On Security Analysis of Proof-of-Elapsed-Time (PoET)

Lin Chen(✉), Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi

Department of Computer Science, University of Houston, Houston, TX 77204, USA
chenlin198662@gmail.com

Abstract. As more applications are built on top of blockchain and public ledger, different approaches are developed to improve the performance of blockchain construction. Recently Intel proposed a new concept of proof-of-elapsed-time (PoET), which leverages trusted computing to enforce random waiting times for block construction. However, trusted computing component may not be perfect and 100% reliable. It is not clear, to what extent, blockchain systems based on PoET can tolerate failures of trusted computing component. The current design of PoET lacks rigorous security analysis and a theoretical foundation for assessing its strength against such attacks. To fulfill this gap, we develop a theoretical framework for evaluating a PoET based blockchain system, and show that the current design is vulnerable in the sense that adversary can jeopardize the blockchain system by only compromising $\Theta(\log \log n / \log n)$ fraction of the participating nodes, which is very small when n is relatively large. Based on our theoretical analysis, we also propose methods to mitigate these vulnerabilities.

1 Introduction

Blockchain technology is believed to have the potential to revolutionize various sectors including financial, manufacturing, transportation, and agriculture (e.g., [28]). As more applications are built on top of blockchain based systems, performance becomes a major bottleneck; and many efforts have been spent in designing a new blockchain backbone to improve the latency, throughput, and scalability (e.g., [8,13,24,30]). Although these works adopt different technology routes, they all try to address the performance problem through purely software based approaches.

Trusted computing technology provides another opportunity to improve the performance of a blockchain. Trusted computing leverages special hardware properties to provide a trusted execution environment where adversaries cannot tamper the execution of an application. All main processor vendors such as Intel, AMD, and ARM have their own trusted computing solutions. Despite differences in design and implementation details, they provide essentially similar security features [1,2,9]. When trusted computing technology is applied to the blockchain, a blockchain client can run inside a trusted environment (e.g.,

P. Spirakis and P. Tsigas (Eds.): SSS 2017, LNCS 10616, pp. 282–297, 2017.
https://doi.org/10.1007/978-3-319-69084-1_19

enclave, secure world, or compartment) with certain security assurance; and the trusted computing environment ensures all embedded protocols will be faithfully followed.

Based on its trusted computing platform SGX, Intel proposed the concept of "proof-of-elapsed-time" (PoET) for blockchain construction [19]. The basic idea is that each node generates a random number to determine how long it has to wait before it is allowed to generate a block. The generation of random numbers is based on certain distribution specified by the system in advance. When a new block is submitted to the system, SGX helps the node creating the block to generate a proof of the waiting time. This proof can be easily verified by other nodes with SGX technology. A statistical test is used to determine whether the waiting time indeed follows the specified distribution. Compared with other blockchain schemes like PoW (proof-of-work, see Sect. 2.1 for details), this approach has two major advantages: (i) Efficiency. PoET does not require participating nodes to carry out expensive computation workload before creating a new block; (ii) Fairness. PoET achieves the goal of "one CPU one vote", which was originally proposed in Nakamoto's paper on Bitcoin [25], but was not fully achieved before.

However, SGX and other trusted computing technologies are not 100% reliable. Especially, they may be vulnerable to sophisticated adversaries with necessary resources and skillsets. It is thus a natural question whether the system remains secure when the underlying trusted computing components of some nodes are compromised. Similar problems have been addressed for systems where proof-of-work is implemented, see, e.g., [11,15,21]. However, there is no theoretical result for a PoET based system and its security is unknown. The major contribution of this paper is to develop a theoretical framework to evaluate Intel's PoET scheme and its variants, and carry out security analyses based on such a framework. Our results demonstrate that the current scheme/protocol implemented on Intel's SGX platform could be vulnerable to security attacks. More specifically, adversaries can hijack the system by simulating the fastest honest node in the system if they successfully compromise $\Theta(\frac{\log \log n}{\log n})$ fraction of the nodes (where n is the total number of nodes in the system). As compromised nodes are merely simulating the fastest honest node, no statistical test can distinguish them. Note that $\Theta(\frac{\log \log n}{\log n})$ is not a constant, which contrasts sharply with the constant threshold of 50% in proof-of-work based systems such as Bitcoin.

Our results suggest two potential approaches that may lead to a constant threshold. One is to alter the probability distribution currently implemented in Intel's platform. Indeed, we show that the more "concentrated" this probability distribution is, the higher the threshold will be. This guides the selection of the probability distribution from the perspective of security. The other approach is to allow the statistical test to reject blocks that are generated by a certain fraction of nodes, even if some honest nodes may be included. In fact, the bound of $\Theta(\frac{\log \log n}{\log n})$ still applies if the statistical test is only allowed to reject blocks generated by a constant number of nodes. Therefore, using this approach, the

statistical test needs to reject blocks generated by a significant amount of nodes, regardless they are honest or not. In summary, our main contributions in this paper include:

- We develop an abstract model of PoET based blockchain systems that capture the critical features of PoET, which opens the door for theoretical analysis and assessment of PoET;
- We analyze design of Sawtooth Lake Scheme and find that the current protocol is vulnerable even under the scenario that only a very small fraction of nodes are compromised;
- Based on our analysis, we provide security guidelines and suggestions for designing blockchain schemes based on the concept of PoET.

