mathematics

MDPI

Article

SmartMeasurer: A Secure and Automated Bandwidth Measurement for Tor with Smart Contract

Zejia Tang [1], Tianyao Pan [2], Yang Han [1], Tongzhou Shen [1], Lei Xu [1] and Dawei Xu [1,3,*]

[1] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China; 3120211369@bit.edu.cn (Z.T.); y_han@outlook.com (Y.H.); 3120211366@bit.edu.cn (T.S.); xu.lei@bit.edu.cn (L.X.)
[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China; tianyaopan@bit.edu.cn
[3] College of Cybersecurity, Changchun University, Changchun 130022, China
* Correspondence: xudw@ccu.edu.cn

Abstract: Tor is now using a centralized measurement system called Sbws to measure the bandwidth of relays to guide clients in selecting relays to balance traffic. Sbws has been proven to be vulnerable to multiple attacks because of the centralized structure and exposed measurements. We present SmartMeasurer, a secure and decentralized system for bandwidth measurement. Combining smart contract, Oracle Chainlink and ECC technology, we achieve decentralization while hiding the measurement circuits among the general circuits by exploiting the dual identity of randomly dynamic measurers and guards. We analyze the security of our system and demonstrate that it defends against three types of attacks. Our experiments on both private and public Tor networks show that our system is decentralized while keeping the error and the average of our measurements converges to a small interval of 0.30 Mbps. Compared to other existing methods, our system reduces trust assumptions and the costs of using smart contract, and enhances the practical feasibility of the solution.

Keywords: Tor; load balancing; smart contract; blockchain; decentralization; security

MSC: 68M14

Citation: Tang, Z.; Pan, T.; Han, Y.; Shen, T.; Xu, L.; Xu, D. SmartMeasurer: A Secure and Automated Bandwidth Measurement for Tor with Smart Contract. *Mathematics* **2023**, *11*, 4105. https://doi.org/10.3390/math11194105

Academic Editors: Jialing He, Zhi Fang and Chunhai Li

Received: 28 August 2023
Revised: 25 September 2023
Accepted: 27 September 2023
Published: 28 September 2023

1. Introduction

In an era of increasing internet surveillance, privacy is more important than ever. Tor obscures users' connections by using distributed volunteer relays to forward an encrypted version of users' traffic, and thus, protects users' communication privacy. Currently, Tor is the most popular anonymous communication system on the Internet, with about 6700 geographically diverse volunteer-operated relays transferring nearly 300 Gbps of traffic [1].

With the number of Tor users growing, the increasing traffic leads to a huge burden to the Tor network. To balance the large traffic load, Tor uses a load-balancing system called Simple Bandwidth Scanner (Sbws). It is run by the Bandwidth Authorities (BWAuths) with their own scanners to monitor the performance of every relay and calculate their bandwidth weights. These weights are used to guide users to select relays, and thus, they lighten the network burden.

However, this kind of centralized measurement scheme is prone to security problems. First, because of the vote rule of consensus generation, a powerful adversary would only attack five DirAuths (nine in total and five of them also take the role of a BWAuth) to make the Tor network and its measurement unavailable [2]. If DirAuths go offline, Tor will eventually fall back to an equal weight (uniform random) policy, which will adversely affect client performance [2–4]. Second, measurement results of Sbws are also vulnerable to manipulation [5], which may increase the risk of traffic correlation attacks [6,7]. An

adversary can disrupt the measurer scanning process by using bandwidth DoS attacks because the fixed scanners' IP addresses or destinations [3]. Third, a subtle attack takes advantage of Sbws two-hop measurement circuits which can easily be recognized. A malicious relay can provide service only to measurement circuits while limiting client traffic, thereby giving the authorities the impression of excess capacity without high costs [5,8–10]. In summary, the system is easy to be attacked because of five overt and fixed BWAuths.

Previous work has shown that the BWAuth can be shifted to a blockchain. SmarTor [2] utilizes smart contracts and trusted execution environments to run the measurements and aggregate results. However, SmarTor is facing cost and feasibility problems due to the ETH price continuously going up [11]. In recent years, the improvement of Tor bandwidth measurement has focused on the measurement technology, such as FlashFlow [10] and MLEFlow [12], not an entire system.

The official development team of Sbws is contributing to build an automated, decentralized reputation system that feeds into the directory authorities, providing them with reliable relay performance information [13]. Based on this vision and the work of SmarTor, we present SmartMesurer, a decentralized, automated, and secure system for bandwidth measurement, which supports measurers as well as measurement circuits concealment and resists multiple attacks on bandwidth measurement. To achieve this goal, our work addresses the following challenges: (1) How to design a measurement mechanism to automate the measurement process and reduce costs? The spontaneous operation of measurements is difficult to achieve as a smart contract needs to meet the trigger conditions for execution. In addition, cost is an important consideration in implementing the system due to the significant increase in gas and ETH prices. We need to reduce the frequency of interaction with contract to reduce costs. (2) How to conceal the identity of the measurer on the blockchain to defend against DoS attacks. The openness of the blockchain allows information to be stored in a distributed manner and difficult to tamper with, but also makes all information available and transparent for everyone. This is detrimental for protecting the identity of the measurer, as an adversary can still accurately launch DoS attacks on the working measurers. (3) How to conceal the measurement circuit during the measurement in order to defend against bandwidth inflation attack. With existing measurement methods, an adversary can use the overt measurers and target destinations to identify the measurement circuit, and thus, launch a bandwidth inflation attack. Inaccurate measurement results will affect client experience and even client privacy.

To tackle the challenges, we offer the following contributions:

- We propose a system to securely and decentrally measure the relay bandwidth of the Tor network, using a smart contract and Oracle Chainlink to distribute and automate measurement process. Our measurement mechanism is a smart contract that actively directs the measurement process, reducing the frequency of transaction between the measurer and the contract and costs.
- We use Elliptic Curve Cryptography to encrypt the measurer information on the blockchain and use Chainlink's function to make the measurer dynamically and randomly change to hide the identity of the measurer, making it difficult for an adversary to launch DoS attacks on working measurers.
- We set the guard to be the volunteer measurer and improve the measurement mechanism based on Sbws. In combination with randomly selected measurers, we make the measurement circuit perform similar to the normal client circuit as a defense against bandwidth inflation attack, thus improving the reliability of the measurement results.
- We provide a formal analysis of SmartMeasurer to demonstrate its security properties. We deploy the contract on Ether Gerilo and run our scheme on private and public Tor network. We compare its measurement results with Sbws and its cost with SmarTor to prove its feasibility and improvement.

2. Related Work

There are several articles that have proposed improvements to the Sbws, but their solutions have some limitations.

SmarTor employs Smart Contract and Trusted Execution Environments (TEEs) to shift the operation of the BWAuth to achieve decentralization. While the security of blockchain and smart contract is reliable, the costs associated with implementing a blockchain-based Smartor need to be considered. TEEs rely on trusted hardware to ensure data security, but the cost and feasibility problems (such as whether there are enough qualified volunteers) also need to be taken into account. On the other hand, the security of TEEs is also debatable. Although attacking secure hardware requires a huge cost, if the attack can bring sufficient benefits, TEEs still face the risk of being targeted. In 2020, computing cybersecurity researchers found that two different attacks [14], i.e., SGAxe [15] and CrossTalk [16], are exploited against Intel CPUs to obtain sensitive information in the TEE. In terms of implementation, Smartor replaces DirAuth with a smart contract, but the full functionality of DirAuth is not realized, which may lead to system unavailability. The practical application of Smartor remains challenging.

In PeerFlow [9], relays periodically report to DirAuth the total number of bytes that they exchange with each other. DirAuth then securely aggregate the traffic data to produce relay weights. In the process of determining weights, PeerFlow produces lower bounds on relay capacities that can be used as capacity estimates.

FlashFlow [10] is a new proposal to replace TorFlow. FlashFlow is a secure and decentralized speed test for the Tor network. The purpose of FlashFlow is to measure Tor capacity, which uses several servers that measure a relay simultaneously, generating a large network load and using cryptographic techniques to ensure max out its capacity. FlashFlow has a guaranteed inflation bound of only 33%, but it is based on the assumption that a relay capacity is based on a hard limit that cannot be exceeded. FlashFlow intends to obtain a sustainable capacity estimate. In practice, however, it is often easier and cheaper to obtain high peak bandwidth capability than sustaining the same bandwidth continuously. Furthermore, FlashFlow also faces the threat of a bandwidth inflation attack.

3. System Overview

The improvement over existing work in this paper is that SmartMeasurer uses random numbers, cryptography technology, and dual identity to hide the identity of the measurer and the measurement circuit to resist DoS attacks and bandwidth inflation attacks. In addition, our improvements to the measurement mechanism reduce trust assumptions and the costs of using smart contract, and enhance its practical feasibility. In this section, we first introduce our system model, and then specify our trust assumptions and threat model.

3.1. System Model

The system model is illustrated in Figure 1. It includes the fundamental components of Tor: Relays, Clients, Directory Authority (DirAuth), Destination, and Web Server. In SmartMeasurer, the function of Bandwidth Authorities is distributed between two types of entities: The Smart Contract (SC) and Measurer. The SC is a computing program stored on a blockchain while Measurers are acted by guards which are highly trusted by DirAuth. The Web Server is a predefined server used to obtain measurement files.

