SM-SIM: A Simulator for Analyzing Selfish Mining Attacks in Blockchain Systems

Yannik Sproll*, Robert Heinrich*, Lan Bao Quang Le*, Niclas Kannengießer*
*Karlsruhe Institute of Technology, Karlsruhe, Germany
{contact.research.ys}@gmail.com
{robert.heinrich, lan.le, niclas.kannengiesser}@kit.edu

***Abstract*—Selfish mining attacks threaten the tamper-resistance of blockchain systems with consensus mechanisms based on proof-of-work and probabilistic finality. Optimizing blockchain system configurations can mitigate these attacks. This, however, requires software architects to thoroughly understand the influence of such configurations on the success probability of selfish mining attacks. This work presents SM-SIM, a discrete event simulator to analyze blockchain system configurations and estimate the success probability of selfish mining attacks. SM-SIM includes a meta-model for representing blockchain system configurations and a simulation model for mimicking blockchain operations and attacks. We demonstrate the plausibility and utility of SM-SIM by conducting experiments with different configurations, such as network topologies, block size, latencies, and number of attackers. SM-SIM enables more flexible, architecture-focused analyses and optimization of blockchain system configurations, reducing their vulnerability to selfish mining attacks.**

***Index Terms*—blockchain, security, selfish mining, simulation, software engineering.**

I. Introduction

Many proof-of-work–based blockchain systems with probabilistic finality, such as Bitcoin, rely on the longest-chain rule to determine the mainchain. Such systems keep blockchains tamper-resistant rather than immutable: attackers can tamper with blockchains by performing selfish mining attacks [1]–[3]. In a selfish mining attack, an attacker secretly works on a private branch of the blockchain and only publishes it when it surpasses the public branch in length. In successful selfish mining attacks, all honest nodes in the blockchain system will accept the attacker's private branch as the public one. Attackers may engage in selfish mining to earn block rewards, causing other nodes to waste resources on invalid blocks, or to double-spend tokens [2], [4].

The success of selfish mining attacks strongly depends on the configuration of the attacked blockchain system, such as the system's network topology and the hashing power of attacker nodes [2], [3], [5]. Software architects can decrease the success probability of selfish mining attacks through optimized configurations of blockchain systems. However, finding optimal configurations is challenging due to inherent trade-offs between system characteristics. [3], [6]. To better protect blockchain systems from selfish mining by design, software architects must thoroughly understand the influence of blockchain system configurations on the success probability of selfish mining attacks.

Analyses of real-world blockchain systems can be a powerful means to better understand the influence of blockchain system configurations on the success probability of selfish mining attacks. In large-scale blockchain systems like the Bitcoin system, however, such analyses often produce only rough estimates because it is difficult to monitor and control the blockchain system behavior. For example, the topology of the Bitcoin system—which can strongly influence the likelihood of selfish mining attacks [1], [3], [7], [8]—is dynamic and hard to manipulate in controlled experiments. Private test networks offer an alternative by allowing more control over blockchain system configurations, such as in terms of block size and network topology. However, benchmarking in private test networks is often resource-intensive, which limits in-depth analyses to only a few configurations. To overcome these shortcomings, simulations offer a powerful and resource-efficient approach to blockchain system analysis. Simulations enable controlled experiments, allowing researchers to test different configurations and attack scenarios. Simulations also reduce the effort required to deploy real-world blockchain systems and adjust their configurations.

Although many simulation tools [9]–[11] help better understand the influence of blockchain system configurations on system performance, only a few tools [8], [12], [13] support investigations of the influence of different blockchain system configurations on the success probability of selfish mining attacks. Simulators that support analyses of such attacks, however, fall short in terms of flexibility in configuring blockchain systems on the software architecture level. To support more thorough analyses of the influence of blockchain system configurations on the success probability of selfish mining attacks, we approach the following research question: *What is a design of a flexible simulation tool for investigating the influence of different blockchain system configurations on the success probability of selfish mining attacks?*

We developed SM-SIM[1], a simulation tool composed of a meta-model and a simulation model. Based on the meta-model, we developed concrete blockchain system models. Using the blockchain system model in combination with the simulation model, we simulated selfish mining attacks on blockchain systems with different configurations to demonstrate the plausibility and utility of SM-SIM.

[1]SM-SIM is available at https://tinyurl.com/yadc7bam

Our main ambition in this work is to support more thorough analyses of the influence of blockchain system configurations on the success probability of selfish mining attacks. The main contributions of this work are as follows. First, we introduce a meta-model that captures the abstract software architecture of blockchain systems with proof-of-work consensus mechanisms and probabilistic finality, utilizing the longest chain rule to determine the mainchain. The meta-model helps develop representations of such blockchain systems for simulations. Second, by presenting a simulation model, we enable the use of blockchain system models derived from the meta-model for flexible simulations in several configurations. SM-SIM provides a novel toolset for in-depth analysis of how blockchain system configurations influence the success probability of selfish mining attacks.

The remainder of this work is structured into five sections. Section II describes foundations of selfish mining in blockchain systems and summarizes the state-of-the-art blockchain simulators on selfish mining. We elucidate the meta-model used in SM-SIM to develop blockchain system models in section III. Section IV describes how SM-SIM uses blockchain system models to simulate blockchain system behavior. Next, we demonstrate the plausibility of the results produced by SM-SIM and the utility of SM-SIM for investigating the influence of blockchain system configurations on the success probability of selfish mining in section V. In section VI, we discuss our principal findings and explain the contributions and limitations of this work. Moreover, we outline future research directions.