It is important to point out that our analysis of PoET focuses on theoretical and protocol design level. The analysis does not depend on any specific hardware implementation flaws or vulnerabilities and thus holds generally.

The remainder of the paper is organized as follows: Sect. 2 provides a short review of Intel's Sawtooth Lake scheme, an implementation based on PoET. Section 3 describes the mathematical tools used in the analysis of PoET. Section 4 provides an abstract model of PoET. A rigorous analysis of PoET is given in Sect. 5. We review related works in Sect. 6 and conclude the paper in Sect. 7.

2 Blockchain and PoET with Trusted Computing

2.1 Blockchain and Proof-of-Work

Blockchain technology was first introduced by Bitcoin as a distributed bookkeeping system [25]. Briefly speaking, a blockchain is a chain of blocks where each block contains a set of records (e.g., records for transactions) together with the hash value of the previous block. Users[1] keep adding blocks to the blockchain through a procedure called "mining". Ideally, a blockchain remains a chain. In case that a branch occurs (e.g., multiple users add blocks simultaneously), the "longest-chain" rule is applied, that is, users will follow the branch containing the most number of blocks. Other branches will be discarded.

Since blocks are linked with hash values, an attacker cannot alter or remove an existing block stored on the blockchain. However, an attacker may choose to branch at a certain block. If he/she successfully generates a longer branch thereafter, all transactions occur in the original branch will be discarded and system is thus compromised. To ensure the security of the whole system, we need a way to prevent users from generating an arbitrary number of blocks in a short time. The most widely used scheme is proof-of-work. Using this scheme, every user needs to solve a computation intensive problem in order to add one block. Solving such a problem requires a lot of computation. If an attacker aims at generating a longer branch, he/she needs more computational power than all the other honest users. It was shown in [25] that a proof-of-work based blockchain system is secure as long as more than 50% of the computational power is controlled by honest users.

[1] Throughout this paper, nodes and users are used interchangably.

2.2 Proof of Elapsed Time

As we have discussed, the proof-of-work scheme requires a node to solve hard problems to limit its speed of generating blocks, which causes a lot of waste in both computational resources and energy. The PoET scheme uses a different approach, as we elaborate below.

Intel's Software Guard Extensions (SGX) technology provides a mechanism to protect selected code and data from disclosure or modification [1]. Based on SGX, Intel proposes Sawtooth Lake, which leverages the idea of "proof-of-elapsed-time" (PoET) to control the construction of new blocks. Using this scheme, each user has to wait for some time before it is allowed to create a block. Such a waiting time needs to follow a probability distribution $\mathcal{F}$ which is determined by the scheme. Briefly, there are two measures utilized by the scheme to make sure that a user has to wait for such a time. First, each user, once generating a block, also needs to generate a proof for the waiting activity with the assistance of SGX hardware, which is submitted together with the block. Second, statistical tests are employed to check whether the waiting times of a user indeed follow a specific probability distribution. We provide details in the following.

Random Waiting Times. As we have described, in the PoET scheme every node has to wait for a time period that follows a distribution $\mathcal{F}$ before generating the next block. In the current Sawtooth Lake method proposed by Intel, this $\mathcal{F}$ can be characterized by a two-stage procedure. At a high level, the procedure works as follows. Each node first uses a formula to generate a number as its temporary waiting time. Such a waiting time can be used to generate multiple blocks until it has to be updated. Specifically, whenever a node has generated a block using the temporary waiting time, it decides at random whether the next block will also be generated using this waiting time, i.e., with certain probability p, it regenerates a new waiting time, otherwise it continues to use the current waiting time. We provide details in the following.

Registration. Every node has to register two things to the system. One is its public/private key pair, which remains unchanged thereafter[2]. The other is a temporary waiting time, which is subject to update. The rule for updating the temporary waiting time is demonstrated by Fig. 1.

Computation of the Waiting Time. Each node uses the following equation to compute its waiting time `wait_time`:

$$\texttt{wait\_time} = \texttt{minimum\_wait} - \texttt{local\_average\_wait} \cdot \log(r) \tag{1}$$

Here $r \in [0, 1]$ is a real number derived from the hash value of the node's previous certificate. If we treat the hash function as a random oracle [4], r is uniformly distributed in $[0, 1]$. `minimum_wait` is a fixed system parameter. To calculate

[2] The SGX component is used to generate a certificate for the public key and send the certificate to the system.

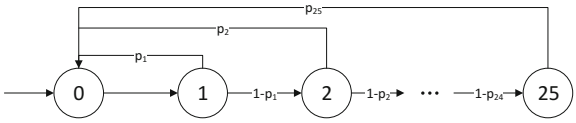

Fig. 1. Each node uses a finite state machine to control the updating process of the waiting time. Each node starts from state 0, where it computes a waiting time. Afterwards, whenever a node goes back to state 0, it updates its waiting time by recomputing a new number. At state $i \in \{1, 2, \cdots, 25\}$, the node first generates a block with the newest waiting time, and goes to state 0 (with probability p_i) or state $i+1$ (with probability $1 - p_i$). The probabilities satisfy the condition that $p_1 < p_2 < \cdots < p_{25} = 1$.

