directed edge $e \in E$ represent a cryptographic link approving a previous transaction in the ledger. The only requirement in the IOTA protocol for a node approving a new transaction is confirming two previous transactions [9]. In addition, as networking delays and disconnects occur, some nodes may create subgraphs of G with differing v and e. For simplicity of analysis, these differences were ignored to create only a single version of the ledger. Any time a transaction is generated, the following steps are performed:

1) A node broadcasted the new transaction x
2) Neighbor received x
3) Neighbor conducted a selection algorithm k times, where $k \geq 2$ to confirm k previous transactions
4) Neighbor conducted a verification scheme to generate k cryptographic links $e \in E$
5) Broadcast e with x to the network
6) Network verified the integrity of the cryptographic link

It was assumed the neighbor verified the data with PoW, though any verification scheme could be used. Between steps 2 and 3, the node has an opportunity to analyze the ledger state.

The ledger state viewed from a single transaction x can be viewed as a discrete Markov chain [30]. Between each state, the total number of unconfirmed transactions $L(t)$ can vary along with the accumulated weight of an individual transaction $W(x)$. $L(t)$ can be modelled as a Poisson process [9] with mean rate λ. $W(x)$ is the accumulation of individual weights of all future transactions approving x, where each weight is proportional to the amount of work done to verify those transactions. It was assumed $W_0 = 1$ and all future weights are also 1 for this study.

The value $|L(t)|$ depends highly on the tip selection algorithm chosen to construct the ledger [9]. The simplest tip selection algorithm is called Uniform Random Tip Selection (URTS), which randomly selected transactions with probability $\frac{1}{|L(t)|}$. Each transaction has a probability > 0 of being selected, so URTS is guaranteed to not leave transactions behind [9]. The stable value for URTS is [9]

$$L(t) = \frac{k\lambda h}{k - 1}, \tag{1}$$

where h is the time taken to confirm a transaction (it is assumed $h = 1$). Generally nodes choose $k = 2$, $L(t) \approx 2\lambda = L_{Ideal}$. If $L(t) < L_{Ideal}$ (due to $k > 2$), then nodes will waste resources confirming past transactions. However, one issue with URTS is that it can be easily gamed into selecting transactions approving an attack on the ledger. If an attack created some conflict in the ledger, an adversary can create numerous empty transactions approving their attack. When a node confirmed one of these dummy transactions, it helps the attacker (see Fig. 1).

With these known vulnerabilities, Markov Chain Monte Carlo (MCMC) was instead developed [9]. With MCMC, a weighted random walk was used to select a new transaction at the ledger edge. MCMC started from transaction in a window $[W, 2W]$, and a random walk was taken from a transaction towards the unconfirmed tip edge. The parameter α biases selection towards transactions with higher weight accumulated from future transaction selection (see Fig. 1). $\alpha > 0$ prevents selection of transactions that would help this type of attack be successful, called a parasite chain attack. The probability of a random walker transitioning between transactions x and y with z transactions in between is found by the following:

$$P_{xy} = \frac{exp(\alpha W_y)}{\Sigma_{z:z \rightsquigarrow x} exp(\alpha W_z)}. \tag{2}$$

High bias will not likely explore the Tangle for legal transactions that have less weight compared to the frequently-selected, meaning they will be selected with probability ≈ 0. As the majority of the network will produce higher weights with legal transactions, it will outpace the attacker. This also applies to other attacks that target cumulative weight, like ledger splitting [9].

Information was presented to a machine learner to find $S \in \mathcal{S}$. The goal of the learner is to avoid attacks while

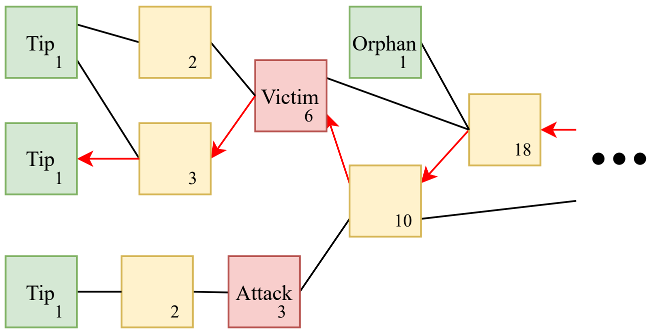

FIGURE 1. Parasite Chain Attack Example

simultaneously confirming late transactions. The distribution of transaction arrival times made from the network Poisson process was approximated with the distribution normal mean μ_{tip} and standard deviation σ_{tip}. The following was used to represent S as a vector,

- Perceived network transaction rate λ
- The difference $L_{Err} = |L_{Ideal} - L(t)|$
- Average tip time μ_{tip}
- Tip time standard deviation σ_{tip}

The actions chosen for the learner consisted of standard tip selection algorithms. These actions were:

- URTS with $k = 2$
- URTS with $k = 3$
- MCMC with $\alpha = 0.01,\ k = 2$
- MCMC with $\alpha = 0.9,\ k = 2$

A learner conducting either URTS or MCMC creates either exploratory or secure actions. Altering α allows a learner to alter the distribution of tips at the side or front edge of the DAG [31].

With these considerations, a reward function for ledger construction was create. Behavior letting transactions fall behind was inhibited while incentivizing secure actions. Actions with $k > 2$ were also discouraged from selection to only when necessary. The reward function with $R \in [0, 1]$ is

$$R(\lambda, \mu_{tip}, \sigma_{tip}, M) = 0.9 - 0.05(k-2) - L_B^{k/\sigma_{tip}} + B_{MCMC} + B_{URTS}, \quad (3)$$

where L_B is the number of transactions left behind,

$$L_B = \frac{|\{x \in L(t) \mid x_{\text{rec. time}} < (\mu_{tip} - 3 * \sigma_{tip})\}| - |M|}{|L(t)|},$$

while B_{MCMC} and B_{URTS} are

$$B_{MCMC} = 0.1\alpha_{MCMC} * \mathbb{1}\{|L_B| = 0\}$$

$$B_{URTS} = 0.01 L_B * \mathbb{1}\{|L_B| > 0\}.$$

The set M is the set of malicious transactions associated with any previous attack. According to most terms in (3), an adversary would not influence the reward. However, if unconfirmed transactions were spammed, L_B increases and creates a drop in reward. This was addressed with an algorithm that is robust to dramatic reward changes.