II. Selfish Mining Attacks

Selfish mining attacks can compromise blockchain systems that operate as replicated state machines, rely on proof-of-work, and determine the mainchain based on the longest branch. A prominent blockchain system vulnerable to selfish mining attacks is Bitcoin [2], [14]. A basic selfish mining attack unfolds in three phases, as described below.

In the first phase, an attacker node A waits until it receives an honest block and appends it to its local blockchain. This block is referred to as the *attack-origin-block*.

In the second phase, starting from the attack-origin-block, A forks its local blockchain into a public and private branch (see Figure 1). A appends blocks from honest nodes to its local public blockchain while simultaneously extending its private branch. To replace the public branch, A continues working on the private branch until it exceeds the public one by m blocks.

In the third phase, A publishes its private branch and waits for the majority of nodes to adopt it as the mainchain.

The success probability of selfish mining attacks strongly depends on blockchain system configurations and attacker capabilities, such as hashing power and network topology position [3], [7], [8]. Previous simulation approaches help generate valuable insights into how blockchain system configurations influence selfish mining success, but often neglect critical aspects of software architecture, limiting their flexibility for detailed system-level analysis. This limitation makes thorough and flexible analysis challenging. Most simulators focus on network-layer behavior [3], [8] or directly estimate attack success probability [13], yet they lack the ability to model and manipulate blockchain system configurations at the software architecture level. This restricts their utility for investigating how architectural design choices impact the robustness of blockchain systems against selfish mining attacks. More flexible and architecture-focused simulation tools are needed to enable more thorough investigations of blockchain system configurations and their security implications.

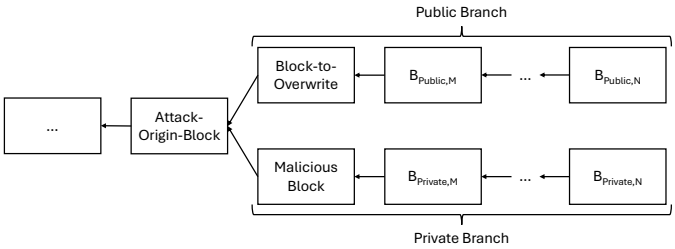


Fig. 1: Forked local blockchain on an attacker node with a public branch, composed of blocks B_{Public}, and a private branch, composed of blocks $B_{Private}$.

III. Blockchain System Model

As a foundational part of SM-SIM, we developed a meta-model that represents the core components of the software architecture of blockchain systems. The meta-model serves as a foundation for modeling blockchain systems with different configurations. To develop the meta-model, we extended the Palladio Component Model (PCM) [15], a general meta-model for representing component-based software architectures.

To develop blockchain system models, the meta-model offers four view types: *repository*, *node system*, *blockchain system*, and *deployment*. These views are interconnected and collectively define the structure of a blockchain system [16]. In model-driven software engineering, a view type defines the elements and relationships that a view can contain [17].

The ***repository view type*** describes components and their interfaces. Within the repository view type, we introduce the *BlockchainSystemNodeComponent*, a new component type that represents the functionalities of nodes in blockchain systems. Other than conventional clients, validating nodes operate on the replicated state machine concept and utilize proof-of-work consensus with the longest chain fork resolution rule. Figure 2 illustrates the structure of a validating node.

Within the *BlockchainSystemNodeComponent* are the *MiningProcessComponent* and *BlockValidatorComponent*. A MiningProcessComponent represents the block production process based on proof-of-work (mining) executed on a node. The attribute *IsMiningProcessEnabled* controls whether the component computes block hashes in block production to propose new blocks [18]. This allows differentiation between validating and non-validating nodes. A *BlockValidatorComponent* represents a component that implements the validation of blocks a node receives from its adjacent node.

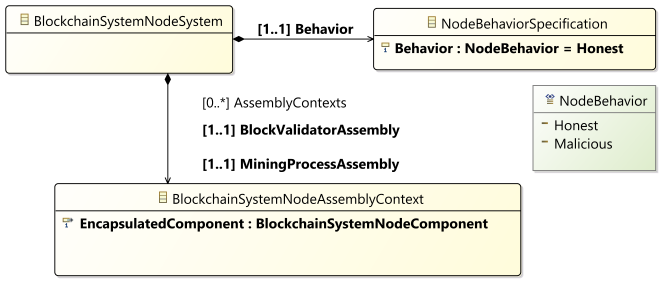


Fig. 2: Structure of the blockchain system node entity.

The ***node system view type*** describes which *BlockchainSystemNodeComponents* are assembled on a blockchain system node. The root element of the node system view type is the *BlockchainSystemNodeSystem*. It represents the software system that is running on a blockchain system node. It has a reference to a *NodeBehaviorSpecification* and has a reference to two *BlockchainSystemNodeAssemblyContexts* for the node's *MiningProcessComponent* and *BlockValidatorComponent*. A $BlockchainSystemNodeAssemblyContext$ represents an instance of a *BlockchainSystemNodeComponent*. A *NodeBehaviorSpecification* specifies if the node system behaves maliciously.

The ***blockchain system view*** type describes a full blockchain system by referencing all other view types and specifying behavior constraints that apply system-wide. The *BlockchainSystem* is the root model element and represents an entire blockchain system. The *BlockchainSystemSpecification* specifies behavioral constraints that apply system-wide.

The ***deployment view type*** describes how the system will be deployed, including the network connections between nodes. It has the following three sub-view types.