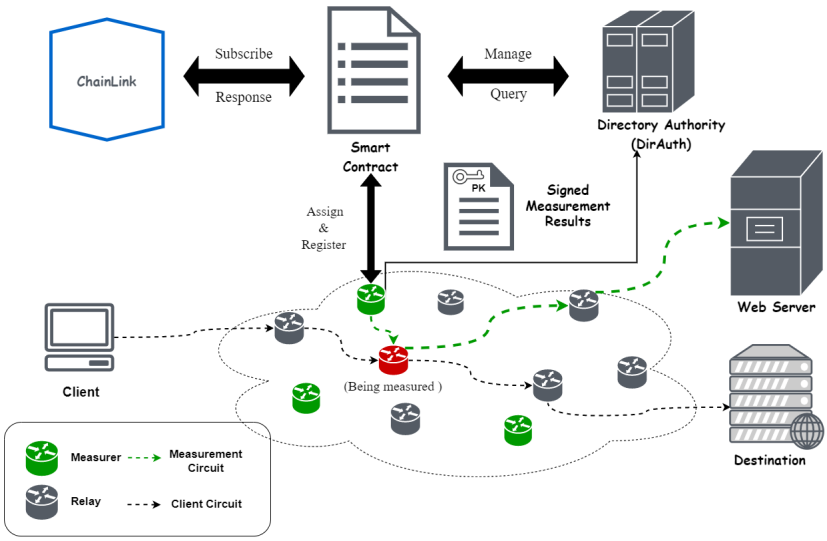

Figure 1. System Overview of SmartMeasurer. The red relay is being measured.

3.2. Threat Model

According to Section 1, we assume that the adversary aims to affect the proper performance of the bandwidth measurement, such as launching a Denial of Service (DoS) attack or a bandwidth inflation attack. For the specific attack, we assumed three types of adversaries.

- Adversary 1: Malicious measurers that tamper with the measurement results. In order to increase the bandwidth weight of their controlled relays or decrease other honest relays' weight, and thus, increase the probability of controlled relays to be selected, malicious measurers may tamper with the measurement results.
- Adversary 2: Malicious relays that launch a bandwidth inflation attack. Malicious relays can suppress client traffic to increase the proportion of traffic it can be observed when detecting measurements and then increase their bandwidth weights to improve chances of being selected.
- Adversary 3: Attackers that launch a DoS attack. Such attackers may launch two types of DoS attacks to disrupt the measurement process, skewing the accuracy of relay weight. One is to register fake measurers (real or virtual) to make the smart contract random selection unavailable, and the other is to directly launch denial-of-service (DoS) attacks on honest measurers to delay or lose their measurement packets.

3.3. Security Assumptions

Security assumptions about Relays and Clients are inherited from the current Tor system—Clients are considered as trusted, while some of the relays can be malicious. Smart contract is usually no trust since it is open and transparent, and anyone can view every line of code (rules). We assume that no attacker can launch a 51% attack on the Ethereum. In addition, we assume the Bandwidth Measurers are believable. Although we set the guards trusted by the authorities as measurers, some of malicious relays still have the opportunities to be guards—malicious relays can perform similar to excellent relays in the beginning until they become guards [17], which will make the a few measurement results incredible. However, because of the high cost to be a guard, we assume that the number of malicious measurers are limited.

Design Goals: SmartMeasurer seeks to provide an automated and decentralized scheme using a public blockchain and a smart contract, which satisfies the following properties.

- Automation: The bandwidth measurement should be processed spontaneously.
- Decentralization: The measurers should be numerous and distributed.
- Security: Strong security should be provided for measurement to limit the amount of bandwidth cheat from malicious relays and for measurer to prevent from DoS attack.

- Measurement accuracy: The measurement results should be reliable and allow clients to estimate relay capacity with high accuracy.
- Unobservability: The identity of measurer and property of measurement circuit should not be easily observed by other relays.

4. Preliminaries

In this section, we will provide a brief overview of bandwidth measurement in Tor and the background of the technologies we will use.

4.1. Tor Bandwidth Measurement

Tor currently uses Sbws [18] as its measurement tool. Sbws was implemented in Tor in 2018. Sbws is an updated version of TorFlow [13] but has the same disadvantages. Thus, we will introduce TorFlow as the main technology and complement the Sbws adjustment.

TorFlow calculates the weights using the self-reported bandwidth of the relays. Each relay estimates its own network capacity by computing the maximum bandwidth sustained output over any 10-s period in the last 5 days and reports this value to directory authorities, which is published every 18 h in a server descriptor. Directory authorities then compile it across all relays and distribute the information to the clients in a consensus document. The Tor directory specification [19] calls this value the observed bandwidth. However, we call it the self-reported bandwidth here to emphasize its origin.

A complete TorFlow bandwidth scanner consists of a scanner and a generator. The scanner works by creating a two-hop measurement circuit with the relay. If the relay is an exit, then we obtain a non-exit relay as a helper; if the relay is not an exit, we use it as a first hop. It then measures the speed by downloading data from a preset Web Server and stores the measurements. The generator reads the measurement results, aggregates, filters, and scales them using TorFlow's scaling method. It then generates a V3bw file, which is read by the directory authorities to report the bandwidth of the relays in its vote [18].

Sbws adjust the way weights are calculated: Sbws uses the minimum of the self-reported bandwidth and the consensus weight: $C^S_{t+1}[j] = \min\left(C^S_t[j], b_t[j]\right) m_t[j]/\mu_t$, where $m_t[j]$ denotes measured bandwidth, μ_t denotes the average measured bandwidth of all relays, and $b_t[j]$ denotes the self-reported bandwidth [12]. Sbws has also made some minor changes to the collection of measurements [20].

This way of the bandwidth weight calculation has been used in Tor since 2012, and was first implemented in TorFlow [12]. These two methods, however, have similar disadvantages. Currently, the five BWAuths run the Sbws measurement tool by their scanners, whose IP addresses are overt and can be detected [3]. An adversary can disrupt the measurer scanning process, such as increasing latency and packet loss. By launching such a bandwidth DoS attack, the adversary can effectively reduce the accuracy of the relays' bandwidth weights, which may break the load balances.

A subtle attack takes advantage of the two-hop measurement circuits, which can easily be recognized. Sbws's active measurements should occur simultaneously with client traffic, but a malicious relay can detect its measurement due to the fixed address of BWAuths' scanners or destinations and restrict client traffic to increase the proportion of traffic it can be measured, thus increasing its ability to deanonymize Tor users through a traffic correlation attack. Thill has demonstrated this attack in 2014 [8], which we call the bandwidth inflation attack.

4.2. Blockchain and Smart Contract

A Blockchain [21] is essentially a decentralized database, a new application of computer technology such as distributed data storage, peer-to-peer transmission, the consensus algorithm, and the encryption algorithm. Information is divided into blocks consisting of data records and headers chained together as each header includes the hash of the previous block header, preventing changes in blocks. To tamper with information stored in the blockchain, an attacker has to control more than 50% of the CPU power, which will cost a

large number of resources. For example, to attack Ethereum [22], an attacker would require the power output of a nuclear power plant. Furthermore, it is much harder to launch a DoS attack on blockchains due to the peer-to-peer network structure [23], such as eclipse attacks [24], routing attacks apostolaki2017hijacking, and mining-based DoS attacks [25]. Therefore, it is possible to store access information of Tor relays with a smart contract, which ensures that only the owner may modify it.

Ethereum [26], described as a second-generation blockchain, is the second largest blockchain platform after Bitcoin, with a current market capitalization of over 250 billion USD (April, 2023). Ethereum has become one of the most popular blockchains in the world due to its high scalability and speed. A more significant reason for its popularity is that it supports smart contract programs that run automatically when predetermined conditions are met.

Solidity is a high-level programming language and typically uses for smart contract writing. Every node in the Ethereum network runs the code translated from a smart contract under a special environment named the Ethereum Virtual Machine (EVM). Ethereum defines a notion of gas which is to be paid with cryptocurrency and needs to be spent in order for the contract to be executed. Executing code, however, require a high computational cost, since it has to be done by everyone verifying the corresponding block [2]. As a result, gas cost is of great consideration when using smart contract.

4.3. Chainlink

Chainlink [27] is a decentralized oracle network that enables smart contracts to securely access off-chain data feeds, APIs, and other real-world resources. It was founded in 2017 by Sergey Nazarov and Steve Ellis and is built on top of the Ethereum blockchain. Chainlink is designed to solve the "oracle problem" in smart contracts, which is the challenge of how to obtain and verify real-world data in an untrusted and decentralized manner. By providing a reliable and decentralized solution for obtaining off-chain data, Chainlink has become a popular tool for powering decentralized finance (DeFi) applications and other blockchain-based services. Chainlink's native token, LINK, is used to pay for the services provided by the network's oracles.

