# Extended SNMP for Integrity Verification in Distributed Systems

Muhammad Amjad[1], Fazal Wahab[2,*], Anwar Shah[2], Muhammad Khalid[3], and Muhammad Irfan Saeed[2]

[1]Department of computer Science, Institute of Management Sciences (IMSciences), Peshawar, Pakistan
[2]Department of AI and Data Science, National University of Computer and Emerging Sciences, Chiniot Campus, Pakistan
[3]School of Software Engineering, Dalian University of Science and Technology, China
Email: contactamjid@gmail.com (M.A.); fazalwahabstu@gmail.com (F.W.); anwar.shah@nu.edu.pk (A.S.); m.khalid@mail.dlut.edu.cn (M.K.); iffiawan4@gmail.com (M.I.S.)
*Corresponding author
Manuscript received September 2, 2025; accepted December 5, 2025; published February 13, 2026

***Abstract*—As dependence on computer technology grows, the need for improved security becomes increasingly essential. The majority of people are connected to networks through various means, such as mobile phones, Automated Teller Machines (ATMs), online banking, and social media. Researchers have proposed various strategies to protect user data, however, security remains a primary concern as users continue to face numerous challenges. In order to cope with these technological challenges, Trusted Computing Group (TCG) introduced the Trusted Platform Module (TPM), a hardware-based cryptographic chip designed for system integrity verification. The TPM provides hardware-based integrity verification; however, there is no existing protocol to remotely monitor integrity across multiple systems in a distributed network. This study proposes an extended Simple Network Management Protocol (SNMP) architecture that collects TPM-verified integrity values of distributed systems and report them to the network administrator. On the administrator side, these values are matched against stored signatures to determine the integrity of these systems. To validate our approach, a Nagios-based monitoring system is used to displays the integrity status of overall network, distinguishing between trusted and compromised devices. Experimental results indicate that the Extended-SNMP solution achieve low overhead, high scalability, and reduced false positives compared to conventional host-based integrity monitoring techniques such as Host-based Intrusion Detection System (HIDs) or Network Intrusion Detection Systems (NIDs). This approach enhances real-time security visibility in distributed environments, making it a practical alternative for large-scale network security management.**

***Keywords*—extended Simple Network Management Protocol (SNMP), integrity, Trusted Computing Group (TCG), Trusted Platform Module (TPM), Integrity Measurement Architecture (IMA), root of trust for measurement**

## I. Introduction

The increasing adoption and rapid development of technology have led to a growing complexity in modern systems. Technology makes it easier for an inexperienced user to access internet services and enables businesses to offer online services such as digital payments and e-commerce. However, this widespread reliance on technology exposes users to numerous security risks every day, leading to an estimated annual loss of $1 trillion due to data breaches, ransom-ware, and financial fraud [1]. Information security experts are responsible for preventing and minimizing these risks and providing a secure digital environment. Server-side security is typically reinforced by implementing various security policies such as firewalls, IDS, anti-viruses, updates, patches, and routine network monitoring tools [2]. The client side needs to manage these vulnerabilities, which requires security-focused strategies. Incorrect software settings or improper programming practices might cause such assaults [3]. Security software, such as antivirus programs, can be used to remove malware; however, recent research has revealed that software alone is not sufficient to ensure system security as it remains susceptible to attacks [4]. Some attacks are so sophisticated that they can compromise the BIOS or modify the Master Boot Record (MBR), infect the operating system, and disrupt the application behavior [5], as conventional antivirus software cannot shield the system from attacks at the BIOS level. Therefore, operating system security cannot be achieved solely by software-based solutions [6, 7].