The *node environment sub-view type* describes the execution environment to which a node system is deployed. The *NodeEnvironment* represents the execution environment in which a *BlockchainSystemNodeSystem* is deployed. It is the root model element of the node environment view type. The *NodeResourceContainer* represents a container that can process workloads.

The *node allocation sub-view type* describes the execution environment for the blockchain node system and how its components are mapped to the node resource containers. The $NodeAllocationContext$ represents an allocation of a *BlockchainSystemNodeComponent* to a *NodeResourceContainer*. The *NodeAllocation* represents the allocation of a *BlockchainSystem-NodeSystem* in a *NodeEnvironment*. A $NodeAllocationRepository$ represents a collection of reusable allocation definitions, each specified in one *NodeAllocation*.

The *P2P network sub-view type* describes the topology of the P2P network of a blockchain system and the nodes' positions in the P2P network. $LinkSpecification$ specifies the transmission properties of the links that connect nodes in the P2P network. $NetworkTopology$ represents the structure of the P2P network.

IV. Simulation Using SM-SIM

This section describes the simulation model that SM-SIM uses to mimic blockchain system behavior and the simulation process. Then, the computation of success probabilities of selfish mining attacks is explained.

A. Blockchain System Simulation

This section describes the foundations of the simulation model for blockchain systems used in SM-SIM. Then, the entities of the simulation model are discussed in section IV-A2. Section IV-A3 details the behavior of honest nodes.

1) Simulation Model: The discrete event simulation model represents the state of a blockchain system over time. We denote the current simulation time t_c as a global discrete state variable $CLOCK$. An event is represented as a tuple $e = (e_t, t_o)$, where e_t denotes the event type and t_o specifies the time of occurrence. We use a *future event list (*FEL*)* to schedule events e:

$$FEL = \{(e_1, t_1), \ldots, (e_n, t_n) | t_1 \leq \cdots \leq t_n\} \quad (1)$$

2) Simulation Behavior: The *mining process* refers to the process of producing new blocks. In the simulation, we assume a constant mining difficulty. However, the mean block time is an input parameter to the simulation. Because the realistic inter-arrival times of blocks are exponentially distributed [19], we model the block times with a Poisson process [13] $\{N(t), t \geq 0\}$. Suppose $\lambda = \frac{1}{t\_mean}$ is the Poisson parameter of $N(t)$, where t_{mean} is the desired mean block time. The desired mean block time equals the expected block time: $E[N(t)] = \lambda * t = \frac{1}{t_{mean}} * t$.

$g(\lambda)$ is a generator function that returns an exponentially distributed value with the parameter λ. When a validating node starts the mining process, a *block mined event* $e_{block\_mined} = (block\_mined, CLOCK + g(\lambda))$ will be added to $FEL = FEL \cup \{e_{block\_mined}\}$.

In addition to proposing blocks, *validating nodes* validates block hashes and transactions batched into blocks. Validating nodes compute the occurrence time t_o by adding the block validation duration $d_{validation}$ to the current simulation time $t_o = CLOCK + d_{validation}$.

The *P2P network interface* of a node n computes the transmission time $t_{time} = t_t + l + size\_of(m) * t_h$, where l is latency and t_h denotes the throughput of the link between n and its adjacent node.

The *block transmission protocol* defines how a node transmits a block to another node. The block transmission protocol uses three types of messages: *inv_message*, *get_data_message*, and *block_message*. Figure 3 illustrates a block transmission between two nodes.

Blockchain is a simulation entity that reflects the local blockchain maintained on a node. For honest nodes, the blockchain is a chain of linked blocks without forks. For attackers, the simulation entity stores the public branch and private branch as a fork (see Figure 1).

A *stale block pool* stores blocks that arrive out of order, such as when a block is received before its previous block.

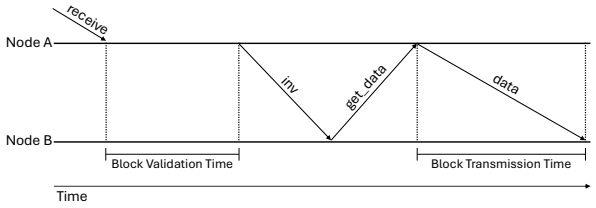


Fig. 3: Communication between nodes in block propagation.

3) Honest Node Behavior: When an honest node receives a new block, a $BlockReceivedEvent$ is scheduled by the block transmission protocol. After an honest node has validated a block, the protocol schedules a $blockvalidatedevent$. If the node has the required previous block, the block is stored in the blockchain; otherwise, the node stores the block in its stale block pool. The $blockminedevent$ is triggered when an honest node has produced a block. Then, the node adds the block to its local blockchain. Honest nodes multicast each validated and produced block to their adjacent nodes using the block transmission protocol.

B. Simulation Process

The simulation process starts with *generate_initial_events()*, a *block mined event* for each honest blockchain system node is generated. Until either the *future event list (FEL)* is empty or the *should_terminate()* call evaluates to $true$, this process continues to generate events (see Algorithm 1). The simulation terminates when its state variables, checked by *should_terminate()*, meet the following conditions:

- The attacker published its private branch.
- All nodes must have received all produced blocks from the attacker.
- All honest nodes have an unforked blockchain in which the longest chain is *n* blocks longer than the second-longest chain.