B. THREAT MODEL

Although one of the main benefits of using a DAG is the increased throughput, the downside is the additional vulnerabilities. The overall goal of an attacker is to create an attack transaction and convince the network it is correct. This is well-known in blockchain literature as a double-spend attack [32]. This attack can be accomplished through various methods with their own attack name, such as the 51% attack [32]. The exact percentage of network computing power a successful attack needs varies depending on blockchain consensus mechanisms and strategies [33]. This study considered either the brute-force strategy, where the attacker can only leverage blank transactions for the attack, or other ledger-specific attacks and was in the network minority. It was assumed the attacker does not control information sent to an IoT node in an Eclipse or Sybil-related attack [34]. The attacker also had perfect knowledge of the ledger learner and all internal parameters, because it was white-box.

The additional vulnerabilities from adding RL were also addressed . An attacker will try to game the decision-making process of an innocent node to help in their attack strategy through several means, such as [35]: evasion, poisoning, and exploratory. It was assumed that an adversary could alter the inputs to influence the reward during testing, meaning robustness against attacks was needed. Exploratory attacks were not beneficial to the attacker, because the learner was already white-box.

C. INTELLIGENT TRANSACTION SELECTION

The general approach for intelligent machine-learning ledger construction is shown in Algorithm 1. The study provided an extension to Bernoulli TS with an approach that emphasizes learning from consistent reward increases in Algorithm 2, called Biased TS. The main novelty with this approach comes from the adjusted shape values for each arm created by frequency of arm pulls. If an adversary poisons several reward values for an non-optimal arm, the few impulses will eventually be forgotten by another arm that creates a more consistent average from the innocent network majority for the beta update in $[0, 1]$. The inspiration for this approach came from batch processing with TS [36] and posterior sampling [37]. Given the literature discussing the adversarial security of TS [26, 29], it was assumed that Biased TS was still vulnerable to manipulation with large reward values. A simple prevention scheme against reward poisoning attacks is presented in Algorithm 3.

To further protect against attacks, the core bandit algorithm's vision was reduced. This was done with a light form of filtering shown in Algorithm 3. In this manner, if an adversary were to create a parasite chain that drastically changed $|L(t)|$, it would not influence the behavior of the learner. This approach does not prevent selection of parasite chain transactions, but it does prevent adversarial gaming.

While [38] originally presented the idea of a trimmed mean to a UCB, the main novelty with this approach was directly removing the adversary influence by filtering out large adversarial attacks that would otherwise influence an arm's mean reward. Notice the careful indexing in Algorithm 3, where actions selected for the current batch are also compared to the previous experience if $f < C$. Legal actors can avoid being added to M by slowly publishing their transactions along with the network.

Algorithm 1 General Ledger Construction Algorithm

```
1: Initialize $L(t), M, A_0 = B_0 = 1, C, t_S = 0 \forall S$
2: **while** receive $x \in L(t)$ **do**
3:     $S_{obv} = \{\lambda, L_{Err}, \mu, \sigma\}$
4:     Get $S \in \mathcal{S}$ from decision tree
5:     $a$ = BiasedTS(S)
6:     Update tree parameters according to [24]
7:     Verify $x$ legibility with $k$ tips from $a$
8:     Broadcast $x$ with $k$ edges
```

Algorithm 2 BiasedTS(S)

```
1: Get $A_k, B_k, R_k, f_k$ from S
2: $a = \text{argmax}_{k \in K}\{beta(A_k, B_k)\}$
3: $f_a += 1$
4: $R_a.append($ reward from (3))
5: **if** $t_S \geq C$ **then**
6:     $R_a = \text{FilteredMean}(R_a)$
7:     $I = f_a * \mathbb{E}(R_a)$
8:     $F = f_a * (1 - \mathbb{E}(R_a))$
9:     $U = beta(I, F)$
10:    $A_a = A_a + U$
11:    $B_a = B_a + (1 - U)$
12:    $R_a.pop(1)$
13: t = 0
14: $f_A = 0$
15: return $A$
```

Algorithm 3 FilteredMean(R)

```
1: $Y = \{|R[1] - R[2]|, ..., |R[C] - R[C-1]|\}$
2: $A = [\,]$
3: $R_{\mu,\Delta R} = \mathbb{E}(Y)$
4: $R_{\sigma,\Delta R} = \sigma(Y)$
5: **for** i in 1 : $|Y|$ **do**
6:     **if** $Y[i] > R_{\mu,\Delta R} + 3 * R_{\sigma,\Delta R}$ **then**
7:         $A.append(i)$
8: **for** $i$ in $A$ **do**
9:     Add transactions causing attack $i$ to $M$
10:    **for** $j$ in $i : C - 1$ **do**
11:        $Y[j] = Y[j] - Y[i]$
12: $R = \{R[1], R[1] + Y[1], ..., R[C-1] + Y[C-1]\}$
13: Return $R$
```

IV. RESULTS

To demonstrate the methodology in Algorithm 1 and from other approaches in the literature, the IOTA ledger simulator[1] was used. A RL environment for the Tangle was created in OpenAI gym in Python [39]. The code showing this simulation is provided in the Github repository[2].