`local_average_wait`, a node checks the most recent `sample_length` (a constant system parameter) blocks to estimate the number of active nodes in the system by checking the waiting time information in these blocks, and multiplies a constant value to get `local_average_wait`. The purpose of `local_average_wait` is to adjust the waiting time according to the number of active nodes. When there are more active nodes, the waiting time will be longer. This design reduces the probability of collisions (i.e., two nodes have the same waiting time and try to create blocks simultaneously) when there are more active nodes.

Block Verification. Whenever a block is generated by a node, it will be verified by other nodes before it is accepted by the system. Straightforward ways of attacks can be excluded by basic verification, e.g., every temporary waiting time can only be used at most 25 times, therefore if a short waiting time is used by a node for 26 times or more, the blocks generated by this node should be rejected. However, a sophisticated attacker, once compromised the SGX, may choose to generate blocks in a sufficiently faster speed but still appears to conform to the scheme (e.g., with constantly updated waiting times). In this case, statistical tests are employed to detect such an attack.

The basic idea is to use z-test to check whether a node is generating blocks too fast (winning too frequently in the competition with other nodes for block creation) [22]. The test assumes that each node has the same winning probability p, and the number of winning times follows binomial distribution $X \sim B(m, p)$, m is the total number of blocks in the system. When m is large enough, it can be approximated by normal distribution $X \sim B(m, p)$, where m is the total number of blocks in the system. When m is also sufficiently large, it can be approximated by normal distribution $X \sim N(mp, \sqrt{mp(1-p)})$, and a z-score can be calculated as $z = \frac{\texttt{win\_num} - mp}{\sqrt{mp(1-p)}}$, where `win_num` is the number of blocks that the node has successfully created. When z is larger than a pre-defined parameter `zmax`, the new block will be rejected. POET provides several candidate values of `zmax` such as 1.645, 2.325, 2.575, and 3.075. This check is conducted multiple times from the latest block to the first on the chain.

Remarks on the Design of Sawtooth Lake. It is relatively easy to understand the intuitions behind the design of Sawtooth Lake Scheme: (i) making the

waiting time longer when there are more active nodes to reduce potential collisions; (ii) using statistical test to detect a potentially compromised node that produces blocks at a higher rate than honest nodes; and (iii) using a random waiting time multiple times to reduce both the generation and verification cost.

However, it is not clear how secure blockchain based system using PoET is, which also depends on the security of the underlying trusted computing platform. Trusted computing hardware is not 100% reliable and assured to thwart any attacks including physical attacks. Indeed, they may be vulnerable to sophisticated adversaries [18,23]. Once compromised, nodes do not need to follow the pre-defined protocol and can take advantage of this to undermine the whole system. Furthermore, Intel's Sawtooth Lake is just one specific implementation of PoET. In general, when compared with other schemes such as proof-of-work, there is a lack of understanding of PoET at protocol and theoretical analysis level. To the best of our knowledge, there is no existing work on analyzing the security of such systems.

3 Preliminaries

We briefly describe the tools that will be used in this paper. We will be using the central limit theorem and apply Berry–Esseen's theorem [3,12] to bound the error of normal approximation:

Theorem 1 (Berry–Esseen's Theorem). *Let $Z_1, Z_2, \cdots, Z_n$ be i.i.d. (independent and identically distributed) random variables with $\mu = \mathbb{E}(Z_1)$, $\sigma^2 = \mathbb{E}[(Z_1-\mu)^2]$, $\rho = \mathbb{E}[|Z_1-\mu|^3]$. There exists a positive constant C such that for $Z = \frac{\sum_{i=1}^{n}(Z_i-\mu)}{\sqrt{n}\sigma}$ and its cumulative distribution function F_n, we have*

$$|F_n(x) - \Phi(x)| \leq \frac{C\rho}{\sigma^3\sqrt{n}}, \quad \forall x \in (-\infty, +\infty)$$

where Φ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0,1)$.

It is shown by Essen [12] that the constant C is upper bounded by 7.59. After a series of improvements over decades, the current best known upper bound for C is 0.4785 [29]. For this paper, it suffices to take $C \leq 1$.

Gordon's Inequality. We use the following inequality by Gordon [17] to bound the tail of the standard normal distribution:

$$\frac{e^{-t^2/2}}{\sqrt{2\pi}} \cdot \frac{1}{t+1/t} \leq \int_t^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \leq \frac{e^{-t^2/2}}{\sqrt{2\pi}} \cdot \frac{1}{t}, \quad \forall t > 0 \tag{2}$$

There are various improved bounds and we refer the reader to a nice technical report [10] which gives a survey. For this paper, the bound by Gordon suffices.

Notations. We summarize most of the notations used in this paper in Table 1.

Table 1. Variables used in the paper.

n	Number of nodes in the system
X_i^j	The i-th random waiting time of node j
X_i	The i-th random waiting time of the n nodes
	(all the random waiting times are ordered arbitarily)
$\mathcal{G}$	The probability distribution of the waiting time
F	Cumulative function of a random variable belonging to $\mathcal{G}$
μ	Mean of the waiting time
Y	A random variable that follows a uniform distribution within $(0, 1)$
ϕ	Fraction of the nodes that are compromised by adversaries

4 Abstract Model of PoET

In this section, we describe the abstract model of PoET based blockchain system.