When a node has a blockchain of at least length x, the *should_terminate()* triggers an event to stop the process. First, all events from the FEL are scheduled in the simulation at the time *t*. Second, the state changes of the events are applied by the simulation. Before updating the global time variable `CLOCK` to the next scheduled event, the simulation schedules the changed events. The simulation executes the next iteration of the loop after updating `CLOCK`. Since we simulate selfish mining attacks, the result of a simulation is either an attack success or an attack failure.

Algorithm 1 Event Processing Loop

```
1: procedure RUN_SIMULATION()
2:     generate_initial_events()
3:     while |FEL| > 0 AND NOT should_terminate() do
4:         t ← get_current_time()
5:         events ← FEL.get_events_at(t)
6:         on_events_occured(events)
7:         advance_clock()
8:     end while
9:     determine_results()
10: end procedure
```

C. Simulation of Selfish Mining Attacks

We simulate a specific type of selfish mining attack that is used for double-spending and tampering with the blockchain. Double-spending refers to the fraudulent multiple spending of the same tokens [6]. That type of selfish mining attack is performed in four phases. In the *first phase*, the attacker waits until it receives an attack-origin-block, or an honest block. Because the selfish mining attack is created immediately behind the attack-origin-block by forking, the attack-origin-block has two successor blocks: one is part of the public branch, and the other belongs to the private branch.

In the *second phase*, A creates a private branch by producing a successor block immediately after the attack-origin-block We term this successor the *block-to-overwrite*, as the attacker intends to replace it after it has been accepted into honest nodes' blockchains. A appends the block-to-overwrite to its local blockchain and publishes it on the network.

In the *third phase*, the attacker creates a private branch that is envisioned to replace the public branch. The first block of the private branch is a successor of the attack-origin-block. The attacker produces blocks to be appended to the private branch until the following conditions are met. First, the public branch must contain at least n successor blocks of the block-to-overwrite. n is the number of blocks that confirm the transaction in the block-to-overwrite. n specifies how long the recipient of the malicious transaction waits before releasing the product the attacker bought. Second, the private branch must be m blocks longer than the public branch.

In the *fourth phase*, the attacker publishes its private branch.

D. Computation of the Success Probability

We use the Monte Carlo simulation method to estimate the success probability of selfish mining attacks [20]. Using the Monte Carlo method, the execution of a simulation round corresponds to an evaluation of the random value generator function. In SM-SIM, a simulation round is a mapping $S_{round} : X \to \{0, 1, \#\}$, where X is the space that contains all blockchain system models. The attack is successful if $S_{round} = 1$ and unsuccessful if $S_{round} = 0$. The result of the simulation round is not defined when $S_{round} = \#$. We denote N as the number of simulation rounds for a simulation run. A simulation run is defined as a mapping $S_{run} : X \times N \to [0, 1], (x, n) \mapsto E[S_{round}(x)]$. Next, we computed a sample S of N simulation round evaluations using $S = \{S_{round,1}(x), ..., S_{round,N}(x)\}$. Then, we computed the success probability as the ratio of successful selfish mining attacks to the total number of unambiguous simulation rounds:

$$E[S_{round}(x)] = \frac{\Sigma_{x \in S} 1_{\{1\}}(x)}{\Sigma_{x \in S} 1_{\{1,0\}}(x)} \tag{2}$$

V. Demonstration

We conducted experiments to demonstrate the plausibility of the results produced by SM-SIM. For the plausibility demonstration, we used three network topologies (see Figure 4): Net, R3, and Ring. We compared the simulation results with those from existing research on selfish mining to demonstrate that the results are consistent and plausible. To demonstrate the utility of SM-SIM, we examined how different blockchain system configurations affect the success probabilities of selfish mining attacks. The following section outlines the experimental design and presents the results.

A. Experimental Design

As SM-SIM uses a stochastic simulation model, we ran multiple simulation rounds until the results converged. We simulated blockchain systems with three configurations: *Net_12*, *Net_21*, and *Net_40*. The configurations correspond to the Net topology with hashing power shares ranging from 12% to 40%, respectively. The network topology of the *Net* model consists of a large, single subgraph defined by specific constraints. We assume this system model leads to a high deviation in the calculated attack success probabilities. For each variation of the *Net* system model, we ran 500, 1000, 2000, and 5000 simulation rounds. Per number of simulation rounds, we collected a sample of ten in each system model. For each simulation round, we generated a new network with the same topology to reduce bias from individual network configurations (e.g., positions of nodes).

We used the *coefficient of variation (CV)* to compute the convergence of the simulation results. CV is defined as $CV = \frac{\sigma}{\mu}$, where σ and μ are the standard deviation and mean of the sample. When the number of simulation rounds increases, a decreasing CV value indicates convergence. We set the threshold for the CV value to 0.05. After 15,000 simulation rounds per simulation run, if the CV does not fall below this threshold, we used the absolute standard deviation to assess the convergence of the simulation model. In *Net_40* model, for example, the CV value at 500 rounds was 0.0315, which is below the threshold of 0.05. It remained below 0.05, indicating strong convergence. Starting with 0.2304 at 500 rounds, the CV value of *Net_21* model decreased to below 0.05 between 10,000 and 15,000 rounds. In the *Net_12* model, the CV value at 500 rounds was 1.527, well above the 0.05 threshold. This value gradually decreased to 0.195 at 15,000 rounds but never fell below the threshold.