A. COMPARISON OF ALGORITHMS

The tip selection algorithm with Biased TS was compared to two other approaches. The current algorithm used by the IOTA foundation is called almostURTS [12]. The almostURTS algorithm is an extension of URTS with an added weight bias towards transactions that were confirmed with recent timestamps. To simulate network delays, noise was added to a transactions received timestamp. Without this noise, almostURTS behaved like regular URTS. The other algorithm selected was 'E-IOTA' [14]. E-IOTA varies algorithm selection randomly with MCMC and varying levels of α. The parameters for this algorithm were selected based on the best parameters presented in [14].

The training and DT parameters are shown in Table 1. Training varied λ uniformly with no attacks. After each trail, tree nodes not frequently visited were pruned. The bandit had difficulty balancing actions without prior reward experience, so learning was frozen until tree nodes were generated. New nodes in the tree inherited A_k, B_k upon splitting, and H_S was reset to 1 each time a node was split.

TABLE 1. DT Training and Testing Parameters

Q-learning/Training	
Learning Rate α	0.9
Reward Bias γ	0.95
Trials	30
Biased TS Batch C	70
λ Range	[5, 50]
Testing Parameters	
λ	10
Trials per. Algo.	10
MCMC α	0.9
almostURTS (C_1, C_{1p}, C_2, M)	$(1, 1, 1, 10)$
E-IOTA (p_1, p_2)	$(0.1, 0.35)$
Init. Split Threshold H_S	1
Visit Decay d	0.999
Split Threshold Decay D	0.9999
Visit Trim Perc.	10%
Max. Tree Depth	20
Regret Simulation Parameters	
Time Steps T	10^4
Trials	10
ϵ-greedy ϵ Start Value	4
Biased TS Batch C	10
Arm Budget B_k	100

The performance of each algorithm was compared by looking at the amount of transactions left behind. These were measured by counting timestamps more than three standard deviations of tip time for the pure URTS selection from the

[1]IOTA Ledger Simulator Github Repository
[2]Biased TS Github Repository to be added in the final version of the paper

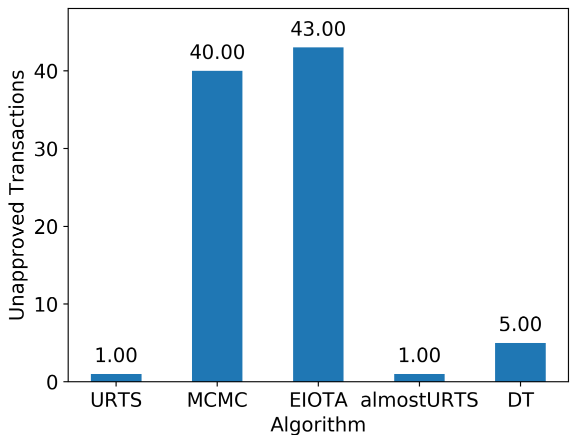

FIGURE 2. Average Transactions Left Behind Comparison

TABLE 2. Average Algorithm Computation Comparison (10 Trials)

	URTS	MCMC	E-IOTA	almostURTS	DT
URTS	1998	0	0	1998	10
MCMC (α=0)	0	0	202	0	0
MCMC	0	1998	1796	0	1988

respective algorithm mean. The best-case (URTS) and worst-case (MCMC) scenarios for leaving transactions behind were also performed (see Fig. 2).

The DT approach was more effective at taking actions to confirm late transactions compared to stochastic algorithms. The performance was comparable to almostURTS but also accomplished secure actions. In addition, the number of computations for each basic algorithm were compared in Table 2. The low amount of URTS action selections was achievable for the DT with a high learning rate and reward bias. The slow learning of Biased TS helped child nodes in the DT improve from experience, with the drawback of additional training.

B. REGRET ANALYSIS AND SIMULATION

For regret analysis, an extension was provided to existing work for regret with Bernoulli TS to estimate the upper bound for regret in Biased TS [40].

Theorem 1. *Let each arm $i = 1, ..., k$ and constant batch number C. The expected amount of regret for Biased TS in Algorithm 2 is*

$$\mathbb{E}[R(T)] \leq \sum_{i}^{k} \frac{4\, lnT}{C} \Delta i + \frac{48}{\mu_i \Delta i^2}.$$

An approach similar to [29] was used to find expected regret of Biased TS under an adversarial setting. This assumed that the adversary had a budget strategy of manipulating an arm with total budget B_i.

Theorem 2. *Let each arm $i = 1, ..., k$ and constant batch number C. The expected amount of regret for Biased TS under an adversarial setting with arm budget B_i is*

$$\mathbb{E}[R(T)] \leq \sum_{i}^{k} \frac{4\, lnT}{C} \Delta i + \max\{\frac{H}{2}, H\sigma logT\},$$

where $H = |3\sigma_{\Delta R} + \mu_{\Delta R}|$ for the reward change distribution.

The full proof of Theorem 1 and 2 with full term definitions is provided in the Appendix.

Other bandit algorithms, ϵ-greedy and Gaussian TS, were compared against for their robustness to arm manipulation. The adversary had a total budget for manipulating each individual arm B_k and individual trial manipulation values $\alpha_{t,k}$ to increase a reward. Each algorithm was tested with and without the Lump Sum Investment (LSI) attack strategy outlined in [29], where the attacker spends the rest of its remaining budget on attacking an arm to convince the bandit it had higher rewards at times τ: $R_k = R_k + B_k - \Sigma_{t=1}^{\tau} \alpha_{t,k}$. Fig. 3 and Fig. 4 compare each algorithm's standard and adversarial regret respectively. The regret for each trial $|R_* - R_k|$ was plotted and averaged over 10 trials. The simulation parameters for regret are shown in Table 1.