The system consists of n nodes (users), and each node is equipped with a trusted computing component. Every node keeps generating blocks and adding them to the system. We assume that the time required for generating a block is negligible. However, once a block is generated by a node, it must wait for certain amount of time (which is called the *waiting time*) before it can generate the next block. Nodes with properly working trusted computing component (honest nodes) always determines their waiting times according to a probability distribution $\mathcal{G}$ specified by the protocol of the system. A statistical test is carried out to determine whether a node has generated too many blocks within a certain time period.

Trusted computing components may fail to defend against tampering due to design/implementation bugs and attacks [18,31], and become compromised. We assume that an attacker may compromise multiple nodes, and each compromised node can generate blocks with any waiting time (as long as it passes the statistical test). We define that an attacker compromises the blockchain system if he/she can succeed in generating blocks using compromised nodes such that those generated blocks pass the statistical tests, and in addition, the total number of blocks generated exceeds the total number of blocks generated by the remaining honest nodes by a constant $H > 0$. This means, all the honest nodes keep adding blocks to the main chain while the attacker can keep adding blocks to an attack chain such that even if initially the attack chain is behind the main chain by H blocks, it will eventually take over the main chain.

Throughout this paper, we focus on the following question: To compromise a PoET based blockchain system, what fraction of the nodes does an attacker have to compromise? For the classical proof-of-work based system, the answer is a constant (50%) [25]. However, for PoET system, the answer may vary depending on the following two important components of the system:

– the probability distribution that the waiting time of an honest node should follow; and
– the statistical test that determines whether the waiting times of a node actually follows this distribution or not.

Regarding the Statistical Test. As we have described, the current PoET scheme uses z-test as the statistical test. However, it is arguable whether this is the most suitable statistical test. Therefore, we do not restrict to z-test throughout this paper. Our main result does not rely on the type of statistical test. Indeed, we prove that the attacker can compromise the system by compromising $\Theta(\log \log n / \log n)$ fraction of the nodes even if the statistical test is perfect, that is, even if an attacker is forced to use exactly the distribution specified by the scheme (i.e., he/she will be identified immediately if a different distribution is used to compute the waiting time), the system is still vulnerable compared with a proof-of-work based system.

Regarding the Probability Distribution Specified by the System. In the current design of Sawtooth Lake Scheme, the waiting time X is set to be $X = c_2 + c_1 n \log \frac{1}{Y}$, where $c_1, c_2 \geq 0$ are constant. Here the scheme sets `local_average_wait` to be $c_1 n$, and $Y \in U(0,1)$ is a random variable that follows uniform distribution within the interval $(0,1)$. Consider an arbitrary honest node j and let its waiting times be $X_1^j, X_2^j, \cdots$. In the current design of Sawtooth Lake Scheme, X_i^j's are not independent but are intertwined using a sophisticated approach (See Sect. 2.2). We will first discuss the simpler case where all the X_i^j's are i.i.d. (independent and identically distributed), and then come to the more sophisticated case where X_i^j's follow the distribution implemented in the Sawtooth Lake Scheme.

5 Security Analysis of PoET

In this section, we analyze the security of PoET based blockchain system. Recall that we assume a perfect statistical test, that is, we aim to show that the current PoET based system is vulnerable even if it is equipped with the strongest statistical test. Omitted proofs can be found in the full version of the paper [7].

Under the perfect statistical test assumption, it appears, at first glance that a compromised node cannot gain any advantage over an honest node. However, consider n nodes, each generating blocks with waiting times according to a probability distribution. Given a fixed time interval, it is likely that the fastest node can generate many more blocks than average; and a compromised node can generate as many blocks, pretending to be the fastest honest one. By letting each compromised node simulating the fastest honest node, the attacker may hijack and compromise the system by compromising $\phi < 50\%$ fraction of the nodes. Note that if a compromised node is simulating some honest nodes, then no statistical test can distinguish the waiting times of a compromised node and those of existing honest nodes.

We call this percentage ϕ as the *conservative ratio* and focus on calculating this ratio. We emphasize that this is the percentage of the nodes that the attacker needs to compromise under a perfect statistical test. With a weaker test, it suffices for the attacker to compromise even fewer nodes.

We start with the case where the waiting times of honest nodes are i.i.d. and follow a fixed distribution $\mathcal{G}$. For $X \sim \mathcal{G}$, we denote by F its cumulative distribution function and $\mu = \mathbb{E}[X]$, $\sigma^2 = \mathbb{E}[(X-\mu)^2]$, $\rho = \mathbb{E}[|X-\mu|^3]$. Considering a time interval of length $k\mu$ for a positive number k, how many blocks can n honest nodes generate in total? We have the following lemma.

Lemma 1. *If μ, σ and ρ are all positive constant numbers, then with high probability, n honest node only generate in total $nk + O(\sqrt{nk})$ blocks within a time interval of length $k\mu$.*

According to Lemma 1, on average an honest node only generates $k + O(\sqrt{k/n})$ blocks within a time interval of length $k\mu$, regardless of the distribution $\mathcal{G}$. On the other hand, the fastest node among all the nodes may generate more blocks than the average. The number of blocks that the fastest node can generate depends on the probability distribution $\mathcal{G}$. We estimate this value in the following.

Lemma 2. *With probability $1-e^{-\lambda}$ the fastest node, among n honest nodes, can generate N or more blocks within a time interval of length $k\mu$ if $N \ln F(\frac{k\mu}{N}) \geq \ln \frac{\lambda}{n}$.*

Based on Lemmas 1 and 2, we have the following theorem.