1) Plausibility Demonstration: We conducted a plausibility test to demonstrate that SM-SIM accurately captures the impact of blockchain system configurations on the success probability of selfish mining attacks. For the plausibility demonstration, we manipulated three common parameters of blockchain system configurations that influence the success probability of selfish mining attacks [7], [8], [21]. The first parameter is the attacker's *share of hashing power*, which directly affects the attacker's ability to produce a new blocks. The second parameter is *block propagation delay*. The last parameter, *network topology*, influences the success probability based on the position of the attacker in the network. Table I provides an overview of the naming conventions for the simulated blockchain system models, based on topology and attacker hashing power share.

As we varied the block propagation delay, we simulated a *base* version and a *delayed* version for each blockchain system model presented in Table I. To ensure practical execution times, we limited the maximum blockchain length to 35. The total hashing power in the simulations was 20 MH/s, and the number of nodes in the network was 15. The latency and throughput of the *base* version are 100*ms* and 10*kb/s*, respectively. They are 200*ms* and 5*kb/s* in the *delayed* version. We used those configurations as they enable comparability of the simulation results with extant research (e.g., [22]).

2) Utility Demonstration: For each blockchain system model, we collected the success probability of the base version, the delayed version, and the difference between their probabilities. The block transmission and validation time were modified in the simulations to vary the block propagation delay. To the best of our knowledge, no publicly available and sufficient reference data exists to validate the results of the simulation model, nor is there a formal definition of the success probability of selfish mining attacks on blockchain systems. Therefore, we used a binary variable with the values *expected* and *unexpected*.

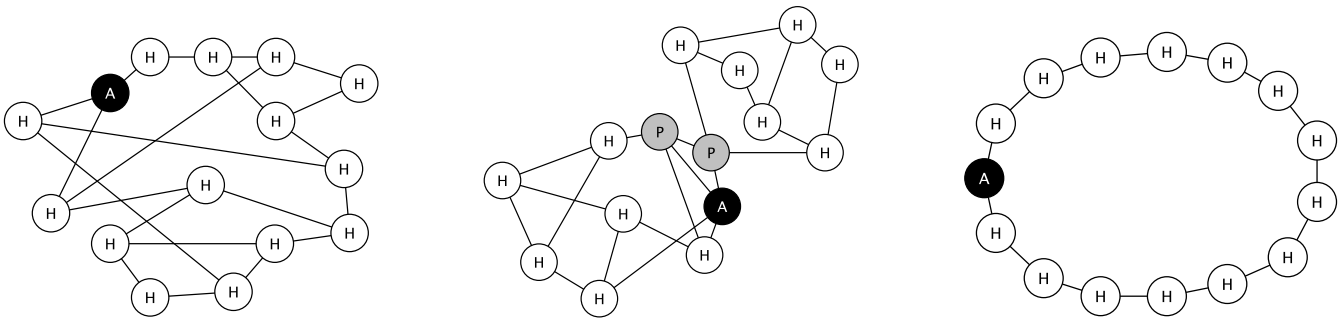
We manipulated the blockchain system configurations by varying the block creation interval (10 min, 2.5 min, 1 min, and 20 s) and seven block sizes (0.075 MB, 0.25 MB, 0.5 MB, 1 MB, 2 MB, 4 MB, and 8 MB) as per [3]. In the simulations using these configurations, the attacker has the same hashing power as the honest nodes.

The lower bound for successful selfish mining attacks is approximately 21% to 23% of the hashing power of a blockchain system [14]. We simulated two scenarios with different hashing power of honest nodes and attackers. In the first scenario, the honest nodes control 21% of the hashing power, while the attacker nodes control 50%. In the second scenario, these roles are reversed. In both scenarios, we used a block size of 0.075 MB for the block creation intervals 10 min, 2.5 min, 1 min, and 20 s.

In the simulations with multiple competitive attackers, we used a simulation model with a block creation interval of 10 min and a block size of 0.075 MB. We simulated 1, 2, and 3 simultaneous and competing attackers using the network topologies Net, R3, and Ring (see Figure 4).

		Hashing Power Share of the Attacker				
		12%	**21%**	**30%**	**40%**	**50%**
Topology	**Net**	Net_12	Net_21	Net_30	Net_40	Net_50
	R3	R3_12	R3_21	R3_30	R3_40	R3_50
	Ring	Ring_12	Ring_21	Ring_30	Ring_40	Ring_50

TABLE I: Overview of blockchain system models based on network topologies and share of an attackers' hashing power per topology.



(a) Net topology (b) R3 topology (c) Ring topology

Fig. 4: Overview of exemplary network topologies used in the demonstration. H denotes an honest node, A denotes an attacker, and P indicates a proxy node.

B. Experimental Results

This section presents the experimental results of the plausibility and utility demonstrations.

1) Plausibility Demonstration: The profitability threshold for the attack is 25% of hashing power [2]. SM-SIM demonstrates that this threshold can be lowered to 21% (see Figures 6-9), which is consistent with extant research [4], [23], [24]. This threshold can be reduced to around 14% when the attack is carried out by multiple attackers (see Figure 5). When attackers have more hashing power than honest nodes, the success probability of selfish mining attacks increases (see Tables II). When the overall hash power of the attackers exceeds 50%, the success probability increases strongly, which is consistent with existing research [2], [12]. If honest nodes have a total hashing power higher than that of attackers (see Figure 10, the success probability decreases. This shows that SM-SIM produces plausible simulation results, compared to influences of blockchain system configurations on selfish mining success reported in previous research [7].