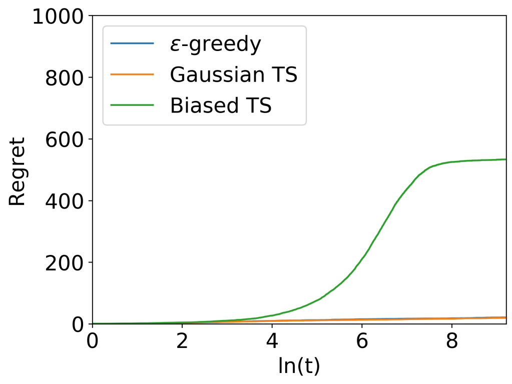

FIGURE 3. Control Regret Comparison ($B_k = 0$)

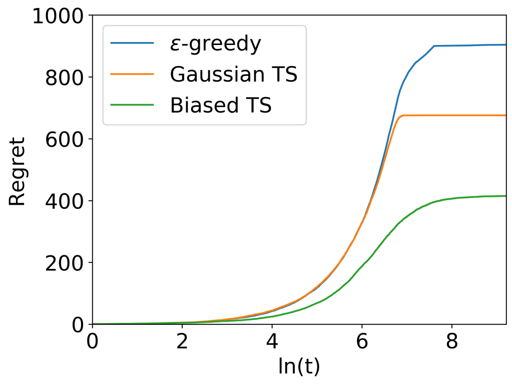

FIGURE 4. Adversarial Regret Comparison ($B_k = 100$)

Biased TS had noticeably worse regret under standard conditions compared to Gaussian TS and ϵ-greedy. However, Biased TS maintained similar regret levels under heavy attack and showed superior performance compared to the other two algorithms. Batch processing with Biased TS did slow performance to find the optimal action, but also prevented attacks.

C. PERFORMANCE MEASUREMENTS

For consideration of IoT devices, tip selection with Biased TS was measured for its energy consumption on physical hardware. A Raspberry Pi 4 Model B with the standard Raspberry Pi OS was running Python 3.8 with code provided in the repository using a standard 5V power supply. The Pi uses a 1.5 GHz CPU with the Arm Cortex-A72 architecture. Each algorithm in Table 3 was executed for 10 trials of creating 1000 transactions with $\lambda = 10$. Each group of trials was repeated 5 times for a total of 50 trials. The Pi was monitored with a USB Power monitor made by MakerHawk and recorded on a separate computer. The voltage and current were measured and averaged over each trial group and recorded in Table 3. In addition, the Linux tool `perf stat` was used to estimate the number of cycles and instructions generated by each algorithm. The results showed that the DT did not cause a drastic power draw or computation compared to tested algorithms.

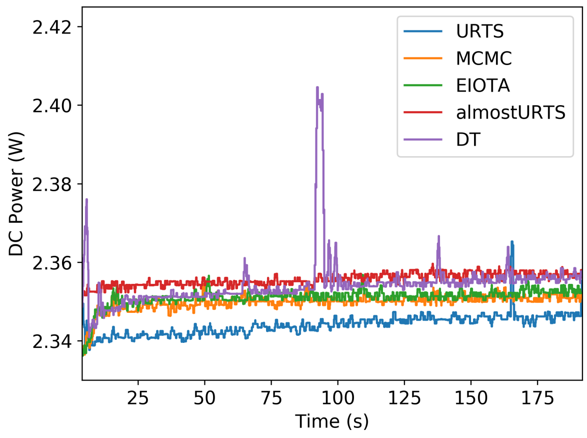

FIGURE 5. Tip Selection Energy Comparison

TABLE 3. Average Computation Experiment Comparison (50 trials)

Algorithm	Power (W)	Instr. [10^9]	Cycles [10^9]
URTS	2.367	209	117
MCMC	2.369	351	217
E-IOTA	2.368	318	196
almostURTS	2.370	234	133
DT	**2.372**	**408**	**260**

D. REFLECTION AND FUTURE WORK

Ledger construction with Biased TS showed improved with intelligent selection. Highlighting consistent reward changes and filtering outliers successfully prevents impulse attacks. Bounding filtered values by outliers in the reward change distribution were key for logarithmic regret bounds. One issue with the learning strength of the DT is that it needed incentive to minimize selection of URTS by adding B_{URTS} and B_{MCMC} to (3), increasing complexity. Another issue worth noting is that an adversary may have more success with manipulation if they are able to create consistent poisoned rewards, though only if they are not network minority.

The next opportunity to be explored is preemptively adding additional transactions to M before they are confirmed. While it is true that attacks in a distributed ledger are repelled by consensus in the DLT network, avoidance is crucial to reduce computation and bandwidth. This direction will be explored from a data clustering approach in future work. Other areas to explore could include altering the reward function to prioritize actions for computational efficiency.

V. CONCLUSIONS

This effort presented a machine learning-based DLT ledger construction algorithm for IoT with Biased TS. The algorithm's performance was compared to other schemes through simulation and improved transaction confirmation performance was found. This approach was also comparable in energy consumption to other tested algorithms. Proof of security against manipulation for Biased TS was shown with better regret under attacks compared to other bandits. Future work will include preemptively flagging attack transactions through data clustering.

ACKNOWLEDGEMENTS

This work is funded by the by the Graduate Assistance in Areas of National Need (GAANN) national fellowship program and Department of Energy's Kansas City National Security Campus, operated by Honeywell Federal Manufacturing & Technologies, LLC, under contract number DE-NA0002839.

References

[1] A. Mohite and A. Acharya, "Blockchain for government fund tracking using hyperledger," in *2018 Int. Conf. on Comp. Techniques, Elec. and Mech. Systems (CTEMS)*, 2018, pp. 231–234. DOI: 10.1109/CTEMS.2018.8769200.