Chainlink Automation: Chainlink Automation uses a decentralized network to monitor clients' automation logic securely and cost-efficiently off-chain, and then initiate an on-chain transaction to execute the smart contract function when predefined conditions are met. Chainlink Automation will reliably execute smart contract functions using two types of triggers. One is a time-based trigger, which will execute functions according to a time schedule. The other is a custom logic trigger, which use custom logic to allow Chainlink Automation to determine when to execute your smart contract functions.

Chainlink VRF: Chainlink Verifiable Random Function (VRF) is a proven fair and verifiable random number generator (RNG) that enables smart contracts to access random values without compromising security or usability. For each request, Chainlink VRF generates one or more random values and cryptographic proof of how those values were determined. The proof is published and verified on-chain before any consuming applications can use it. This process ensures that results cannot be tampered with or manipulated by any single entity, including oracle operators, miners, users, or smart contract developers.

Chainlink has a wide range of application cases and already has practical applications in many scenarios. Chainlink is currently used mainly in DeFi (Decentralized Finance) projects and applications to feed data and market prices for their contracts [28], such as the cryptocurrency loan platform Liquity and Hedge. There are also articles proposing to combine Oracle and Smart Contract, as in the work done by Gonçalve in [29]. As time progresses, more and more platforms are using other products of Chainlink. In blockchain games, Aavegotchi and Polychain Monsters use Chainlink VRF to provide random numbers to determine the winner [30]. Chainlink Automation is also used in COTI, RoboVault to automate their contracts to complete their products.

4.4. Elliptic Curve Integrate Encrypt Scheme

Elliptic Curve Integrate Encrypt Scheme (ECIES) [31] is a hybrid encryption scheme based on Elliptic Curve Cryptography (ECC). ECC is a 32-byte system, which is much shorter than RSA but provides more comparable security and higher processing speed [32]. It is often used to exchange a shared key and sign with a private key, but can also be used to encrypt data. Another reason for choosing ECIES is that a measurer has to upload its public key in the registration transaction. The shorter the public key is, the lower the costs for the measurer. We describe the operation of the ECIES encryption algorithm as follows, where E is the elliptic curve over the prime field p with G as a base point of order n. We describe the operation of the encryption algorithm in Algorithm 1.

Algorithm 1: The encryption algorithm

Input: ECC parameters(E, p, G, n), msg
Output: C_1, C_2
1 KeyGen: Based on the ECC parameters, choose a private key k and calculate public key $K = kG$;
2 EncMsg: Based on the E, G, K, a sender encodes msg to a point M and get a random number r;
3 Calculate: The sender calculates $C_1 = M + rK$, $C_2 = rG$, and sends them to the receiver;

5. Detailed Overview

We propose a decentralized scheme for measuring the bandwidth of Tor relays called SmartMeasurer. SmartMeasurer aims to decentralize the bandwidth measurement tasks of BWAuth using a smart contract and ensures that DirAuth obtain reliable relay bandwidth information from measurers for aggregation and vote to generate a new consensus.

The specific process is shown in Figure 2. A measurer needs to register on SC before joining measurement. During a measurement round, the SC randomly selects measurers from the Measurer_list using a set of random numbers generated by Chainlink, and the measurers call the query function of the smart contract to obtain the permission to participate and start measuring relays. Finally, the measurers send the measurement results to DirAuths, who collect the measurements and aggregate them to vote on a new consensus.

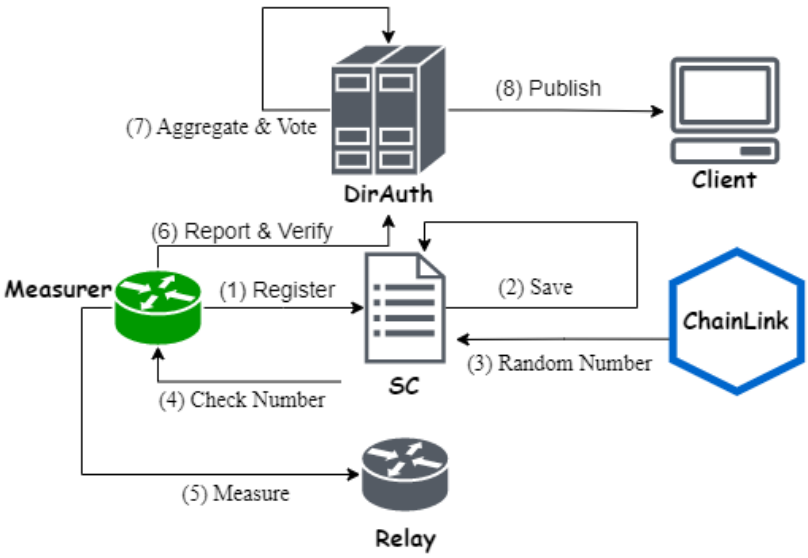

Figure 2. Work Process of SmartMeasurer.

One challenge is that the function of smart contract is limited, especially when it comes to implement the BWAuth verification of the self-reported bandwidth, as it requires

synchronous communication across the network, accurate timer, and the ability to protect cryptographic keys by the contract [2]. Additionally, the cost of using smart contract needs to be considered. Therefore, replacing DirAuth is a significant challenge. To improve the feasibility of our approach, unlike SmarTor, we do not replace DirAuth. Instead, we separate its bandwidth measurement task without affecting the other functions of DirAuth.

Due to the conditional trigger nature of smart contract and the pseudo-random nature of the blockchain parameter, it is difficult to get smart contract to actively direct the measurement process and generate reliable random numbers. Fortunately, the Oracle Chainlink helps smart contract solve this problem. Chainlink Automation will execute the specified function according to the measurement interval set by the system administrator. The function aims to trigger the Chainlink VRF to generate a set of random numbers which used to pick the measurers in the list. The smart contract and Chainlink are at the core of the scheme. The role of a smart contract in SmartMeasurer is a database and a Chainlink VRF subscriber while Chainlink acts as a decentralized entity. Only if they work together can a measurement process be initiated.

In terms of bandwidth measurement technique, the innovations of SmartMeasurer are that we choose guards trusted by the Tor authority to serve as measurers instead of untrusted third parties, such as ISPs or internet hosting providers. According to Section 4.1, TorFlow uses traffic that is explicitly labeled for bandwidth probing due to the special public fixed address of BWAuths' scanners. Malicious relays can easily identify measurement circuits and preferentially forward probe traffic to inflate bandwidth estimate [33]. The guard can be considered the most trustworthy relays, as they protect clients' anonymity. To become a guard, a relay must have sufficient bandwidth, stable online time, and long enough working time [34]. This scheme can theoretically defend against a way of bandwidth inflation attack because a measurement circuit established by a guard appears like a client circuit, thus hiding its circuit property. We will discuss this point in more detail in Section 6.2.

For information security, the measurer uses ECIES to digitally sign the measurements and encrypt the relay's Tor fingerprint. The anonymity of the measurer identity is one of the keys for our scheme. One problem is that the openness of the blockchain makes it possible for anyone to obtain the data on the blockchain. However, our goal is for DirAuth to have access to all measurer information on the blockchain while others do not. Another is that DirAuth needs to authenticate the measurers who send the results according to the information in the Measurer_list to resist fake measurers. In order to protect the identity of the measurers while enabling DirAuth to verify the identities of the measurers, we design to encrypt the relay fingerprint using ECIES encryption; for example, the measurer fingerprint is encrypted using the public key of DirAuth, and only DirAuth can decrypt the measurer information and verify it with the latest consensus. In addition, to prevent man-in-the-middle attack, we need the measurers to digitally sign using its own private key.

5.1. Measurer Registration

Algorithm 2 presents the pseudocode of measurer registration process. In order to complete our measurement work, measurers must register in the Measurer_list. We require DirAuth to add a Measurer Flag in future consensus, which will be issued by DirAuth to guards who meet the requirements and want to be measurers. A measurer sends a registration transaction signed by their wallet key to the SC, including the measurer's public key and the Tor fingerprint encrypted with the public key of DirAuth. DirAuth searches for the measurer information based on the latest consensus when it receives its measurement result and verifies its identity (check the Measurer Flag and IP). If the verification passes, the smart contract receives and saves the data in the transaction.

It is worth noting that the key of our scheme is the implementation of a Measurer_list that is only accessible to the administrator (DirAuth), since knowing the identities of the measurers would be meaningless for hiding the measurements. An adversary could attack the measurers based on random numbers and the list on the blockchain. Unfortunately,

data stored on smart contract are not private, which has been demonstrated in many cases and has caused huge financial losses [35,36]. Therefore, we need to encrypt the Tor fingerprint of the measurer with DirAuth's public key. With the anonymity of the Ethereum address, the data on the blockchain will not expose any information about the identity of the measurer. Each measurer can call the function knowing only its serial number in the list but not the others. Even if an adversary obtains the complete list from the blockchain, it will not be able to correlate it to any measurer.