In line with prior research [2], [12], [14], [24], the simulation results indicate that the network topology influences the success probabilities of selfish mining attacks (see Figures 5-10). The demonstration results indicate that the benefit threshold for the selfish mining attack is at least 21% of hashing power, which is aligned with the value of 25% in the original description of the selfish mining attack [2]. Consistent with [24], the results of SM-SIM demonstrate the success probability of the attack will be sharply increased when the hashing power of the attacker reaches 50%. As the results produced by SM-SIM are consistent with those of existing research, we consider SM-SIM to produce plausible results.

		Hashing Power Share of the Attacker and Honest Nodes				
		12%	**21%**	**30%**	**40%**	**50%**
Net	**Base**	0.001178	0.36578	0.197644	0.407006	0.483752
	Delayed	0.002667	0.61333	0.270244	0.449252	0.492434
	Difference	0.001489	0.24756	0.0726	0.042246	0.008682
R3	**Base**	0.001267	0.0424	0.21744	0.423839	0.4887
	Delayed	0.0028	0.072067	0.299247	0.461754	0.49535
	Difference	0.001533	0.029667	0.081807	0.037915	0.0065
Ring	**Base**	0.001222	0.046022	0.242439	0.443019	0.491667
	Delayed	0.001333	0.0464	0.262733	0.4508	0.484852
	Difference	0.000111	0.000378	0.020294	0.007781	0.006816

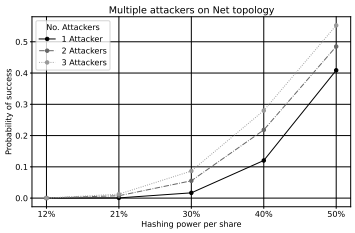
TABLE II: Comparison of success probabilities of selfish mining attacks in different blockchain systems with different network topologies (i.e., Net, R3, and Ring), hashing power of the attacker, and network delays.

2) Utility Demonstration: To demonstrate the added value of SM-SIM, the following presents simulation results regarding the influence of exemplary network topologies and hashing power distributions on the success probability of selfish mining attacks.

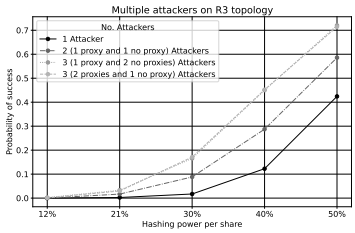
a) Influence of Network Topologies on Success Probabilities of Selfish Mining Attacks: Table II presents the simulation results for a blockchain system with a block size of 0.075 MB and a block creation interval of 1.5 seconds, for the network topologies Net, R3, and Ring. The simulation results show that blockchain systems with Ring topology are most vulnerable to selfish mining attacks. Blockchain systems with the *Net* topology seem less prone to successful selfish mining attacks.

For multiple competing attackers, the success probability per attacker strongly increases (see Figures 5a, 5b, and 5c), especially in blockchain system models with Ring topology.

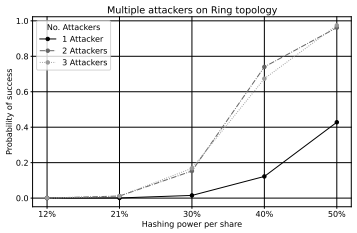
b) Influence of Hashing Power Distribution on Success Probabilities of Selfish Mining Attacks: From 0.1178% at 12% hashing power share in the basic version of the Net System, the success probability reaches 48.3752% at 50% hashing power share. In the delayed system, the attacker has higher success probabilities. The increase in success probabilities for delayed systems compared to base systems is *expected* because a higher block propagation delay can favor the attacker by favoring forks in the system [4]. When increasing the hashing power share, the probability of success in the delayed system is greater than in the base system in most cases for three topologies, as we *expected*. In the case of 50% hashing power share in the Ring network topology, the success probability for the base model is greater than for the delayed system.



(a) Multiple attackers on Net topology

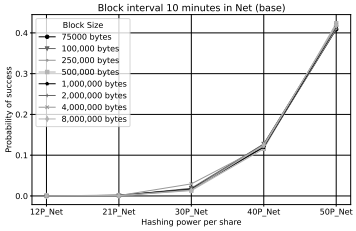


(b) Multiple attackers on R3 topology

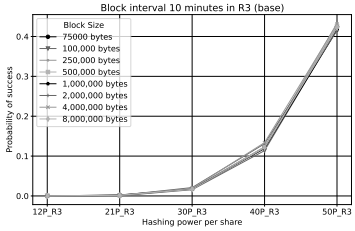


(c) Multiple attackers on Ring topology

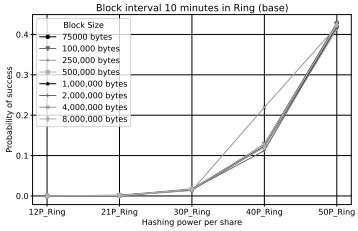
Fig. 5: Multiple attackers on three topologies.



(a) Net topology

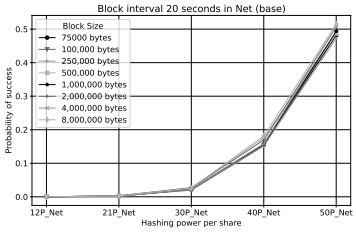


(b) R3 topology

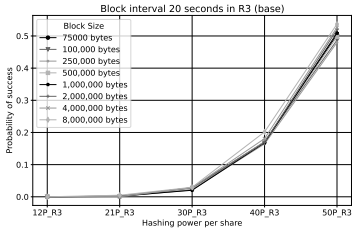


(c) Ring topology

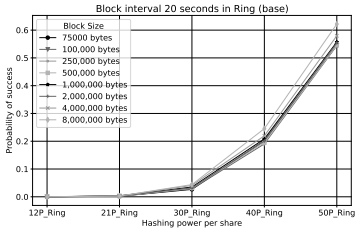
Fig. 6: Bitcoin system with block creation interval 10 min, latency 100 ms, block size from 0.0075 MB to 8 MB.