[2] B. Chen, Z. Tan, and W. Fang, "Blockchain-based implementation for financial product management," in *2018 28th Int. Tele. Net. and App. Conf. (ITNAC)*, 2018, pp. 1–3. DOI: 10.1109/ATNAC.2018.8615246.

[3] J. H. Jeon, K.-H. Kim, and J.-H. Kim, "Block chain based data security enhanced iot server platform," in *2018 Int. Conf. on Info. Networking (ICOIN)*, 2018, pp. 941–944. DOI: 10.1109/ICOIN.2018.8343262.

[4] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5,

pp. 8182–8201, 2019. DOI: 10.1109/JIOT.2019.2935189.

[5] M. Ghasemi, M. Saadaat, and O. Ghollasi, "Threats of social engineering attacks against security of internet of things (iot)," in *Fundamental Research in Electrical Engineering*, S. Montaser Kouhsari, Ed., Singapore: Springer Singapore, 2019, pp. 957–968.

[6] S. Nakamoto. (2008). "Bitcoin: A peer-to-peer electronic cash system," [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[7] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050–8062, 2019. DOI: 10.1109/TVT.2019.2924015.

[8] N. Mhaisen, N. Fetais, A. Erbad, A. Mohamed, and M. Guizani, "To chain or not to chain: A reinforcement learning approach for blockchain-enabled IoT monitoring applications," *Future Generation Computer Systems*, vol. 111, pp. 39–51, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.04.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19334399.

[9] S. Popov. (2018). "The tangle."

[10] D. L. Baird, M. Harmon, and P. Madsen. (2020). "Hedera: A public hashgraph network & governing council," [Online]. Available: https://hedera.com/hh-whitepaper.

[11] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," 2017. [Online]. Available: https://content.nano.org/whitepaper/Nano_Whitepaper_en.pdf.

[12] IOTA Foundation. (Jan. 2020). "The coordicide," [Online]. Available: https://files.iota.org/papers/20200120_Coordicide_WP.pdf.

[13] G. Bu, Ö. Gürcan, and M. Potop-Butucaru, "G-IOTA: Fair and confidence aware tangle," *CoRR*, vol. abs/1902.09472, 2019. arXiv: 1902.09472. [Online]. Available: http://arxiv.org/abs/1902.09472.

[14] G. Bu, W. Hana, and M. Potop-Butucaru, "Metamorphic IOTA," *CoRR*, vol. abs/1907.03628, 2019. arXiv: 1907.03628. [Online]. Available: http://arxiv.org/abs/1907.03628.

[15] P. Ferraro, C. King, and R. Shorten, "On the stability of unverified transactions in a DAG-based distributed ledger," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3772–3783, 2020.

[16] F. S. Chafjiri and M. Mehdi Esnaashari Esfahani, "An adaptive random walk algorithm for selecting tips in the tangle," in *2019 5th International Conference on Web Research (ICWR)*, 2019, pp. 161–166. DOI: 10.1109/ICWR.2019.8765264.

[17] Andreas Penzkofer, Bartosz Kusmierz, Angelo Capossele, William Sanders, Olivia Saa, "Parasite Chain Detection in the IOTA Protocol," Apr. 2020.

[18] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for ai: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10 127–10 149, 2019. DOI: 10.1109/ACCESS.2018.2890507.

[19] W. Zhao, C. Jiang, H. Gao, S. Yang, and X. Luo, "Blockchain-enabled cyber–physical systems: A review," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4023–4034, 2021. DOI: 10.1109/JIOT.2020.3014864.

[20] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Systematically evaluating security and privacy for consumer iot devices," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1–6, ISBN: 9781450353960. DOI: 10.1145/3139937.3139938. [Online]. Available: https://doi.org/10.1145/3139937.3139938.

[21] H. A. Abdulghani, N. A. Nijdam, A. Collen, and D. Konstantas, "A study on security and privacy guidelines, countermeasures, threats: Iot data at rest perspective," *Symmetry*, vol. 11, no. 6, 2019, ISSN: 2073-8994. DOI: 10.3390/sym11060774. [Online]. Available: https://www.mdpi.com/2073-8994/11/6/774.

[22] A. Churyumov. (2016). "Byteball: A decentralized system for storage and transfer of value," [Online]. Available: https://obyte.org/Byteball.pdf.

[23] C. Watkins, "Learning from delayed rewards," Jan. 1989.

[24] A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso, "Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy," 2019. arXiv: 1907.01180 `[cs.LG]`.

[25] R. S. Sutton and A. G. Barto, "Multi-armed bandits," in *Reinforcement Learning: An Introduction*, Second, The MIT Press, 2018, ch. 2, pp. 25–27.

[26] F. Liu and N. Shroff, "Data poisoning attacks on stochastic bandits," 2019. arXiv: 1905.06494 `[cs.LG]`.

[27] K.-S. Jun, L. Li, Y. Ma, and X. Zhu, "Adversarial attacks on stochastic bandits," *CoRR*, vol. abs/1810.12188, 2018. arXiv: 1810.12188. [Online]. Available: http://arxiv.org/abs/1810.12188.

[28] W. R. Thompson, "On the likelihood one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, Dec. 1933, ISSN: 0006-3444. DOI: 10.1093/biomet/25.3-4.285. eprint: https://academic.oup.com/biomet/article-pdf/25/3-4/285/513725/25-3-4-285.pdf. [Online]. Available: https://doi.org/10.1093/biomet/25.3-4.285.

[29] Z. Feng, D. C. Parkes, and H. Xu, "The intrinsic robustness of stochastic bandits to strategic manipulation," 2020. arXiv: 1906.01528 `[cs.LG]`.