To mitigate these challenges, hardware-based techniques are required to encipher and store sensitive data in tamper-resistant memory with restricted access to unauthorized users. TCG implements a hardware Root of Trust (RoT) strategy through the Trusted Platform Module (TPM). In 2004, TCG TPM 1.2 and later updated to TPM 2.0 [8, 9]. The TPM chip is a hardware security module that incorporates encryption algorithms and provides cryptographic functions. Verifies the integrity status of running applications and stored data using Platform Configuration Registers (PCRs) that are tamper-proof storage locations and can be accessed only with a trusted software stack [10]. Attestation is another key feature of TPM that ensures the trustworthiness of a target platform by generating hash measurement [11]. This process begins at the hardware root of trust and extends through the boot loader, BIOS, and Operating System kernel, enabling to detect any modification to critical system modules. The TPM works in conjunction with the processor (CPU) and other components to provide secure operations, key management, and system integrity verification.

The Simple Network Management Protocol (SNMP) is an application layer (Layer 7) protocol designed to collect and report network configuration and status information to administrators. Extended SNMP enhances this functionality by allowing customized implementation to meet specific security and monitoring requirements. The proposed design leverages Extended SNMP to securely transmit TPM-generated integrity status of each client to a centralized monitoring server for security enforcement.

This research aims to achieve the following objectives.

- Extend SNMP to securely collect and report the integrity status of connected node with minimum overhead and

maximum scalability.
- Develop a network-wide monitoring interface to enable real-time assessment of network integrity.

The paper is divided into multiple sections. Section II provides background information and a review of related literature, while Section III details the proposed solution, while Section IV describes its implementation and presents the experimental results. Finally, Section V concludes the paper by summarizing key findings and outlining potential directions for future research.

## II. Background

This section presents the foundation concepts relevant to this study, including Simple Network Management Protocol (SNMP), trusted computing, the Trusted Computing Group (TCG), Trusted Platform Module (TPM), Integrity Measurement Architecture (IMA), and distributed systems. Additionally, it provides a comparative analysis of proposed architecture with existing approaches in the literature.

### A. Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) is a widely used standard protocol designed to monitor and manage network-based devices. It collects network status information, which is reported to the administrators for necessary actions. SNMP operates at the application layer of TCP/IP suit standardized by Internet Engineering Task Force (IETF) [12]. Each client (managed device) maintains its configuration data in a hierarchical database called Management Information Base (MIB). The MIB consists of many objects, identified by unique Object Identifier (OID). Each OID corresponds to specific configuration data such as system boot time, system name, memory information, and kernel version. An SNMP Manager (server) retrieves this information by querying specific OID from the managed device through Network monitoring tools like Nagios or MRTG [13].

### B. SNMP Architecture

The network management model for SNMP consists of the following key elements [14].
- Manager is the administrative software installed on the server to manage client devices. Manager queries agents for required information or receives trap notification when any changes or errors occur.
- The agent is a software process running on client (managed) devices interacting with MIB and responding to Manager queries.
- Management Information Base is a hierarchical database that define OIDs representing various configuration parameters of a managed device.
- The Network Management System (NMS) provides an interface for network administrators to monitor and control network devices.
- Managed Device is any IP-enabled node (e.g., computer, camera, printer) monitored by NMS.

### C. Supported Transport Protocol

SNMP can use both UDP and TCP to communicate across a network. UDP is often preferred due to its light weight design and low overhead in large-scale network. In situations where reliable delivery and acknowledgment is required, TCP can be used as an alternative for SNMP transport.

### D. SNMP Version-3 Format

In order to enable communication between the Manager and Agent, Protocol Data Units (PDUs) are encapsulated in SNMP message format. The Field “Message Version Number” refers to the SNMP version in use, such as SNMPv1, SNMPv2 or SNMPv3. The field “Maximum Message Size” represents the maximum allowable length of an SNMP message, ensuring compatibility with network. The “Message Security Model” specifies security framework applied in the SNMP communication, such as User-based Security Model (USM). The fourth byte of the security model field “Security Flags”, which consists of four elements: ‘Auth’, ‘Priv’, ‘Reportable Flag’, and ‘Reserved’. These flags indicate the security mechanisms used in SNMPv3 communication, including authentication (Auth), privacy (Priv), message reporting (Reportable Flag), and a reserved field for future use.