Algorithm 2: Measurer Registration

Input: $pk_{DirAu}, fp, sk_{wallet}$
Output: txn_{rcpt}
Function `EncryptFingerprint`(pk_{DirAu}, fp)**:**
  $fp_{enc} \leftarrow$ PKE.encrypt(pk_{DirAu}, fp);
Function `RegisterTransaction`$(pk_{DirAu}, fp_{enc}, sk_{wallet})$**:**
  $(pk_{mea}, sk_{mea}) \leftarrow$ PKE.generateKey();
  $msg \leftarrow pk_{mea}, fp_{enc}$;
  $txn \leftarrow$ buildTransaction$(addr, msg)$;
  $txn_{sgnd} \leftarrow$ signTransaction(txn, sk_{wallet});
  $txn_{hash} \leftarrow$ sendTransaction(txn_{sgnd});
  $txn_{rcpt} \leftarrow$ waitForTransactionReceipt(txn_{hash});
  return txn_{rcpt};

Smart contracts do not have the ability to verify identity, which could lead to attacks by malicious adversaries, which we will discuss in Section 6.1. This would require the contract to have an administrator to maintain the Measurer_list. Only the administrator can add or remove measurers in the list. We expect DirAuth to act as the administrator.

5.2. Measurement Preparation

Two of Chainlink's products, Chainlink VRF and Chainlink Automation, are used in this scheme and the process are described in Figure 3.

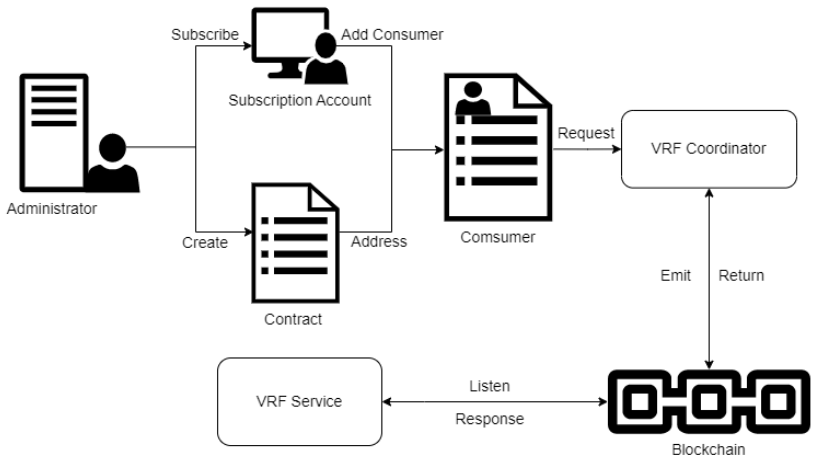

Figure 3. The process of using Chainlink VRF. An administrator should take these steps to subscribe random numbers from Chainlink VRF.

Randnum Request Contract Deployment: In order to subscribe Chainlink VRF to generate random numbers, the system administrator must register a subscription account and obtain an account ID. Then, the administrator needs to create a consumer contract that must inherit the contract VRFConsumerBaseV2 (from the official) and fill in the parameters related to the request for random numbers (e.g., number of requests, blockchain ID, etc.). Once the contract is created, the administrator deploys it on the blockchain that supports

smart contract and add the address of the consumer (smart contract) to the subscription account. When the administrator requests a random number, the VRF coordinator emits the request to the blockchain. The VRF Service listens to the blockchain and response after a period of time. The VRF coordinator fetches the random number from the blockchain and returns it to the contract.

Automation Subscription: In order to subscribe Chainlink Automation, the system administrator must register a new upkeep on the Automation network. During the registration, the administrator should select time-based trigger, provide the contract address, and select the target function. Once configured, Chainlink Automation will trigger the contract function at regular intervals as configured by the administrator.

Measurer Assignment: We delegate Chainlink to assist the SC in directing the measurement process. ChainLink Automation can reliably execute a SC function using a time-based trigger, while ChainLink Verifiable Random Function (VRF) can provide verifiable random numbers. Thus, using these two functions, a set of random numbers can be automatically generated in a measurement round. Chainlink Automation triggers the SC to request a random number from ChainLink VRF at every measurement round, generating a set of random numbers used to select the measurers participating in the measurement round from the list.

Permission Verification: Measurers should query the blockchain (call SC function) at the beginning of each measurement round to obtain the random numbers and compare with theirs. Once a guard is successfully verified, it will prepare for the measurement process.

5.3. Bandwidth Measurement

Building Circuits: Similar to the current approach, SmartMeasurer still uses a two-hop circuit. The difference is that our scheme fixes the relay to be measured at the first hop and searches for a suitable exit in the exit set as the second hop to achieve better concealment.

Helper Selection: To avoid relays at both ends of the circuit affecting the measurement results, the measurer and exit should be faster than the middle relay. Since the threshold of guard is having a bandwidth higher than the median bandwidth or 2 Mbps [19], the bandwidth resources of the measurers are sufficient. When selecting the second hop, considering the exit policy, an exit with a bandwidth higher than the middle relay should be sought within the exit set as a helper (using the bandwidth value in the latest consensus as the reference). If no such helper is available, the fastest exit among them should be found to minimize the impact.

Measurement Process: Each measurer will sort the relays according to the bandwidth information in the consensus. To avoid its own bandwidth limits the measurement results, a measurer selects the relays with lower bandwidth than its own to measure. The measurer and the relay to be measured will establish a two-hop circuit to a pre-determined Web Server through Tor, download a file, and measure the download speed multiple times to calculate the average value and median.

5.4. Reporting and Aggregating Results

To prepare for verification, DirAuth calls the SC function to obtain the corresponding measurer information in the list based on the random numbers of the current round, decrypts the encrypted fingerprints with its private key, and searches for full information in the latest consensus referring to unique fingerprint of every relay. After the measurement is completed, a measurer digitally signs the result with its private key and sends it to DirAuth, which temporarily stores it and logs the IP. Then, DirAuth checks the IP and Measurer Flag against the latest consensus. If the identity of the measurer is authenticated, then DirAuth uses the public key of the measurer (storing in the list) to verify the signature. After all verifications are passed, its measurement result is received. DirAuth will repeat the operation after enough results are collected and aggregated. Once the measurements have been aggregated, DirAuth initiates a vote to generate a new consensus. If the number of received results is less than half of the participating

measurers, the results of the current round are discarded and the previous consensus is used continuously. An overview of the verification process is shown in Algorithm 3.

Algorithm 3: Verification Process of DirAuth.

Input: $sk_{DirAu}, fp_{enc}, Dir\_consensus$
Output: Receive/Discard
Function `Decryptfp`(sk_{DirAu}, fp_{enc})**:**
  $fp \leftarrow$ PKE.decrypt(sk_{DirAu}, fp_{enc});
  return fp;
Function `GetRelayinfo`$(fp, Dir\_consensus)$**:**
  $relay\_info \leftarrow$ getRelay$(Dir_consensus, fp)$;
  return $relay\_info$;
Function `Validate`$(V3bw_{sgnd}, ip, relay\_info)$**:**
  $tmp\_V3bw \leftarrow V3bw_{sgnd}$;
  $tmp\_ip \leftarrow ip$;
  if *relay_info.ip == tmp_ip* **then**
    if *relay_info.flag == measurer* **then**
      if $V3bw_{sgnd}$*.signer == relay* **then**
        Receive tmp_V3bw;
  else
    Discard tmp_V3bw;
  end

5.5. Feedback Mechanism

If something goes wrong in our system, DirAuth will receive no measurement results sent by measurers during a round. The administrator can check the transaction records in Etherscan to find the problem. A more detailed feedback mechanism platform will be designed in the future, allowing the administrator to visualize system errors rather than searching by themselves.

6. Security Analysis

The security goal of the SmartMeasurer is to prevent attackers from interfering with measurements. In this section, we discuss three kinds of attack and give the security analysis.

For specific attacks, we apply a mathematical model to demonstrate the low success rate of malicious measurers trying to influence relays' bandwidth weights of the consensus. Second, we perform a formal analysis to explain why SmartMeasurer can defend bandwidth inflation attack. Third, we give a formal definition of DoS attack and demonstrate SmartMeasurer can defend this attack.

6.1. Attack from Malicious Measurers

In TorFlow or Sbws, such attacks do not exist due to the fixed, official measurers (BWAuths). However, in the decentralized measurement, the presence of malicious measurers situation is inevitable. We assume that the attacker manipulates several relays and register measurers with the smart contract in an attempt to increase the probability of participating in the measurement. A malicious measurer participating in a measurement can tamper with the measurement results of some relays, to increase the bandwidth weights of its own relays or decrease the bandwidth weights of honest relays. Whether or not the Measurer Flag is obtained does not matter to the malicious measurer. They can even be 'ghost' (virtual) relays, since the smart contract does not verify the identities of measurers. We will discuss 'ghost' relays in Section 6.3. A malicious measurer can send a registration transaction to the smart contract and may be selected by a random number in future measurement round.

Thus, Suppose G is an event: the malicious measurers are randomly assigned to the same round and occupy more than half involved in the round. Thus, we have probability $P(G) \in [0,1] \subset R$. Let $p \in [0,1] \subset R$ denote the proportion of malicious measurers to the all. The randomness of Chainlink VRF can be trusted, so that each measurer has the same probability of being selected in a round, so $P(G)$ obeys a binomial distribution, that is:

$$P(G) = \sum_{i \geq n/2} C_n^i p^i q^{n-i},$$

where $q = 1 - p$, n denotes the number of measurers participating in each round.