[30] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu. (2020). "Direct acyclic graph-based ledger for internet of things: Performance and security analysis," [Online]. Available: https://files.iota.org/papers/Direct_Acyclic_Graph-based_Ledger_for_Internet_of_Things.pdf.

[31] B. Kusmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the Tangle for uniform random and random walk tip selection," in *2019 IEEE Int. Conf. on Blockchain (Blockchain)*, IEEE, 2019, pp. 228–236.

[32] M. Saad, J. Spaulding, L. Njilla, C. A. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen, "Exploring the attack surface of blockchain: A systematic overview," *CoRR*, vol. abs/1904.03487, 2019. arXiv: 1904.03487. [Online]. Available: http://arxiv.org/abs/1904.03487.

[33] I. Eyal and E. G. Sirer, *Majority is not enough: Bitcoin mining is vulnerable*, 2013. arXiv: 1311.0243 [cs.CR].

[34] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 299–314, Dec. 2003, ISSN: 0163-5980. DOI: 10.1145/844128.844156. [Online]. Available: https://doi.org/10.1145/844128.844156.

[35] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, *Adversarial attacks and defences: A survey*, 2018. arXiv: 1810.00069 [cs.LG].

[36] S. R. Chowdhury and A. Gopalan, *On batch bayesian optimization*, 2019. arXiv: 1911.01032 [cs.LG].

[37] I. Osband, D. Russo, and B. V. Roy, *(More) efficient reinforcement learning via posterior sampling*, 2013. arXiv: 1306.0940 [stat.ML].

[38] L. Niss and A. Tewari, "What you see may not be what you get: UCB bandit algorithms robust to ε-contamination," in *Proceedings of the 36th Conf. on Uncertainty in Artificial Intelligence (UAI)*, J. Peters and D. Sontag, Eds., ser. Proceedings of Machine Learning Research, vol. 124, PMLR, Aug. 2020, pp. 450–459. [Online]. Available: http://proceedings.mlr.press/v124/niss20a.html.

[39] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[40] S. Agrawal and N. Goyal, "Near-optimal regret bounds for thompson sampling," *J. ACM*, vol. 64, no. 5, Sep. 2017, ISSN: 0004-5411. DOI: 10.1145/3088510. [Online]. Available: https://doi.org/10.1145/3088510.

[41] M. Mitzenmacher, E. Upfal, and C. U. Press, *Probability and computing: randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005, ISBN: 9780521835404. [Online]. Available: https://books.google.com/books?id=0bAYl6d7hvkC.

.

APPENDIX A PROOFS

A. DEFINITIONS AND ASSUMPTIONS

The definitions below are for understanding the proceeding analysis. Several definitions are also found in [40], while others are unique to this work.

Definition 1. *$(n_i(t), \hat{\mu}_i, \mu_1, \mu_{i,C}, \Delta_i, \theta_i(t))$ Let $n_i(t)$ be the number of times arm i was pulled. $\hat{\mu}_i$ is the empirical mean for arm i, which is defined as*

$$\hat{\mu}_i = \frac{\sum_{\tau=1:i(\tau)=i}^{t-1} r_i(\tau)}{n_i(t)+1}$$

μ_i is the reward mean for arm i and μ_1 is the optimal arm mean. $\mu_{i,C}$ is the mean regret created by the beta distribution update in Algorithm 2, This value is expanded upon in the adversarial proof but used on its own in the general proof. $\Delta_i = |\mu_1 - \mu_i|$ is the remaining difference between the optimal and non-optimal arms for each arm i. $\theta_i(t)$ is a posterior distribution sample.

Definition 2. *(x_i and y_i) x_i and y_i are two thresholds for $\theta_i(t)$ where $\mu_i < x_i < y_i < \mu_1$.*

Definition 3. *($E_i^{\mu}(t)$ and $E_i^{\theta}(t)$) Event $E_i^{\mu}(t)$ occurs when $\hat{\mu}_i \le x_i$ and $E_i^{\theta}(t)$ occurs when $\theta_i(t) \le y_i$.*

Definition 4. *(Distribution Measurements) $F_{n,p}^{B}$ and $f_{n,p}^{B}$ denote the binomial distribution's cdf and pmf respectively. $F_{\alpha,\beta}^{beta}$ is the cdf of the beta distribution. In addition, to relate the binomial and beta distributions: $F_{\alpha,\beta}^{beta}(y) = 1 - F_{\alpha+\beta-1,y}^{B}(\alpha-1)$*

Definition 5. *($\mathcal{H}_t$) $\mathcal{H}_t$ is the arm pull history where $\mathcal{H}_t = \{i(\tau), r_{i(\tau)}(\tau), \tau = 1, ..., t\}$, $i(\tau)$ is the arm played for time τ, $r_{i(\tau)}(\tau)$ is the reward observed from time τ.*

Definition 6. *(Multiplicative Chernoff Bound) The multiplicative Chernoff Bound defines a bound for the tails of a distribution. This general bound applies to any random variable with $\mu = \mathbb{E}(x)$. (Theorem 4.4 in [41])*

$$P(X \ge (1+\delta)\mu) \le exp(\frac{-\mu\delta^2}{3})$$

Definition 7. *(H_i) H_i is the maximum value that a change in reward for arm i can take before it is filtered out by Algorithm 3: $H = |3\sigma_{\Delta R} + \mu_{\Delta R}|$. This value assumes that the reward and absolute change distributions are sub-Gaussian.*

Definition 8. *(B_i and Attack Strategies) Let B_i note the budget an adversary has to influence the reward for non-optimal arms. We assume the adversary chooses the optimal attack strategy with the LSI approach [29]. For a general attack, the adversary influence is $\beta_i = \frac{B_i}{n_i(t-1)}$. For the LSI attack, the adversary influence is a massive influence over the arm by using the accrued values $\alpha_i = \frac{B_i}{t}$. The adversary applies the influence using the LSI strategy by $\beta_{i,t} = B_i - \sum_{\tau=1}^{t-1} \alpha_{i,\tau}$.*

B. BANDIT ALGORITHM GENERAL REGRET

The approach to finding the general regret analysis upper bound is an extension of thorough work in [40]. Where this work deviates is changing the reward to TS with an augmented beta distribution found in Algorithm 2.