This study proposes utilizing one of the reserved SNMPv3 flags as Integrity Verification (IV flag). When the integrity state of a node is queried, the IV flag is activated, signaling the presence of verified integrity token in the SNMP response.

### E. SNMP Version-3 Security

SNMP version 3 was introduced to address the security limitations of its predecessors by incorporating enhanced security features such as confidentiality, integrity, and availability [15, 16]. It achieves this by encapsulating SNMP version 2 Protocol Data Units (PDUs) and implementing advanced security mechanisms, including the User-based Security Model (USM), Transport Layer Security (TLS) and View Access Control Model (VACM) [17, 18].

In this research, SNMPv3 is implemented to enhance network security by enabling integrity measurement. SNMPv3 comprises the following key.

1) Security Subsystem: The security subsystem of the SNMPv3 engine is responsible for ensuring authentication and privacy for managed devices. In SNMPv1 and SNMPv2, community strings are used for authentication, whereas SNMPv3 implements user-based authentication mechanism. The User-based Security Model (USM) replaces plain text passwords with cryptographic Message Digest 5 (MD5) or the Secure Hash Algorithm (SHA-1) for authentication purposes [19]. However, due to known vulnerabilities in existing cryptographic algorithms, modern implementations increasingly integrate stronger algorithms, such as SHA-256 and AES encryption, to enhance security [20].
2) Access Control Subsystem: This module regulates access to Management Information Base (MIB) objects, ensuring that only authorized users can retrieve or modify designated data [21]. It enforces access control by restricting read/write operations based on predefined permissions.
3) The Message Processing The message processing subsystem is responsible for encoding SNMP messages for transmission and decoding relevant information from received messages. It may include different submodules to handle different versions of SNMP, ensuring compatibility and efficient message processing.

4) The dispatcher component is responsible for sending and receiving messages across local and global networks. Identifies the SNMP version of each incoming message and directs it to the appropriate message processing module for handling.

## III. Trusted Computing

It is an advanced security paradigm that strengthens existing frameworks by integrating a hardware-based security mechanism. The Trusted Computing Group (TCG) aims to enhance the security of computing platforms by establishing standardization guidelines, originally developed by the Trusted Computing Platform Alliance (TCPA) [22]. The Trusted Platform Module (TPM) is a dedicated hardware security chip that employs cryptographic techniques to verify the integrity and worthiness of software running on a device. Additionally, TPM enhances the security of input/output (I/O) operations and data storage by restricting access to authorized entities. Beyond traditional computing devices, TPM is increasingly utilized in cloud computing environment to enable secure boot, remote attestation, and encryption [23]. The Trusted Computing Group (TCG) enhances computer security privacy by developing standardized protocol and specifications. Establishing trust in remote systems is crucial for enterprise applications and protecting confidential data. The Trusted Computing Group (TCG) addresses this through an attestation mechanism that originates from the Hardware Root of Trust and extends to the BIOS, bootloader, OS kernel, and application layer to verify system integrity.

This research optimizes the attestation process by restricting integrity verification to executable, binary files, and library files, reducing computational overhead while maintaining security. This targeted approach enables attack detection at every stage of the system. A chain of trust is initiated that verifies the entity responsible for measuring trust to be itself verifiable and trustworthy.