We simulate the probability of event G occurring for different p and n. Figure 4 illustrates that when $p < 0.5$, $P(G)$ decreases and converges to 0 as n increases; when $p > 0.5$, $P(G)$ decreases and converges to 1 as n increases. Moreover, $p \geq 0.5$ means that the attackers control more than half of the measurers; at this time, the attacker can manipulate the measured bandwidth arbitrarily. However, in this situation, the Tor network will no longer be useful because most of the guards are under control. Thus, $P(G)$ is only discussed when $p < 0.5$.

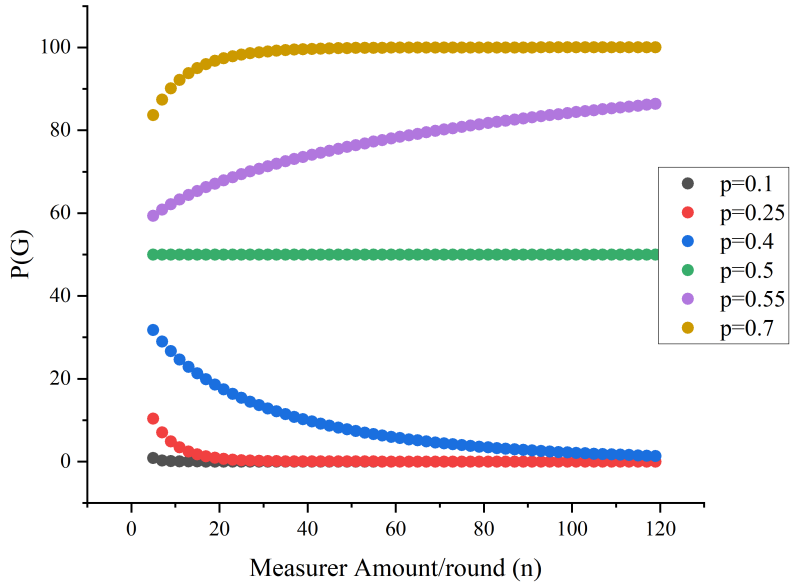

Figure 4. Probability $P(G)$ with increasing measurer amount per round under different proportion of malicious measurers p.

When n is equivalent, $P(G)$ increases as p increases; when p is equivalent, the probability of such an attack occurring decreases and converges to 0 as n increases. Assuming a measurement round with 5 scanners as in the current system, the $P(G)$ is 10.35% for $p = 0.25$ and 31.74% for $p = 0.4$. However, for $p = 0.25$, the probability can be reduced to less than 1% by expanding n to 19. In practice, controlling 10% of the measurers requires a great cost, because it is 'expensive' to become a guard (maintain a high uptime for weeks and have good bandwidth) and then obtain the Measurer Flag. When $n = 5$, the probability of the attack occurring is only 0.86%. An attacker who wants to influence the consensus would have to control more than half of the measurers in a measurement round. We find that there are 3643 guards in the Tor network from the 2023-04-14 consensus file, which is roughly half the number of all relays. With a huge set of guards, an attacker trying to control even $0.1 * 3643 = 364$ guards would have to pay a considerable cost. In other words, if an attacker controls no more than $n/2$ malicious measurers in a measurement round, it will have no effect on the consensus.

In addition, we discuss a type of attacker that disguises a measurer. Such an attacker may not have the Measurer Flag or even not be the relay in Tor. They simply upload the public fingerprint from the consensus and a casual public key to join the Measurer_list

disguising as a particular measurer. If it is chosen in a round, as described in Section 5.4, DirAuth will verify the identities of result senders, and then determine whether to receive the result or not. Administrators can see these dishonest measurers' records and remove their registration information from the Measurer_list. Furthermore, the attacker has to pay a gas fee for registration, which is a great cost.

6.2. Bandwidth Inflation Attack

In TorFlow, Sbws, and FlashFlow, there is a bandwidth inflation attack in which a malicious relay restricts client traffic by identifying the measurement circuit to increase the measurement speed, and thus, its measured bandwidth value. In Section 4, we mention that the BWAuth runs Sbws to measure the relays in Tor. Due to the fixed scanner and destination address, the measurement circuits are easily detected, and the relay can restrict client traffic to boost the measured bandwidth value. Suppose an attacker takes control of some Tor relays and wants to deliberately increase their bandwidth weight to improve their probability of being selected by clients, and thus, achieve some goals, such as deanonymization attacks on clients.

Intuitively, an attacker may detect the relay traffic characteristics and identify the property of the circuit. Suppose a measurer build a measurement circuit to a malicious relay. We place the relay to be measured at the first hop of the measurement circuit. For the malicious relay, it will detect that a guard forwards traffic to it, but will not be able to distinguish whether the circuit is established by a client or by a measurer. According to the Tor protocol, it can only forward traffic as a middle relay without performing special operations on the circuit.

An attacker may try to decrypt the Measurer_list to obtain the measurer information. The Measurer_list stored on the smart contract can be obtained by anyone in Ethereum. An attacker may obtain and decrypt the list, and combines with the random numbers publicly available on the blockchain to acquire the measurers participating in each round. Then, it launches a bandwidth inflation attack. However, during the registration, the measurers encrypt their fingerprints with DirAuth public key. Furthermore, its public key and Ethereum address do not include any information about identity. Unless the private key of DirAuth is exposed, it is difficult for an attacker to decrypt.

We discuss here a more powerful attacker. The attacker can listen to communication of DirAuth and track the measurers who send measurement results to it, aiming to obtain the identities of the measurers in a round. Theoretically, it is possible to obtain the Measurer_list with continuous listening. This attack is easy to perform when measurers are sparse, which only takes the attacker a few rounds' listening. Even if it is hard to correspond to the serial numbers, the traffic sent by these limited number of measurers (guards) can be forwarded to the maximum extent possible, regardless of whether they are measuring or not. However, in Tor, there are enough guards to comprise a large set of measurers. In Figure 5, we logarithmize the number of combinations of measurers in order to visualize the data. The exponential increase in the number of combinations as N and n increase. The difficulty of launching such an attack also increases significantly. Variable measurers make the attacker difficult to confirm the measurers' serial numbers. As long as it does not know the serial numbers, it could not confirm these measurers whether to be chosen in a given round.

6.3. DoS Attack

The aim of the DoS attack is to make it impossible for the selected measurers to carry out measurement work thus having an impact on the generation of consensus bandwidth. We will discuss two types of DoS attacks. Specifically, one is launched against a smart contract due to information flooding, where an attacker may perform a DoS attack on the system by registering non-existent 'ghost' measurers. Another is the DoS attack launched against the measurers' scanners [3]. For these two different types of attack causing the same result, we now give its definition.

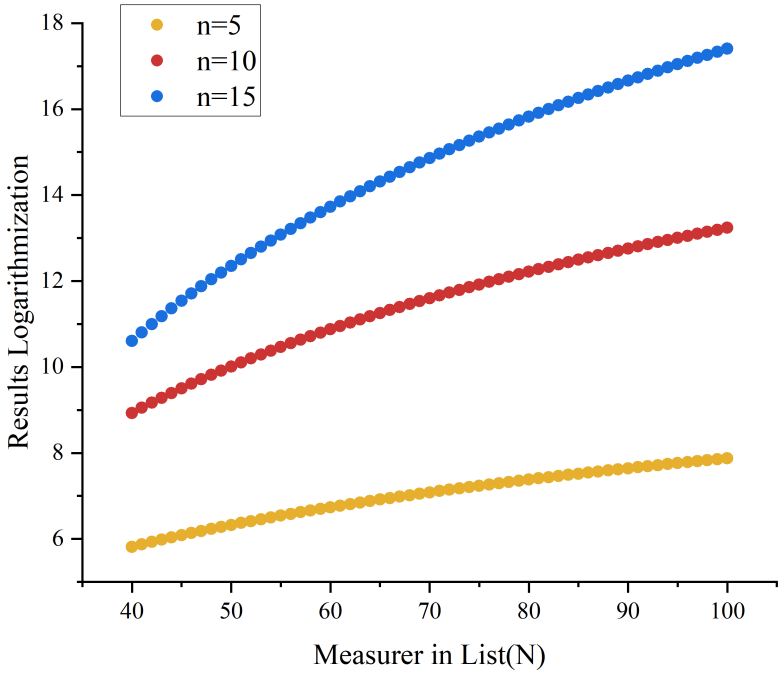

Figure 5. The trend of the order of magnitude of the combination of measurers as N and n increases.