(a) Net topology

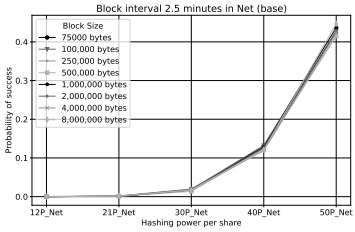


(b) R3 topology

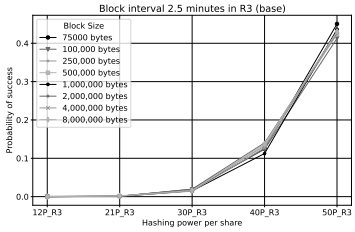


(c) Ring topology

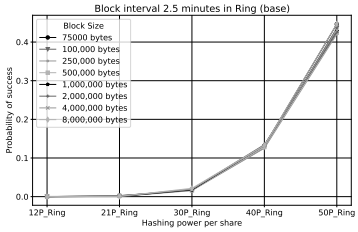
Fig. 7: Bitcoin system with block creation interval 20 sec, latency 100 ms, block size from 0.075 MB to 8 MB.



(a) Net topology

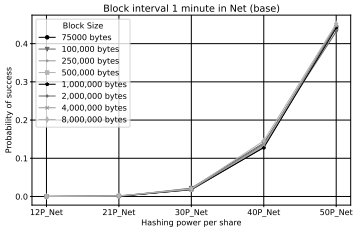


(b) R3 topology

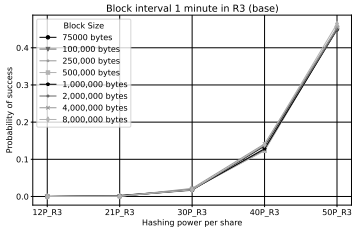


(c) Ring topology

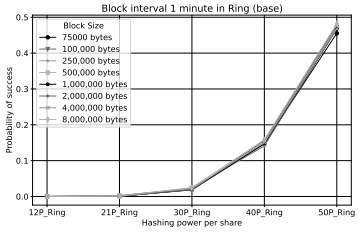
Fig. 8: Bitcoin system with block creation interval 2.5 min, latency 100 ms, block size from 0.075 MB to 8 MB.



(a) Net topology

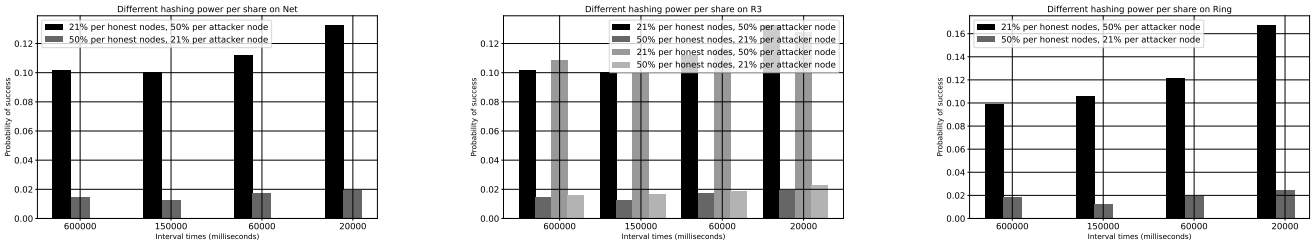


(b) R3 topology



(c) Ring topology

Fig. 9: Bitcoin system with block creation interval 1 min, latency 100 ms, block size from 0.075 MB to 8 MB.



(a) Net topology (b) R3 topology (c) Ring topology

Fig. 10: Relationship between the hashing power of honest nodes and the attacker influences the success probability of selfish mining attacks.

If honest nodes and attackers have different hashing power (see Figure 10), the success probability increases with a decreasing block creation interval. Moreover, if the attacker has more hashing powers than the honest nodes, the probability is significantly higher than its counterpart. Block size does not seem to significantly affect the success probability.

The simulation results showcase the utility of SM-SIM for investigating success probabilities of selfish mining attacks in blockchain systems with different configurations, including software architectures, network topologies, and number of competing attackers.

VI. Conclusions

This paper presents SM-SIM, a blockchain system simulator that enables software architects to evaluate how blockchain configurations influence the success of selfish mining attacks. SM-SIM comprises a meta-model for developing blockchain system models for simulations and a simulation model itself. We demonstrate the plausibility and utility of SM-SIM by simulating selfish mining attacks on blockchain systems with different configurations. The simulations reveal that block size has minimal impact on selfish mining success probabilities, whereas a higher number of attackers significantly increases their effectiveness. Among tested configurations, blockchain systems with a Ring topology are the most vulnerable.

Our primary ambition is to enable more thorough and efficient investigations into the influence of various blockchain system configurations on the success probability of selfish mining attacks. By presenting a meta-model focused on blockchain system software architecture, we lay a foundation for the flexible development of custom blockchain models for simulations. This meta-model defines the key components of blockchain systems and their relationships, supporting detailed modeling of blockchain systems with custom configurations.

SM-SIM also offers a simulation tool for software architects to evaluate the success probability of selfish mining attacks in blockchain systems that use proof-of-work-based consensus mechanisms. It enables the evaluation of different blockchain system configurations in terms of their robustness against selfish mining attacks, which can support the development of blockchain systems that are more robust by design.