## IV. Trusted Platform Module

The Trusted Platform Module (TPM) is a hardware security chip that adheres to international standards to store passwords, encryption keys, and digital certificates to authenticate a platform [24]. Its technical specifications were established in 2009 by the Trusted Computing Group, a consortium of leading technology companies. Authentication verifies a device's identity, while integrity ensures that the platform's state remains unaltered. TPM 2.0 is the latest version of TPM specification, supporting various cryptographic algorithms including Advanced Encryption Standard (AES), Elliptic Curve Cryptography (ECC), RSA, and the SHA-2 family (e.g., SHA-256). It also enables secure authentication and data integrity mechanisms using HMAC [24]. Table 1 details the comparison of TPM 1.2 and TPM 2.0. Hardware-based security is designed to address the limitations of software-based security which is inherently prone to failures and cyber-attacks, and misconfiguration [25, 26]. Hardware-based mechanisms significantly reduce the risk of unauthorized access to sensitive information. The Trusted Platform Module (TPM) checks system integrity by performing boot time verification and detects unexpected configuration changes, ensuring that only trusted code is executed during system start-up [27].

TPM-enabled systems are deployed to enhance security in various applications that leverage TPM technology. These include secure email by storing cryptographic keys, online shopping, internet banking for secure transactions, Bit-Locker for disk encryption, secure communication via Trusted Network Connect (TNC) for secure network access, and password management systems.

Table 1. TPM 1.2 and TPM 2.0 specification

| Features | TPM 1.2 | TPM 2.0 |
|---|---|---|
| Algorithm Support | SHA-1, RSA | SHA-256, RSA, ECC |
| Algorithm Flexibility | Fixed algorithms | Extensible algorithms |
| Key Hierarchies | Single hierarchy | Multiple hierarchies |
| PCR Support | Fixed (24) | Flexible |
| NVRAM Storage | Limited | Extended |
| Authorization | HMAC-based | Policy-based |
| OS and Hardware | Limited | Broad adoption |

### A. Secure Storage

TPM provides secure Platform Configuration Registers (PCRs), which store integrity measurements in protected memory space. These PCRs hold cryptographic hashes representing system states. The TPM_Seal operations encrypt sensitive data while binding it to specific PCR values; the data can only be unsealed if the system integrity matches the expected state [28]. For remote integrity verification, a remote system challenges TPM, which responds by signing the challenge with its private Attestation Identity Key (AIK). The remote system then verifies this signature using the TPM public key to verify the system integrity before establishing communication [29].

### B. Secure Execution

Attestation ensures the integrity of code and system component. The secure execution engine in TPM in conjunction with the Integrity Measurement Architecture (IMA) measure the integrity of executable and stores these hashes in PCRs, and in Stored Measurement Log (SMLs) for audit purpose. The chain of trust begins at boot, where each executable hash is matched against reference values for integrity verification.

### C. Key Generation

TPM generates various cryptographic keys for security and attestation. These include Storage Root Key (SRK) for protecting other keys, Endorsement Key (EK) for TPM authenticity, Attestation Identity Key (AIK) for signing integrity measurements, and other general keys for attestation and encryption purposes.

### D. Cryptographic Functions

TPM has many cryptographic functions, including the Rivest Shamir Adleman (RSA) encryption for asymmetric encryption and Secure Hash Algorithm 1 (SHA-1) for cryptographic hashing. With TPM 2.O, Elliptic Curve Cryptography (ECC) was introduced to enhance cryptographic efficiency and security. The Input/Output interfaces allow external applications to interact with these functions. The following is a detailed explanation of each of these functions.

- RSA Engine in TPM 2.0 is used for public key operations such as digital signature verification, key exchange, and

encryption/decryption. Additionally, it supports the generation of RSA key pairs that can be utilized in these operations.