Definition 1. *Given a bandwidth report amount set* $\mathcal{T} = \{0, 1, 2, \ldots, n\}$*, a selected measurer set* $\mathcal{SM} = \{sm_1, sm_2, \ldots, sm_n\}$*, and an attacked measurer set* $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}, k \leq n$*, we define a function* $F : \mathcal{R} \to \mathcal{T}$*, where* $\mathcal{R}$ *refers to any set. Normally, we have* $F(\mathcal{SM}) \to \mathcal{T}$*, but the DoS attack makes* $F(\mathcal{SM} \cup \mathcal{A}) \to \mathcal{T} \backslash \{n\}$*.* $\mathcal{T} \backslash \{n\}$ *is the attack result set of the adversary.* n *is excluded because it means that no measurer is attacked.*

To defend this attack, our mechanism F' should satisfy:

$$\left| Pr[F(\mathcal{SM}) \to \mathcal{T}] - Pr\left[F'(\mathcal{SM} \cup \mathcal{A}) \to \mathcal{T}\right] \right| \leq negl(n).$$

For non-existent virtual measurers, in order to add ghost measurers to the list, the attacker has to pay gas fees for them, which will incur a huge cost. In addition, random selection in SmartMeasurer can limit the number of 'ghosts' participating in the measurement unless the 'ghosts' occupy the majority of measurers and the contract administrator can remove these 'ghosts' once noticed. Random selection and anonymity of identity can also defend another type of DoS attack. The encrypted fingerprint and the blockchain address hardly expose the real information of the measurers. Random selection makes the attacker difficult to block all measurers simultaneously in a measurement round. As long as enough measurers are not attacked and successfully submit measurement results, a new consensus can be published. Thus, SmartMeasurer resists to the DoS attack by making the measurers distributed and anonymous, and minimize the amount of simultaneously attacked measurers, i.e., $|Pr[F(\mathcal{SM}) \to \mathcal{T}] - Pr[F'(\mathcal{SM} \cup \mathcal{A}) \to \mathcal{T}]| \leq negl(n)$.

7. Experiments

The purpose of our scheme is to propose a decentralized bandwidth measurement scheme in conjunction with smart contract on the blockchain and Tor bandwidth measurement. To test the feasibility of the scheme, we have conducted prototype development and validation experiments based on Sbws open-source code and the Ethereum smart contract. We run simulation experiments through two types of networks to demonstrate the feasibility of our scheme.

7.1. Experiment Settings

Smart Contract: We choose a representative blockchain platform—Ethereum and its testnet—because it is the largest platform supporting smart contracts today, and has a large number of nodes (which means safer than a blockchain with few nodes), mature technology, stable testnet, and high compatibility (a lot of sidechains support smart contract). We use Remix, an Ethereum smart contract development platform, and Solidity 0.8.5 to complete the development of smart contract. The smart contract implements the function of registering the measurer and calling Chainlink to complete the periodic call of random numbers. Due to the gas limitation of the smart contract execution, the contracts are deployed on the Ethernet testnet Goerli and gets a 'Goerli ETH' token through its faucet. For the periodic call of random numbers, we subscribe to secure random numbers on Chainlink VRF and apply a keeper on Chainlink Automation, using 'Testnet Link' token for our experiments. We use the *web3.py* library for each relay to communicate with smart contract. For simplicity, our experiments use Infura as an API to access Ether without actually running an Ether node.

Private Tor Network: We build a small private Tor network based on the Tor open-source code using 10 relays, 3 clients, and 1 directory authority. Our relays are distributed in various countries, such as Hong Kong, Singapore, Germany, and Australia, in an attempt to simulate the relay distribution of the public Tor network. These relays have been running for 5 weeks and have steady traffic in the network. The 10 relays include 3 guards, with provider-claimed bandwidths of 10 Mbps, 6 Mbps, and 5 Mbps, while the others are 5 Mbps. However, due to Tor's processing limitations, the bandwidth of these relays does not necessarily equal the provider-claimed bandwidth when working in Tor. We evaluate the performance of each of these relays in the private Tor network by the provider's traffic monitor. The actual forwarding bandwidth of part of relays are reflected in Table 1.

Table 1. Part of relays' information.

	HK1	**AU**	**HK2**	**IND**
Provider-claimed BW (Mbps)	5	5	5	5
Actual BW (Mbps)	5.06	2.28	5.04	3.01
CPU cores	2	2	4	2
RAM (GiB)	8	4	8	4

We manually simulate client traffic for the private Tor network and select the relay to be measured as one of the hops of the client circuit. The scheme is run by guards to verify its feasibility. To further explore the impact on the guard, we will conduct experiments with the guards under high and low load, and discuss the impact on guard performance in Section 7.3.

Denote the total measurer number by N. The $N = 3$ guards, respectively, send registration transactions to the registration contract using their wallet addresses, including the relay's public key and the relay's encrypted fingerprint. In a measurement round, we query a 256-bit random number from Chainlink and split it into $n = 2$ numbers to select n measurers. At the beginning of the measurement, each of the N measurers calls the function of the smart contract to view the random numbers in the current round and checks it with their own serial numbers. After confirming, the measurers scan the whole network using our measurement scheme written based on Sbws 1.6.0 source code. Since our private Tor scale is relatively small and we wanted to obtain more realistic data, we scan the network for 15 days, measuring each relay 20 successful times a day.

Public Tor Network: In order to verify the feasibility of the measurement scheme on a real network and to compare it with the currently Sbws. We run 1 client using HK1 (5 Mps) and run our scheme on the Tor network at different times of the week. Since the bandwidth weight of each relay is affected by the bandwidth of all relays being measured, we choose the average bandwidth value denoted by bw_mean as the comparison. To reduce the impact of the scanner's bandwidth on the measurement results, we add the constraint that the self-reported bandwidth is less than 5 Mbps when selecting the measurement relays.

7.2. Measurement Accuracy

Our evaluation shows the measurement results in private and public Tor network, respectively. We present the average bandwidth of the measured relays and compare it with the official measurement results to demonstrate the measurement accuracy.

Private Tor Simulation: A part of the average bandwidth of our measurements is shown in Figure 6. Considering the occasional instability of the network, there are some outliers in the results, but 50% of the data is still concentrated in a relatively small interval of 0.30 Mbps, which is close to the bandwidth observed by the provider.

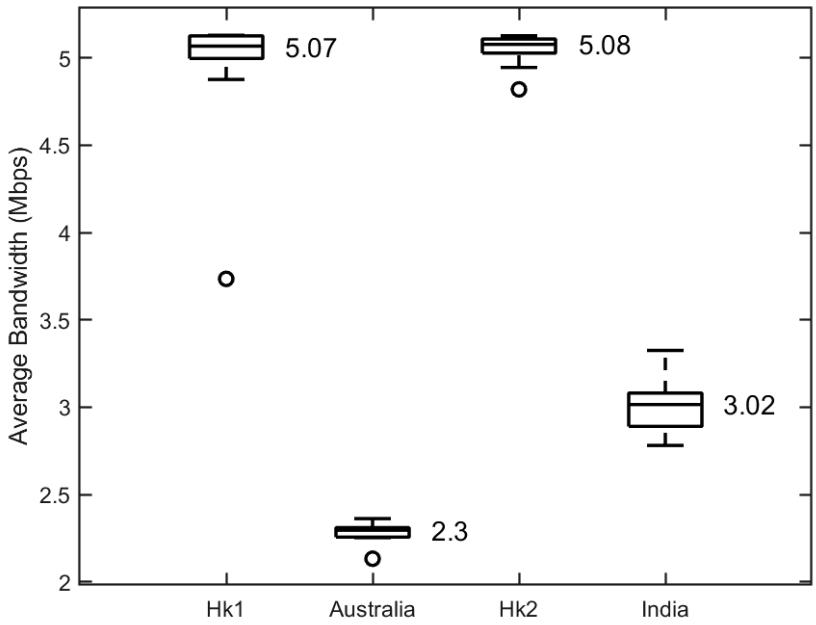

Figure 6. Average bandwidth of relays in private Tor network. The numbers in the figure refer to the median of the average bandwidth of each relay.

Figure 7 shows the average bandwidth measured by SmartMeasurer (SM) and the official (provider). Our results are slightly higher than that of the official and the error is at most 2.2%. This shows that the measurements executed by SM are trustworthy.

Public Tor Simulation: *Moria1* is one of BWAuths. We choose the V3bw files generated from moria1, which are from 10 May 2023 to 16 May 2023, for comparison. We use B_hk_i and B_mor_i to calculate the measurement error $\varepsilon = \mid 1 - (B\_hk\_i / B\_mor\_i) \mid$, where B_hk_i and B_mor_i represent the average bandwidth of relay_i measured by HK1 and moria1, respectively. The measurement errors for both methods are shown in Figure 8. It can be seen that 50% of the data is concentrated between 5% and 20%. The maximum error is under 30% and the average of all relays is 14%. Overall, our experiments show that our method is capable of undertaking the work of the existing network.

7.3. Comparison

In this part, we show the performance impact on the guards during the measurements as well as the cost of SmartMeasurer compared to SmarTor, aiming to prove the high feasibility and lower cost of the solution. We also analyze the scalability of SmartMeasurer and compare it with other bandwidth measurement tools.

Performance Impact: To further explore the feasibility of our scheme, we investigate the impact of guards on bandwidth measurement work. We simulate the scenarios of relays in the Tor network and demonstrate the feasibility by showing that measurement accuracy can be maintained under different workloads, while also supporting client operations. We select 'Singapore' as the stationary guard for all client circuits in the private Tor network,

performing measurement work while forwarding client traffic. We use scripts to manipulate the client to continuously download a 2 MB file, to increase the traffic load on the guard.