Theorem 2. *Even with perfect statistical test, adversaries may hijack or compromise the system if they compromise $\phi \geq (1+\epsilon) \cdot \frac{k}{k+N}$ fraction of the nodes for positive k and N, where $\epsilon > 0$ is an arbitrary small constant and k, N satisfy that $N \ln F(\frac{k\mu}{N}) \geq \ln \frac{\lambda}{n}$.*

5.1 Discussion on Fixed Probability Distributions

To estimate the value of ϕ, the most important thing is to analyze the value of N that can lead to the inequality $N \ln F(\frac{k\mu}{N}) \geq \ln \frac{\lambda}{n}$ for a positive number k. Note that λ is a constant that measures how likely the distribution that a compromised node simulates should exist within n honest nodes. With $\lambda = 5$, this probability already exceeds 99% (see Lemma 2). For ease of understanding, it suffices to view λ as a small constant like 5. However, our results in this section holds for λ being an arbitrary constant. In the following, we assume that the probability density function of the distribution $\mathcal{G}$ has support (a, b) (i.e., $0 = F(a) < F(b) = 1$) with the mean $\mu = \mathbb{E}[X]$. We further assume that the probability distribution is fixed (i.e., it is independent of the network), therefore a, b and μ are all constants. The goal of this subsection is to prove the following.

Theorem 3. *If the probability density function of $\mathcal{G}$ is independent of the network and has support within the interval (a, b) (i.e., $0 = F(a) < F(b) = 1$), then the adversaries can compromise the system if they compromise $\frac{a_\epsilon}{a_\epsilon+\mu}$ fraction of the nodes, where a_ϵ satisfies that $F(a_\epsilon) = \epsilon$. Furthermore, if it holds additionally that $a = 0$ and $F(x) \geq x^c$ where $x \in (0, \delta)$ for constant c and δ, then the adversaries can compromise the system if they compromise $\Theta(\frac{\log\log n}{\log n})$ fraction of the nodes.*

It is worth mentioning that $\frac{a_\epsilon}{a_\epsilon+\mu}$ is a constant, whereas in general adversaries can compromise the system by compromising a constant fraction of the nodes. However, for some class of distributions, adversaries can compromise the system even by compromising a significantly smaller fraction of the nodes, as is implied by the second half of the theorem.

Proof. Note that μ and a_ϵ are all constant, for $k = O(1)$, $N = \frac{k\mu}{a_\epsilon} = O(1)$ ensures that $F(\frac{k\mu}{N}) = \epsilon$, and also $N \ln \epsilon \geq \ln \frac{\lambda}{n}$ for sufficiently large n. By Theorem 2, adversaries can compromise the system if they compromises $(1+\epsilon)\frac{a_\epsilon}{\mu+a_\epsilon}$ fraction of the nodes for an arbitrarily small constant ϵ. The first half of the theorem is proved.

Now suppose $a = 0$ and there exists a constant $c > 0$ and $\delta \in (0, 1)$ such that for $x \in (0, \delta]$, $F(x) \geq x^c$. We claim that $N \ln F(\frac{k\mu}{N}) \geq \ln \frac{\lambda}{n}$ is satisified with $k = O(1)$ and $N = \Theta(\frac{\log n}{\log\log n})$. To see why, notice that $-\ln F(\frac{k\mu}{N}) \leq c \ln \frac{N}{k\mu} = \Theta(\log\log n)$, therefore $N \log N = \Theta(\log n)$ (indeed, for a sufficiently small positive number c', $N = \frac{c' \log n}{\log\log n}$ ensures that $N \ln N \leq \ln n$). In this case, we have

$$\phi \geq (1+\epsilon) \cdot \frac{k}{k+N} = \Theta(\frac{\log\log n}{\log n}),$$

that is, as long as the adversaries compromise $\Theta(\frac{\log\log n}{\log n})$ fraction of the nodes, the system will be compromised.

Note that $a_\epsilon \to a$ when $\epsilon \to 0$. The following lemma implies that the upper bound of $(1+\epsilon)\frac{a_\epsilon}{\mu+a_\epsilon}$ for ϕ is essentially tight if $a > 0$.

Lemma 3. *If $a > 0$, then the adversaries have to compromise at least $\frac{a}{\mu+a}$ fraction of the nodes.*

So far we focus on probability distributions that are independent of the network. Things become substantially more sophisticated if it is dependent on the network, or X_i's are not independent. For this case, we directly focus on the probability distribution currently implemented in designs similar to Sawtooth Lake Scheme.

5.2 Discussion on the Probability Distribution in Sawtooth Lake

The Simple Setting with the Independent Assumption. We start with the simpler setting, that is, the waiting times of each node $X_1^j, X_2^j, \cdots$ are independent and they follow the same distribution $\mathcal{G}$, where $\mathcal{G}$ is the distribution such

that for $X \sim \mathcal{G}$, $X = c_2 + c_1 L \ln \frac{1}{Y}$, where $c_1, c_2 \geq 0$ are constants, $Y \sim U(0,1)$ and L is a function that depends on n. Specifically, in the current implementation of Sawtooth Lake Scheme, $L = n$. Note that Sawtooth Lake Scheme uses the distribution $\mathcal{G}$ in a more complicated way that the independent assumption on X_i's may not necessarily be true. As the analysis on the simpler case with the independent assumption is crucial for the analysis on the general problem, we start with this simpler case. For the simple setting, we have the following conclusion.