While SM-SIM offers flexibility in modeling blockchain configurations, its current scope is limited to proof-of-work-based systems using the longest chain fork resolution rule, such as Nakamoto consensus [18]. Since selfish mining primarily targets proof-of-work–based blockchain systems, this focus is justified. In order to make SM-SIM more versatile, future work could extend the meta-model to simulate blockchain systems with alternative consensus mechanisms that use other leader election protocols and fork resolution rules. This would enable analyses of the influence of a larger number of blockchain system configurations on the success probabilities of different attacks.

We validated the meta-model and the blockchain system model based on software architectures of established systems like Bitcoin. However, we did not quantitatively verify the simulation results in terms of accuracy. Instead, we used existing literature on selfish mining to demonstrate the plausibility of outputs produced by SM-SIM. Future research should focus on empirical validation, comparing SM-SIM's simulated results with data on selfish mining attacks on real-world blockchain systems to assess its accuracy.

In conclusion, SM-SIM provides a valuable tool for software architects and blockchain developers alike, enabling deeper insights into the vulnerabilities of blockchain systems to selfish mining attacks. As blockchain technology continues to evolve, we hope SM-SIM will play a key role in enhancing the security of these systems, fostering the development of more secure blockchain systems.

Acknowledgment

This work was supported by the KASTEL Security Research Labs and the German Federal Ministry of Education and Research and the NextGenerationEU project by the European Union grant number 16KISA086 (ANYMOS).

References

[1] E.-E. Gojka, N. Kannengießer, B. Sturm, J. Bartsch, and A. Sunyaev, *Security in Distributed Ledger Technology: An Analysis of Vulnerabilities and Attack Vectors*, ser. Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2021, vol. 285, p. 722–742.

[2] I. Eyal and E. G. Sirer, "Majority is not enough: bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, p. 95–102, Jun. 2018.

[3] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, New York, NY, USA, 2016, p. 3–16.

[4] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, *Optimal Selfish Mining Strategies in Bitcoin*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, vol. 9603, p. 515–532.

[5] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "Vibes: Fast blockchain simulations for large-scale peer-to-peer networks: Demo," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, ser. Middleware '17, New York, NY, USA, 2017, p. 19–20.

[6] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, "Trade-offs between distributed ledger technology characteristics," *ACM Computing Surveys*, vol. 53, no. 2, Apr. 2020.

[7] M. Saad, L. Njilla, C. Kamhoua, and A. Mohaisen, "Countering selfish mining in blockchains," in *2019 International Conference on Computing, Networking and Communications*, 2019, pp. 360–364.

[8] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," *Simulation Modelling Practice and Theory*, vol. 114, p. 102413, Jan. 2022.

[9] S. Pandey, G. Ojha, B. Shrestha, and R. Kumar, "Blocksim: A practical simulation tool for optimal network design, stability and planning," in *2019 IEEE International Conference on Blockchain and Cryptocurrency*, Seoul, Korea (South), May 2019, p. 133–137.

[10] R. Banno and K. Shudo, "Simulating a blockchain network with simblock," in *2019 IEEE International Conference on Blockchain and Cryptocurrency*, Seoul, Korea (South), May 2019, p. 3–4.

[11] X. Ma, H. Wu, D. Xu, and K. Wolter, "Cblocksim: A modular high-performance blockchain simulator," in *2022 IEEE International Conference on Blockchain and Cryptocurrency*, Shanghai, China, May 2022, p. 1–5.

[12] M. Alharby and A. van Moorsel, "Blocksim: An extensible simulation tool for blockchain systems," *Frontiers in Blockchain*, vol. 3, p. 28, 2020.

[13] P.-O. Goffard, "Fraud risk assessment within blockchain transactions," *Advances in Applied Probability*, vol. 51, no. 2, p. 443–467, Jun. 2019.

[14] S.-N. Li, C. Campajola, and C. J. Tessone, "Statistical detection of selfish mining in proof-of-work blockchain systems," *Scientific Reports*, vol. 14, no. 1, p. 6251, Mar. 2024.

[15] R. H. Reussner, S. Becker, J. Happe, R. Heinrich, and A. Koziolek, *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

[16] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, H. Koziolek, K. Krogmann, and M. Kuperberg, "The palladio component model," 2011.

[17] T. Goldschmidt, S. Becker, and E. Burger, "Towards a tool-oriented taxonomy of view-based modelling," vol. P-201, 2012, p. 59 – 74.

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

[19] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.

[20] A. Barbu, S.-C. Zhu, A. Barbu, and S.-C. Zhu, "Introduction to monte carlo methods," *Monte Carlo Methods*, pp. 1–17, 2020.

[21] J. Göbel, P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, vol. 104, p. 23–41, Oct. 2016.

[22] C. Natoli, P. Ekparinya, G. Jourjon, and V. Gramoli, "Blockchain double spending with low mining power and network delays," *Distributed Ledger Technologies: Research and Practice*, 2024.

[23] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, "A deep dive into blockchain selfish mining," in *ICC 2019-2019 IEEE international conference on communications*, 2019, pp. 1–6.

[24] C. Schwarz-Schilling, S.-N. Li, and C. J. Tessone, "Stochastic modelling of selfish mining in proof-of-work protocols," *Journal of Cybersecurity and Privacy*, vol. 2, no. 2, pp. 292–310, 2022.