Proof of Theorem 1. Following a similar strategy to Theorem 1.1 and Lemmas 2.9-2.12 in [40], regret analysis followed a martingale approach by finding individual term bounds for biased TS. The expected number of arm pulls can be found by

$$\mathbb{E}[n_i(T)] = \sum_{t=1}^{T} P(i(t)=i, E_i^{\mu}(t), E_i^{\theta}(t)) + \sum_{t=1}^{T} P(i(t)=i, E_i^{\mu}(t), \overline{E_i^{\theta}(t)}) + \sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\mu}(t)}). \tag{4}$$

The first term is bounded by $\mathcal{O}(1)$ according to Lemma 2.10 in [40]. The remaining two terms are influenced by the beta distribution reward and their bounds are determined by Lemmas 3 and 4. Plugging in the bounds for these terms,

$$\mathbb{E}[n_i(T)] \le \mathcal{O}(1) + 1 + \frac{4ln(T)}{C} + 1 + \frac{48}{\mu_i \Delta_i^2} = \frac{4ln(T)}{C} + \frac{48}{\mu_i \Delta_i^2},$$

and then finding the standard regret,

$$\mathbb{E}[R(T)] \le \sum_i \Delta_i \mathbb{E}[n_i(T)] \le \sum_i \frac{4ln(T)}{C}\Delta_i + \frac{48}{\mu_i \Delta_i}.$$

□

Lemma 3.

$$\sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)) \le \frac{4\ lnT}{C} + 1$$

Proof of Lemma 3. A function that bounds $n_i(t)$ for each arm $Q_i(T)$ is found by

$$\sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)) = \sum_{t=1}^{T} P(i(t)=i, n_i(t) \le Q_i(T), \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)) + \sum_{t=1}^{T} P(i(t)=i, n_i(t) > Q_i(T), \overline{E_i^{\theta}(t)}, E_i^{\mu}(t))$$

$Q_i(t)$ bounds $n_i(t)$ in the first term. For the second term,

$$\sum_{t=1}^{T} P(i(t)=i, n_i(t) > Q_i(T), \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)) \le \mathbb{E}[\sum_{t=1}^{T} \mathbb{I}(n_i(t) > Q_i(T), \hat{\mu}_i(t) \le x_i) \cdot P(\theta_i(t) > y_i | \mathcal{H}_{t-1})]$$

When $\hat{\mu}_i(t) \le x_i$, then the biased reward update in Algorithm 2 stochastically dominates $\theta_i(t)$: $beta(U, (1-U))$, If $U = beta(f_i \cdot \mu_{i,C}, f_i \cdot (1-\mu_{i,C}))$, the standard beta distribution average reduces the update to $\mu_{i,C}$. $\theta_i(t)$ is then dominated by $beta(x_i B_i \mu_{i,C} + \mu_{i,C}, (1-x_i) B_i \mu_{i,C})$, where B_i is the number of batches where i is selected. Given an instance $H_{t=1}$ of $\mathcal{H}_{t-1}$ where $\hat{\mu}_i(t) \le x_i$ and $n_i(t) > Q_i(T)$,

$$P(\theta_i(t) > y_i | \mathcal{H}_{t-1} = H_{t-1}) \le 1 - F^{beta}_{\mu_{i,C}(x_i B_i + 1), \mu_{i,C}(1-x_i) B_i}(y_i)$$

According to Def. 4,

$$1 - F^{beta}_{\mu_{i,C}(x_i B_i + 1), \mu_{i,C}(1-x_i) B_i}(y_i) = F^{B}_{\mu_{i,C}(1+B_i), y_i}(\mu_{i,C}(x_i B_i + 1))$$

Using Def. 6 and substituting $B_i = \lfloor \frac{n_i(t)}{C} \rfloor$ for $C > 1$, set $Q_i(T) = \frac{4\ lnT}{C}$

$$F^{B}_{\mu_{i,C}(1+B_i), y_i}(x_i \mu_{i,C} B_i + x_i) \le exp(-\frac{\mu_{i,C}(x_i B_i)^2}{3}) \le exp(-\frac{\mu_{i,C}(x_i \lfloor \frac{n_i(t)}{C} \rfloor)^2}{3}) \le exp(-\frac{\mu_{i,C}(x_i Q_i(T))^2}{3}) \le \frac{1}{T} \le 1.$$

Putting in the bounds for the original equation,

$$\sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)) \le \frac{4\ lnT}{C} + 1.$$

□

Lemma 4.

$$\sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\mu}(t)}) \le \frac{48}{\mu_i \Delta_i^2}$$

Proof of Lemma 4. Let τ_n denote the n^{th} trial that arm i is pulled,

$$\sum_{t=1}^{T} P(i(t)=i, \overline{E_i^{\mu}(t)}) \le \sum_{n_i=0}^{T-1} P(\overline{E_i^{\mu}(\tau_{n_i+1})}) = \sum_{n_i=0}^{T-1} P(\hat{\mu}_i(\tau_{k+1}) > x_i).$$

At time τ_{k+1} for $k \ge 1$, $\hat{\mu}_i(\tau_{n+1}) = \frac{\mu_{i,C}}{n_i+1} \le \frac{\mu_{i,C}}{n_i}$. Using Def. 6 to define a bound,

$$\sum_{n_i=0}^{T} P(\hat{\mu}_i(\tau_{k+1}) > x_i) \le \sum_{n_i=0}^{T} P(\frac{\mu_{i,C}}{n_i} > x_i) \le \sum_{n_i=0}^{T} exp(-\frac{n_i \mu_{i,C} \delta_i^2}{3}).$$