- SHA Hashing TPM supports multiple cryptographic hashing algorithms, including the Secure Hash Algorithm (SHA) family. Earlier version of TPM relied on SHA-1; however, due to its known vulnerabilities, it has been deprecated. TPM 2.0 incorporates SHA-256, which offers enhance security and greater resistance against cryptographic attacks [30]. Additionally, The TPM hashing module support sealing and unsealing data operations, which protect sensitive data by binding it to a specific platform state. This process use a proof key, derived from public part of Attestation Identity Key (AIK), to enable controlled access and delegation of specific privileges based on predefined authorization policies.
- The Random Number Generator (RNG) in TPM 2.0 produces highly unpredictable sequence essential for key generation, nonce creation, and other cryptographic operations. TPM 2.0 supports NIST-compliant random number generation and policy-based entropy control, enhancing security and flexibility in cryptographic applications [31].
- Elliptic Curve Cryptography (ECC) is a public-key cryptographic scheme that provides equivalent security to traditional algorithms like RSA while requiring smaller key size leading to improved computational efficiency, reduced memory usage, and lower power consumption. These attributes make ECC particularly well-suited for embedded systems and Internet of Things (IoT) devices [32].

## V. Integrity Measurement Architecture

The Integrity Measurement Architecture (IMA) is a module in the Linux operating system designed to measure and verify the integrity of files and applications executed on a system. It integrates with Trusted Platform Module (TPM) to enforce system's integrity during boot time and runtime. The IMA maintains a secure measurement log, along with their cryptographic hash values. This log is stored in system memory while the hash values are extended in TPM's Platform Configuration Registers (PCRs). This allows remote attestation and verification of system integrity at any given time [33]. Fig. 1 illustrates the design and workflow of Integrity Measurement Architecture. ima_tcb = 1 command is used to enable IMA in Linux.

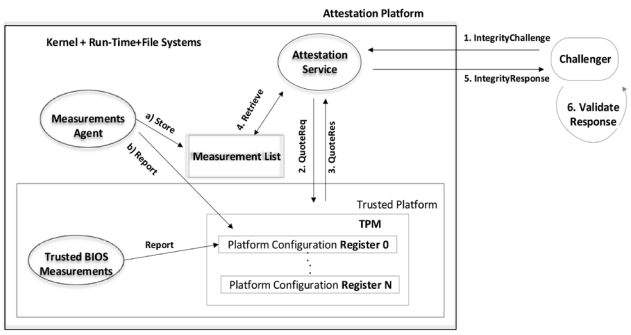


Fig. 1. Integrity measurement architecture design (34).

## VI. Literature Comparison with Proposed Architecture

This paper presents two main contributions. First, it utilizes Trusted Platform Module (TPM) to verify system integrity. Second, it extends the Simple Network Management Protocol (SNMP) to facilitate remote reporting of integrity measurements over a network. This approach enables remote monitoring and management of the system's security postures. While various techniques exists for remote platform attestation and reporting; this study specifically integrates TPM and SNMP to achieve a secure and efficient solution. The following section provides a detailed review of existing literature, analyzing various methodologies and their relevance with proposed approach.

Sailer *et al.* [34] developed the Integrity Measurement Architecture (IMA) for Linux, enabling remote attestation via the TPM. During boot, the system computes and stores the measurements (i.e., binary hashes) of all its components and applications in the Platform Configuration Registers (PCRs). This aggregate value (hash) is presented to the challenger, who recomputed and compares these values. A match confirms the integrity of the system. However, IMA primarily performs static integrity measurement, lacks the capability to monitor runtime behavior, and is not very practical in heterogeneous environments. PRIMA [35] extends IMA by integrating information flow tracking in Security Enhanced Linux (SELinux) for integrity verification. PRIMA aims to reduce the overhead of the policy-based measurement system by minimizing the number of measurements. Unlike IMA, that measures all loaded binaries, PRIMA enables selective integrity measurements, allowing the verification of specific applications based on policy rules.

Du *et al.* [36] proposed a model for runtime behavior measurements in virtual machines, specifically in cloud computing environments. Their approach involves a virtual TPM (vTPM) to perform integrity verification. In contrast, our research employs a physically installed TPM at the endpoint to report the integrity of the target system.

In contrast to the work of Faisal *et al.* [37], which focuses on establishing trust between users of Internet of Thing (IoT) Devices, our research leverages TPM for integrity verification in a distributed network. While both studies utilize TPM for security purposes, the intended application and implementation scope differ significantly.