Theorem 4. *Under the independent assumption, if the random waiting time X satisfies that $X = c_2 + c_1 L \ln \frac{1}{Y}$ for $c_1, c_2, L \geq 0$ such that $c_1, c_2 = O(1)$ and $L = \omega(1)$ (i.e., $L \to +\infty$ when $n \to +\infty$), then the adversaries can compromise the system if they compromise $\Theta(\frac{\log \log n}{\log n})$ fraction of the nodes.*

Note that the situation changes significantly when L becomes $O(1)$. Recall that, we have shown in Lemma 3 that to compromise the system, the adversaries have to compromise at least $\frac{a}{a+\mu} = \frac{c_2}{2c_2 + c_1 L} = O(1)$ fraction of the nodes, that means, compromising $\Theta(\frac{\log \log n}{\log n})$ fraction of the nodes is not enough for adversaries when $L = O(1)$.

The General Setting. Now we come to the general setting of the problem, which is exactly how Sawtooth Lake Scheme is implemented. We briefly describe how the scheme works and the reader may refer to Sect. 2.2 for details. Again let $\mathcal{G}$ be the probability distribution such that for $X \sim \mathcal{G}$ we have $X = c_2 + c_1 L \ln \frac{1}{Y}$ where $Y \sim U(0,1)$. Let $0 < p_1 \leq p_2 \leq \cdots \leq p_{25} = 1$ be 25 fixed constants. Every node initializes its state level as 0. If the state level of a node is 0, it generates a random number according to distribution $\mathcal{G}$, sets this number as its waiting time, and updates its state level to 1. If the state level of a node is i where $1 \leq i \leq 25$, it waits for the time length equals its waiting time, generates a block, then with probability p_i it updates its state level to 0, and with probability $1 - p_i$ it updates its state level to $i + 1$.

It is easy to see that the waiting time of a node remains the same if its state level does not go to 0. Therefore, the waiting times $X_1^j, X_2^j, \cdots$ of each node j are not necessarily independent, and consequently we cannot directly apply Theorem 2. We need a new approach to estimate the number of blocks generated by n honest nodes and also the number of blocks generated by the fastest node.

Estimating the Number of Blocks Generated by the Fastest Node. Consider an arbitrary node j. Note that if the state level of node j becomes 0 after it generates the $(i-1)$-st block, then X_i^j is independent of each X_h^j for $h < i$, otherwise $X_i^j = X_{i-1}^j$. Let N be a number to be fixed later. Consider all the subsets $S = \{a_1, a_2, \cdots, a_s\} \subseteq \{1, 2, \cdots, N\}$ such that $a_{h+1} - a_h \leq 25$ for any $1 \leq h \leq s-1$. Let $\mathcal{G}$ be the set of all such subsets.

Now we consider $X_1^j, X_2^j, \cdots, X_N^j$ and let $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_s\} \subseteq \{1, 2, \cdots, N\}$ be the set of all the indices such that the node returns to state 0 before generating the corresponding block, i.e., for any τ_h, we have that $X_{\tau_h}^j$ is

independent of all the X_i^j where $i < \tau_h$, and furthermore, $X_{\tau_h}^j = X_{\tau_h+1}^j = \cdots = X_{\tau_{h+1}-1}^j$. Note that according to the scheme, $\tau_{h+1} - \tau_h \leq 25$. Any fixed $\Gamma \in \mathcal{G}$ denotes a scenario that happens with the probability

$$\pi(\Gamma) = \prod_{h=1}^{s} \prod_{i=1}^{\tau_{h+1}-\tau_h-1} (1 - p_i).$$

Note that $\mathcal{G}$ consists of all the possible scenarios that may happen, and consequently $\sum_{\Gamma \in \mathcal{G}} \pi(\Gamma) = 1$. Now we consider the probability that node j generates N or more blocks within a time length of $k\mu$, that is,

$$P(\sum_{i=1}^{N} X_i^j \leq k\mu) = \sum_{\Gamma \in \mathcal{G}} P(\sum_{i=1}^{N} X_i^j \leq k\mu | \Gamma) \cdot \pi(\Gamma),$$

where $P(\sum_{i=1}^{N} X_i^j \leq k\mu|\Gamma)$ denotes the probability that the event $\sum_{i=1}^{N} X_i^j \leq k\mu$ happens conditioned on the event that the scenario Γ happens. Let $\omega_h(\Gamma) = \sum_{i=\tau_h}^{\tau_{h+1}-1} X_i = (\tau_{h+1} - \tau_h) X_{\tau_h}$, it follows that $X_{\tau_h} \leq \omega_h(\Gamma) \leq 25 X_{\tau_h}$. Therefore,

$$P(\sum_{i=1}^{N} X_i \leq k\mu|\Gamma) = P(\sum_{h=1}^{s} \omega_h(\Gamma) \leq k\mu|\Gamma) \geq P(\sum_{h=1}^{s} 25 X_{\tau_h} \leq k\mu|\Gamma).$$