Letting $\delta_i = \frac{\Delta_i}{4}$,

$$\sum_{n_i=0}^{T} exp(-\frac{n_i\mu_{i,C}\delta_i^2}{3}) \leq 1 + \sum_{n_i=1}^{T} exp(-\frac{n_i\mu_{i,C}\delta_i^2}{3})$$
$$\leq 1 + \frac{48}{\mu_i\Delta_i^2}.$$

Thus, the bound is

$$\sum_{n_i=0}^{T} P(i(t) = i, \overline{E_i^\mu(t)}) \leq 1 + \frac{48}{\mu_i\Delta_i^2}.$$

□

C. ADVERSARIAL REGRET

The adversarial regret analysis approach also extends on the work of [29], but an adversary has a limited budget to influence a bandit algorithm towards non-optimal arms.

Proof of Theorem 2. Following the same approach as Theorem 1, we find regret bounds for individual terms of the expected number of arm pulls in (4). Bounds for the first two terms are found like Theorem 1, the third term is bounded by Lemma 5 to show adversary influence. Plugging in these terms,

$$\mathbb{E}[n_i(T)] \leq \mathcal{O}(1)+1+\frac{4\,ln(T)}{C}+1+max\{\frac{H}{2\Delta_i}, \frac{H\sigma logT}{\Delta_i}\}$$
$$= \frac{4ln(T)}{C} + max\{\frac{H}{2\Delta_i}, \frac{H\sigma logT}{\Delta_i}.\}$$

Following with standard regret, the bound is

$$\mathbb{E}[R(T)] \leq \sum_i \Delta_i\mathbb{E}[n_i(T)] \leq \sum_i \frac{4ln(T)}{C}\Delta_i +$$
$$max\{\frac{H}{2}, H\sigma logT\}$$

□

Lemma 5.
$$\sum_{t=1}^{T} P(i(t) = i, \overline{E_i^\mu(t)}) \leq max\{\frac{H}{2\Delta_i}, \frac{H\sigma logT}{\Delta_i}\} + 1$$

Proof of Lemma 5. We find a function that bounds $n_i(t)$ for each arm $Q_i(T)$, but under an adversarial setting like Lemma C.5 in [29].

$$\sum_{t=1}^{T} P(i(t) = i, \overline{E_i^\mu(t)})$$
$$= \sum_{t=1}^{T} P(i(t) = i, n_i(t) \leq Q_i(T), \overline{E_i^\mu(t)})$$
$$+ \sum_{t=1}^{T} P(i(t) = i, n_i(t) > Q_i(T), \overline{E_i^\mu(t)})$$

The first term is bounded by $Q_i(T)$, but the adversary influences the second term.

$$\sum_{t=1}^{T} P(i(t) = i, n_i(t) > Q_i(T), \overline{E_i^\mu(t)})$$
$$\leq \sum_{t=1}^{T} P(\hat{\mu}_i + \beta_i \geq x_i | n_i(t-1) \geq Q_i(T))$$

To represent the adversary's influence in the second term, we establish how the adversary influences $\hat{\mu_i}$ with the security algorithms from μ_i. Since any large changes in reward are removed by algorithm 3, $\beta_i \leq H$. The bias introduced by f_i in Algorithm 2 also limits the adversaries attacks by $\frac{f_i}{C}$, where we assume $f_i \leq \frac{C}{2}$ for $i \neq 1$. With both of these limits,

$$\beta_i = \frac{f_i}{C}(\frac{B_i}{n_i(t)}) \leq \mu_i \leq \frac{f_i}{C}H_i \leq \frac{H_i}{2\Delta_i}$$

β_i is bounded by $\frac{H\sigma\ logT}{\Delta_i}$,

$$Q_i(T) = max\{\frac{H}{2\Delta_i}, \frac{H\sigma logT}{\Delta_i}\}.$$

Using the above bound and Def. 6,

$$\sum_{t=1}^{T} P(\hat{\mu}_i + \beta_i \geq x_i | n_i(t-1) \geq Q_i(T))$$
$$= \sum_{t=1}^{T} P(\hat{\mu}_i + \frac{f_i}{C}(\frac{B_i}{n_i(t-1)}) \geq x_i | n_i(t-1) \geq Q_i(T))$$
$$\leq \sum_{t=1}^{T} exp(-\frac{\hat{\mu_i}(\frac{f_i}{C}(\frac{B_i}{n_i(t-1)}))^2}{3}) \leq \frac{1}{T} \leq 1$$

Plugging in the appropriate terms, the bound is

$$\sum_{t=1}^{T} P(i(t) = i, \overline{E_i^\theta(t)}, E_i^\mu(t)) \leq max\{\frac{H}{2\Delta_i}, \frac{H\sigma logT}{\Delta_i}\}+1.$$

□

CHARLES C. RAWLINS is a PhD Computer Engineering student at Missouri University of Science and Technology. He received his B.S. in Electrical Engineering from Montana Technological University with an interest in Computer Science and high honors. His research interests involve Internet of Things devices and security. Some of his hobbies include scuba diving and learning about cybersecurity.

DR. S. JAGANNATHAN is at the Missouri University of Science and Technology (former University of Missouri-Rolla) where he is a Rutledge-Emerson Distinguished Professor of Electrical and Computer Engineering. He served as a Site Director for the NSF Industry/University Cooperative Research Center on Intelligent Maintenance Systems. His research interests include machine learning and cyber-physical system security, neural network control, prognostics/bigdata analytics, and autonomous systems/robotics.