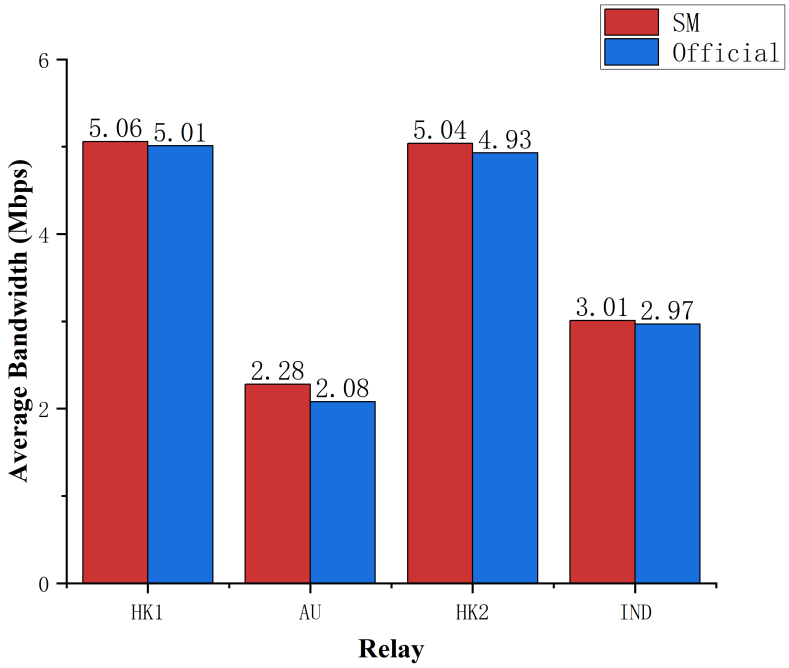

Figure 7. Average bandwidth of relays from SmartMeasurer and the official bandwidth. The measurement results of both parties are relatively close.

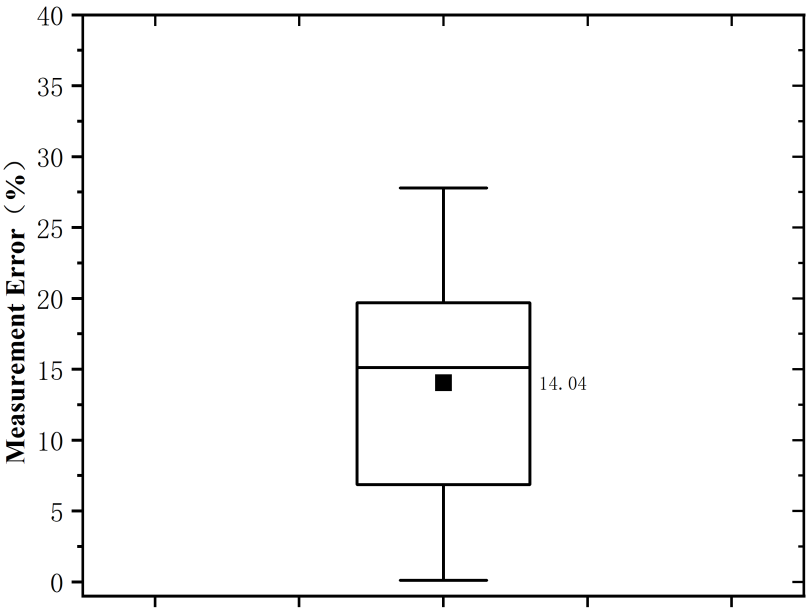

Figure 8. Measurement error in public Tor simulations. The error between the SmartMeasurer and Sbws is from 0% to 30%. The mean measurement error is 14.04%.

Figure 9 reflects the average bandwidth of 'HK1' measured by 'Singapore' with varying client numbers. We find that the measurements overestimate the bandwidth of the relay and the average bandwidth drops more slowly as load increases. This is because the measurement results by SmartMeasurer is the whole circuit average bandwidth while the official shows the bandwidth of single relay. In fact, we prefer the whole circuit bandwidth since it reflects the real bandwidth in use. However, the error is always within an acceptable

range (the error is less than 0.3 MB). Therefore, we believe that the guards have the ability to act as measurers on a part-time basis. It also demonstrates the feasibility of our system.

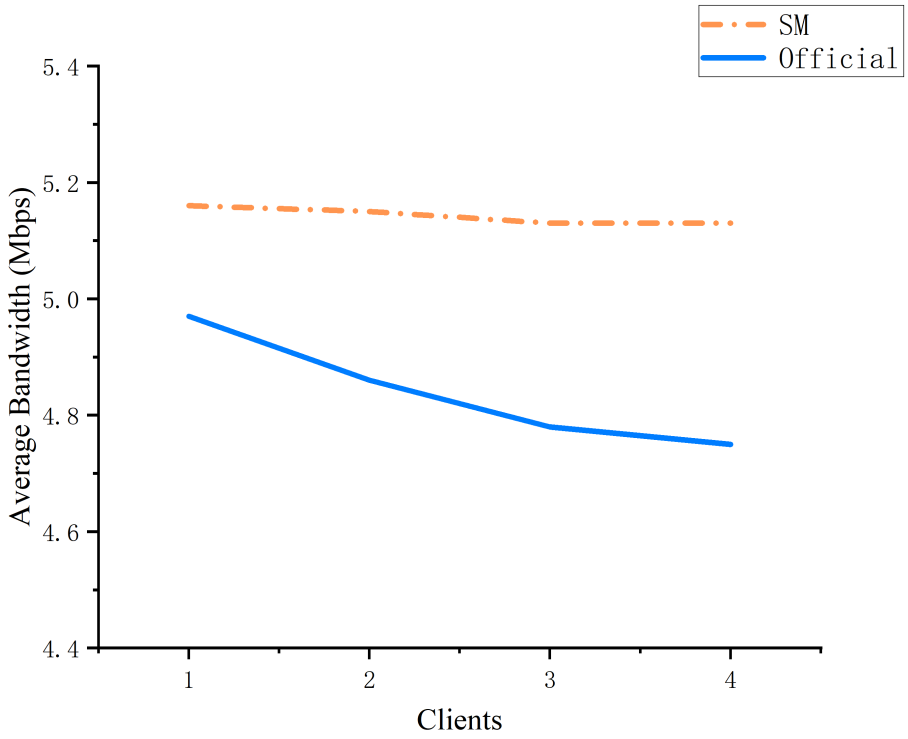

Figure 9. The average bandwidth of 'HK1' measured by 'Singapore' under different loads.

System Cost: We estimate our system cost based on the functions required for the entire system, as shown in Table 2. To estimate the real cost of our system, we use the rate of 19 April 2023, which is 1 Link = 0.004 ETH, 1 ETH = 1982.09 USD. Deploying the smart contracts and registering an Upkeeper are the most expensive operations but, needs only be performed once. The total cost of once operations is just under 25 USD. In Tor, the BWAuth's scanner needs 2 days to scan the entire network with a bandwidth of 1 Gbps [10]. However, considering that the bandwidth of the guards is less than that of the BWAuth, we can expand the number of measurers and we will have 10 measurers per round to complete the measurement. However, we do not need to increase the request numbers (which will increase costs), since a random number from VRF has 256 bits and it can be split into many numbers as configuration. This would result in a system cost of less than 4 USD per month for the administrator.

Compared to SmarTor, our solution significantly reduces gas costs. SmarTor requires measurers to actively send transactions to join measurement rounds and report the measurement results to the smart contract. Additionally, the smart contract incurs significant gas costs in assignment, verification, and report aggregation. In our solution, SmartMeasurer actively assigns tasks (by Chainlink) and receives and aggregates reports off-chain, resulting in a significant reduction in gas consumption. Moreover, SmarTor assumes that 1 gas = 3 gwei and 1 ETH = 500 USD. In 2023, the gas price reached 140 gwei or even more, and the Ether price is already four times higher than it was then. Assuming that the Ether used remains the same, its realistic cost will become four times greater than it was, which would be far greater than our scheme.

Table 2. System Cost.

Operation	ETH	US Dollar
Once Cost		
Deploy Registration Contract	0.0018	3.79
Deploy Randomness Contract	0.0034	7.16
Create a Subscription	0.0004	0.84
Add Consumer	0.0003	0.63
Register new Upkeeper	0.0057	12.01
Measurer Registration	0.0045	9.48
Total	0.0116	24.44
Non-Once Cost		
Perform Upkeep	0.0004	0.84
Call Request Function	0.0003	0.59
Request a Random Number	0.0012	2.38

Scalability: The scalability of SmartMeasurer for an increasing number of Tor relays is consistent with Tor, because SmartMeasurer uses the same measurement tool. The current scale of the Tor network is increasing, and there are still only 5 BWAuths officially scanning the entire network. As the number of relays in the Tor network continues to increase, the time of scanning entire network will be longer. We measured 134 nodes in 1 hour with one measurer. At this efficiency, a measurer can measure nearly half of the network nodes in 24 h. This means that SmartMeasurer can update information on more than 3000 nodes per day. This is enough for the requirement of the Tor, because Tor does not measure all relays every round in fact and it does not affect the usage whether the bandwidth weights are latest or not. If the size of the network is too large to require additional measurers, the administrator can redeploy the contract to modify the number of measurers.