Further notice that conditioned on Γ, $X_{\tau_h}^j$'s are i.i.d. (each following the same distribution of X) and $s \leq N$. Thus, let Y_i be i.i.d. random variables, each following the same distribution as X, we know that

$$P(\sum_{h=1}^{s} X_{\tau_h}^j \leq k\mu/25|\Gamma) = P(\sum_{h=1}^{s} Y_h \leq k\mu/25) \leq P(\sum_{h=1}^{N} Y_h \leq k\mu/25).$$

$$\begin{aligned} \text{Hence, } P(\sum_{i=1}^{N} X_i^j \leq k\mu) &= \sum_{\Gamma \in \mathcal{G}} P(\sum_{i=1}^{N} X_i^j \leq k\mu|\Gamma) \cdot \pi(\Gamma) \\ &\geq \sum_{\Gamma \in \mathcal{G}} P(\sum_{h=1}^{N} Y_h \leq k\mu/25) \cdot \pi(\Gamma) = P(\sum_{h=1}^{N} Y_h \leq k\mu/25), \end{aligned}$$

Since Y_i's are i.i.d., we are able to apply Lemma 2, that is, if N satisfies that $N \ln F(\frac{k\mu}{25N}) \geq \ln \frac{\lambda}{n}$, then $P(\sum_{i=1}^{N} X_i^j \leq k\mu) \geq P(\sum_{h=1}^{N} Y_h \leq k\mu/25) \geq \lambda/n$, that is, the fastest node can generate N or more blocks with probability $1 - e^{-\lambda}$ within a time length of $k\mu$.

Estimating the Total Number of Blocks Generated by Honest Nodes. Using a similar argument as above, we can prove that with high probability n honest nodes can generate only $M = 25nk + O(\sqrt{nk})$ blocks. See the full version of this paper [7].

Estimating the Compromised Fraction. Now suppose that the adversaries compromise ϕ fraction of the nodes, then they can catch up H blocks if

$$\phi n \cdot N - H \geq 25(1-\phi)nk + \sqrt{(1-\phi)nk}.$$

Applying the same argument as Theorem 2, the adversaries can catch up arbitrary H blocks if $\phi \geq 25(1+\epsilon)\frac{k}{k+N}$ where N satisfies that $N \ln F(\frac{k\mu}{25N}) \geq \ln \frac{\lambda}{n}$. By setting $k = O(1)$, $N = \eta \frac{\ln n}{\ln \ln n}$ and using the same argument as before, it is easy to verify that for $L = \omega(1)$, $N \ln F(\frac{k\mu}{25N}) \geq \ln \frac{\lambda}{n}$ is true for η being a sufficiently small constant, whereas $\phi = \Theta(\frac{\log \log n}{\log n})$. Therefore we have the following conclusion on the security of Sawtooth Lake Scheme.

Theorem 5. *In a PoET system similar to Sawtooth Lake Scheme adversaries can hijack or compromise the whole system if they compromise* $\Theta(\frac{\log \log n}{\log n})$ *fraction of the nodes.*

6 Related Works

In this section, we briefly review related works on the security of blockchain. According to the discussion in Sect. 2.1, the major security requirement is tolerance of malicious users.

As the most popular blockchain construction method, PoW has received intensive study. In the initial paper of Bitcoin, Nakamoto showed that when the majority of users are honest, the probability that an attacker can successfully build his/her own branch decreases exponentially with the depth of the position of the branch [25]. Garay et al. [15] and Pass et al. [26] further verified these security features. Formal frameworks are also developed to study the relationship between performance of a PoW based blockchain and its security level [5,16]. These results, however, cannot be applied to PoET due to its fundamental difference from PoW.

There are also a series of studies focusing on game theory aspects of users involved in mining. From a game theory perspective, Eyal and Sirer [14] showed that even a majority of honest miners is not enough to guarantee the security of the Bitcoin protocol. Sapirshtein et al. [27] and Kiayias et al. [20] studies mining as a game in Bitcoin and analyzes the best strategy of users. Chen et al. further studies the execution of smart contracts as a game [6]. All of these works assume users can determine their behaviour under the restriction of PoW, which is not applicable to PoET. For PoET, a attacker is either forced to follow the protocol (when the trusted computing hardware is not compromised) or able to do whatever he/she wants (when the trusted computing hardware is compromised).

7 Conclusion

Leveraging trusted computing technology for blockchain construction opens a new direction in blockchain design. In this paper, we consider the security of PoET and Intel's implementation Sawtooth Lake. We show that, the

current implementation of Sawtooth Lake Scheme is vulnerable to potential security attacks at protocol level. Indeed, as long as adversaries compromise $\Theta(\frac{\log \log n}{\log n})$ fraction of the participating nodes, they can compromise a PoET based blockchain system by using the compromised nodes to simulate the fastest honest mining nodes in the system. Note that $\Theta(\frac{\log \log n}{\log n})$ is not a constant, which contrasts sharply with the constant threshold of proof-of-work scheme implemented in crypto-currency systems such as Bitcoin and other blockchain based applications. Our results also suggest several possible solutions to overcome this issue.

Changing the Probability Distribution of $\mathcal{F}$. As we show in this paper, if the probability distribution $\mathcal{F}$ does not rely on n, then adversaries have to compromise $\frac{a}{a+\mu}$ fraction of the nodes in order to compromise the system, which increases when a, the minimal value that the random variable can take, is approaching the mean μ. Therefore, the system is more secure if the distribution $\mathcal{F}$ is more concentrated. In the extreme case when the waiting time must be a fixed value, $a = \mu$ and adversaries will have to compromise more than 50% of the nodes in order to compromise the system. It is worth pointing out, the more concentrated the probability distribution is, the more likely a collision will occur. This means that different users may generate blocks at the same time, yielding a branch. We characterize the security issue with respect to $\mathcal{F}$ in this paper. It is an interesting open problem to characterize the collision with respect to $\mathcal{F}$, assessing the trade-off between security and collision.