Another Tool: We compare the measurement results with iPerf (a network performance measurement tool). We use iPerf to measure 'HK1' on work for a few times and calculate the average bandwidth. Table 3 shows that the two measurements are close, but the result of iPerf are higher because it is point to point measurement. The result of SmartMeasurer is closer to the real bandwidth of relay working in Tor.

Table 3. Measurement results by iPerf and SmartMeasurer.

	iPerf	SM
HK1	5.11	5.06

8. Conclusions

In this paper, we present SmartMeasurer, a method for decentralizing measurement system by using emerging blockchain technologies such as Smart Contract and Oracle, to assign measurements to measurers that also act as guards in Tor. We implement the SmartMeasurer and simulate on a private network as well as the public network and evaluate its performance, security, and cost. The results demonstrate the higher feasibility, lower cost, and higher security of our solution.

Future Work: In SmartMeasurer, the measurements are operated by guards trusted by the authority and their services are not rewarded in any way. Furthermore, their interactions with smart contract increases the cost of gas. We plan to investigate additional incentives for measurers in future work. Furthermore, we also have to consider deploying smart contract on different blockchains to enhance the robustness and reduce the cost of our system. In addition, as our method is an improvement on the source code of Sbws, which has limitations regarding its measurement techniques, in the future, we plan to introduce existing measurement techniques such as FlashFlow and MLEFlow into our system to make the measurements more accurate.

Another issue worth discussing is about the security of smart contracts. Nowadays, the security of smart contracts is gradually being taken seriously, and there have been many

cases showing that smart contract vulnerabilities can lead to huge economic losses [35,36]. The contract of SmartMeasurer does not involve external transfers, and the current attack methods cannot steal the contract's native funds, so it will not affect the SmartMeasurer. Other types of attacks are beyond the scope of the research in this paper. In the future, we consider using formal methods [37,38] to analyze the code of smart contract to further prove the security of smart contract in our scheme.

Author Contributions: Conceptualization, Z.T.; methodology, Z.T. and T.P.; software, Z.T. and Y.H.; validation, Z.T.; writing—original draft preparation, Z.T., T.S. and T.P.; writing—review and editing, Z.T., T.P., L.X. and D.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2022YFB2702402 and 2022YFB2702405 and National Natural Science Foundation of China (NSFC) under the grant No. 62172040.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to express their deepest gratitude to the National Key Research and Development Program of China for funding the work and the editors and reviewers for their invaluable suggestions which led to the betterment of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tor Metrics. Available online: https://metrics.torproject.org/ (accessed on 5 May 2020).
2. Andre, G.; Alexandra, D.; Samuel, K. Smartor: Smarter tor with smart contracts: Improving resilience of topology distribution in the tor network. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 677–691.
3. Jansen, R.; Vaidya, T.; Sherr, M. Point Break: A Study of Bandwidth Denial-of-Service Attacks against Tor. In Proceedings of the USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; pp. 1823–1840.
4. Wacek, C.; Tan, H.; Bauer, K.S.; Sherr, M. An Empirical Evaluation of Relay Selection in Tor. In Proceedings of the NDSS, San Diego, CA, USA, 25–27 February 2013.
5. Biryukov, A.; Pustogarov, I.; Weinmann, R.P. Trawling for tor hidden services: Detection, measurement, deanonymization. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 19–22 May 2013; pp. 80–94.
6. Johnson, A.; Wacek, C.; Jansen, R.; Sherr, M.; Syverson, P. Users get routed: Traffic correlation on Tor by realistic adversaries. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 337–348.
7. Nasr, M.; Bahramali, A.; Houmansadr, A. Deepcorr: Strong flow correlation attacks on tor using deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1962–1976.
8. Thill, F. Hidden Service Tracking Detection and Bandwidth Cheating in Tor Anonymity Network. Ph.D. Thesis, University of Luxembourg, Luxembourg, 2014.
9. Johnson, A.; Jansen, R.; Hopper, N.; Segal, A.; Syverson, P. PeerFlow: Secure Load Balancing in Tor. *Proc. Priv. Enhancing Technol.* **2017**, *2017*, 74–94. [CrossRef]
10. Traudt, M.; Jansen, R.; Johnson, A. Flashflow: A secure speed test for tor. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 7–10 July 2021; pp. 381–391.
11. Why Is Ethereum (ETH) Going Up Today? Available online: https://www.forbes.com/advisor/in/investing/cryptocurrency/why-is-ethereum-going-up/ (accessed on 5 May 2023).
12. Darir, H.; Sibai, H.; Cheng, C.Y.; Borisov, N.; Dullerud, G.E.; Mitra, S. MLEFlow: Learning from History to Improve Load Balancing in Tor. *Proc. Priv. Enhancing Technol.* **2022**, *2022*, 75–104. [CrossRef]
13. Torflow. Available online: https://gitlab.torproject.org/tpo/network-health/torflow (accessed on 5 May 2009).
14. Nilsson, A.; Bideh, P.N.; Brorsson, J. A survey of published attacks on Intel SGX. *arXiv* **2020**, arXiv:2006.13598.
15. Sgaxe: How Sgx Fails in Practice. Available online: https://sgaxe.com/ (accessed on 5 May 2020).
16. Ragab, H.; Milburn, A.; Razavi, K.; Bos, H.; Giuffrida, C. Crosstalk: Speculative data leaks across cores are real. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021; pp. 1852–1867.
17. Malicious Relays and the Health of the Tor Network. Available online: https://blog.torproject.org/malicious-relays-health-tor-network/ (accessed on 25 April 2022).
18. Simple Bandwidth Scanner's Documentation. Available online: https://tpo.pages.torproject.net/network-health/sbws/README.html (accessed on 5 May 2018).

19. Tor Directory Protocol, Version 3. Available online: https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt (accessed on 8 May 2023).
20. Differences between Torflow and Sbws. Available online: https://tpo.pages.torproject.net/network-health/sbws/differences.html (accessed on 5 May 2018).
21. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decent. Bus. Rev.* **2008**, 21260. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 5 May 2023).
22. Ye, C.; Li, G.; Cai, H.; Gu, Y.; Fukuda, A. Analysis of security in blockchain: Case study in 51%-attack detecting. In Proceedings of the 2018 5th International Conference on Dependable Systems and Their Applications (DSA), Dalian, China, 22–23 September 2018; pp. 15–24.
23. Chen, H.; Pendleton, M.; Njilla, L.; Xu, S. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–43. [CrossRef]
24. Heilman, E.; Kendler, A.; Zohar, A.; Goldberg, S. Eclipse attacks on bitcoin's peer-to-peer network. In Proceedings of the 24th {USENIX} Security Symposium ({USENIX} Security 15), Washington, DC, USA, 12–14 August 2015; pp. 129–144.
25. Mirkin, M.; Ji, Y.; Pang, J.; Klages-Mundt, A.; Eyal, I.; Juels, A. BDoS: Blockchain denial-of-service. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 9–13 November 2020; pp. 601–619.
26. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
27. Chainlink Overview. Available online: https://docs.chain.link/getting-started/conceptual-overview (accessed on 5 May 2023).
28. Kaleem, M.; Shi, W. Demystifying pythia: A survey of chainlink oracles usage on ethereum. In *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, 5 March 2021, Revised Selected Papers 25*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 115–123.
29. Gonçalves, J.P.D.B.; Alochio, G.; Villaça, R.D.S.; Gomes, R.L. Data Integrity Verification in Network Slicing using Oracles and Smart Contracts. In Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 22–25 August 2022; pp. 476–481.
30. Chainlink Use Cases. Available online: https://chain.link/use-cases (accessed on 5 May 2023).
31. Gayoso Martínez, V.; Hernández Encinas, L.; Sánchez Ávila, C. A survey of the elliptic curve integrated encryption scheme. *J. Comput. Sci. Eng.* **2010**, *2*, 7–13.
32. Abbas, M.M.; Merad-Boudia, O.R.; Gasmi, S. Secure Multidimensional Data Aggregation in IoT-Fog Environments using ECIES. In Proceedings of the 2022 First International Conference on Computer Communications and Intelligent Systems (I3CIS), Jijel, Algeria, 22–23 November 2022; pp. 7–12.
33. Darir, H.; Borisov, N.; Dullerud, G. DiProber: Using Dual Probing to Estimate Tor Relay Capacities in Underloaded Networks. *arXiv* **2022**, arXiv:2211.16751.
34. The Lifecycle of a New Relay. Available online: https://blog.torproject.org/lifecycle-new-relay (accessed on 11 September 2013).
35. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart Contract: Attacks and Protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
36. He, D.; Deng, Z.; Zhang, Y.; Chan, S.; Cheng, Y.; Guizani, N. Smart Contract Vulnerability Analysis and Security Audit. *IEEE Netw.* **2020**, *34*, 276–282. [CrossRef]
37. Krichen, M.; Lahami, M.; Al–Haija, Q.A. Formal Methods for the Verification of Smart Contracts: A Review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022; pp. 1–8. [CrossRef]
38. Abdellatif, T.; Brousmiche, K.L. Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–5. [CrossRef]