Allowing Blocks Generated by Honest Mining Nodes to be Rejected. We assume that the statistical test will not reject a block that is generated by an honest node, whereas the adversaries can simulate the fastest honest node in the system. It is possible to get beyond the threshold of $\Theta(\frac{\log \log n}{\log n})$ if we allow the statistical test to reject blocks generated by honest users. As we have shown that among n honest users, fast nodes can generate significantly more blocks than the average. If the statistical test is allowed to reject blocks generated by a certain fraction of the nodes. Specifically, if the statistical test can reject blocks generated by f fraction of the nodes that are fastest for a suitable f, then a constant threshold is likely to exist. Note that using this method, the statistical test should be allowed to reject blocks generated by a certain fraction, instead of a constant number of nodes even if all the nodes are honest. If it is only allowed to reject blocks that are generated by c nodes where c is a constant, then using essentially the same arguments in this paper we can still prove the bound of $\Theta(\frac{\log \log n}{\log n})$.

Acknowledgement. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 113039. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

1. Intel Software Guard Extensions Programming Reference, October 2014. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf
2. ARM: ARM security technology building a secure system using trustzone technology (2009)
3. Berry, A.C.: The accuracy of the gaussian approximation to the sum of independent variates. Trans. Am. Math. Soc. **49**(1), 122–136 (1941)
4. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM (JACM) **51**(4), 557–594 (2004)
5. Chen, L., Xu, L., Shah, N., Diallo, N., Gao, Z., Lu, Y., Shi, W.: Unraveling blockchain based crypto-currency system supporting oblivious transactions: a formalized approach. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 23–28 (2017)
6. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: Decentralized execution of smart contracts: agent model perspective and its implications (2017)
7. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: On security analysis of proof-of-elapsed-time (PoET) (full version) (2017). http://i2c.cs.uh.edu/tiki-download_wiki_attachment.php?attId=70&download=y
8. Courtois, N.T., Emirdag, P., Nagy, D.A.: Could bitcoin transactions be 100x faster? In: 2014 11th International Conference on Security and Cryptography (SECRYPT), pp. 1–6. IEEE (2014)
9. Kaplan, D., Powell, J., Woller, T.: AMD memory encryption. Whitepaper, April 2016
10. Duembgen, L.: Bounding standard Gaussian tail probabilities. arXiv preprint arXiv:1012.2063 (2010)
11. Duong, T., Fan, L., Zhou, H.S.: 2-hop blockchain: combining proof-of-work and proof-of-stake securely (2016)
12. Esseen, C.G.: On the Liapounoff Limit of Error in the Theory of Probability. Almqvist & Wiksell, Stockholm (1942)
13. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-NG: a scalable blockchain protocol. In: 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, pp. 45–59 (2016)
14. Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 436–454. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45472-5_28
15. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_10
16. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16. ACM (2016)
17. Gordon, R.D.: Values of Mills' ratio of area to bounding ordinate and of the normal probability integral for large values of the argument. Ann. Math. Stat. **12**(3), 364–366 (1941)
18. Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: Cache attacks on Intel SGX. In: Proceedings of the 10th European Workshop on Systems Security, p. 2. ACM (2017)
19. Intel: Sawtooth Lake (2017). https://intelledger.github.io/

20. Kiayias, A., Koutsoupias, E., Kyropoulou, M., Tselekounis, Y.: Blockchain mining games. In: Proceedings of the 2016 ACM Conference on Economics and Computation, pp. 365–382. ACM (2016)
21. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. Technical report, Cryptology ePrint Archive, Report 2016/889 (2016). http://eprint.iacr.org/2016/889
22. Lawley, D.: A generalization of Fisher's z test. Biometrika **30**(1/2), 180–187 (1938)
23. Lee, J., Jang, J., Jang, Y., Kwak, N., Choi, Y., Choi, C., Kim, T., Peinado, M., Kang, B.B.: Hacking in darkness: return-oriented programming against secure enclaves. In: USENIX Security (2017)
24. Luu, L., Narayanan, V., Baweja, K., Zheng, C., Gilbert, S., Saxena, P.: SCP: a computationally-scalable byzantine consensus protocol for blockchains. Technical report, Cryptology ePrint Archive, Report 2015/1168 (2015)
25. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
26. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. IACR Cryptol. ePrint Arch. **2016**, 454 (2016)
27. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. arXiv preprint arXiv:1507.06183 (2015)
28. Tapscott, D., Tapscott, A.: Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. Penguin, City of Westminster (2016)
29. Tyurin, I.S.: An improvement of upper estimates of the constants in the Lyapunov theorem. Russ. Math. Surv. **65**(3), 201–202 (2010)
30. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). doi:10.1007/978-3-319-39028-4_9
31. Weichbrodt, N., Kurmus, A., Pietzuch, P., Kapitza, R.: AsyncShock: exploiting synchronisation bugs in intel SGX enclaves. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 440–457. Springer, Cham (2016). doi:10.1007/978-3-319-45744-4_22