Lu *et al.* [38] proposed a technique employing TPM to prevent unauthorized access and mitigate malware attacks using hardware-based security mechanism. Similarly, our research uses hardware based security (i.e.,TPM) for integrity verification.

Xing *et al.* [39] introduced an attestation technique for guest Virtual Machines (VMs) termed Out-of-Box IMA (OB-IMA). Unlike conventional approaches, OB-IMA extends integrity verification to system configurations, interpreters, scripts, and other critical files considered by existing techniques. This flexible approach supports both system-generated and manual defined measurements. The technique has a key limitation as its exclusive focus on Guest VMs integrity, lacking network-level reporting of system integrity. Thom *et al.* [40] presented a technique for integrity measurement that leverages the Trusted Platform Module

(TPM) of the host device when a client device lacks a TPM chip. This approach allows the clients to utilize the host's TPM device for trust services.

Matoušek *et al.* [41] extended SNMP's capabilities to monitor nodes in IoT such as IP cameras, sensors, actuators, and intelligent devices integrating their reports into a remote monitoring interface. However, this extension lacked security measures particularly integrity and confidentiality of the nodes. Our research extends SNMP to address these concerns.

Kim *et al.* [42] proposed an approach that integrates SNMP with an Intrusion Detection System (IDS) to detect attacks in network traffic. This approach uses SNMP Management Information Base (MIB) to create a lightweight and fast attack detection system compared to traditional packet-level IDS analysis, which is often resource-intensive and prone to delayed reporting. However, this technique does not incorporate integrity verification at the operating system level. In contrast, our approach utilizes TPM-based integrity verification.

Wang *et al.* [4], in their research patent on Network Operation, Administration, and Maintenance (OAM) for network devices proposed an extension of SNMP to report fault alarm and messages for maintenance. Our research also extends the SNMP to collect and report the integrity status of nodes within a network.

Based on the existing literature discussed above, this research proposes an OS-level integrity verification mechanism using extended SNMP. This approach integrates TPM with SNMP for remote monitoring of OS integrity. This approach is lightweight and maintains network confidentiality.

## VII. Methodology

This section outlines the methodology applied in this research. The implementation was carried out in a real world environment without any simulation. The primary objective is to enhance SNMP capabilities, enabling it to collect and report integrity information alongside standard system status metrics.

### A. Proposed Architecture

This section provides a detailed description of the proposed architecture and its components. This architecture comprises two main entities: The challenger and the target client, whose integrity status is verified. The developed module enables the challenger to retrieve client integrity metrics using extended SNMP. The client system is equipped with TPM that generates cryptographic hashes for specified files. These hashes are bonded with customized MIB within the extended SNMP and returned as a response to the challenger's query. The complete architecture workflow is illustrated in Fig. 2 and is detailed in the following sections.

1) Request Initialization: The challenger initiates the process by sending an SNMP GET request to the target to retrieve integrity measurements.
2) Authentication and Authorization: Upon receiving the request, client system authenticates the challenger and, if valid, extends the PCR values from TPM.

a) The TPM is configured to measure specific files such as executable, binaries, and libraries. This is an internal process of each node. This requests the TPM to provide the latest hash value by reading PCRs.

b) The TPM respond by current PCR hash value i.e., computes hashes for the targeted files.

3) All these computed hashes are extended into PCR-10 using PCR_EXTEND command ensuring cumulative integrity measurement. The output is redirected to a text file.
4) A custom Hash-OID binding Function associates the retrieved PCR value with an SNMP-reserved OID (Object Identifier), allowing it to be queried like other SNMP standard OIDs.
5) The SNMP Agent now has the hash value stored in the MIB, making it available for remote querying.
6) The SNMP agent responds with the bound hash value sending to challenger for verification.
7) Integrity Verification on Challenger side: A Python based "Hash-Appraisal Function" compares the retrieved hash in with a previously stored reference hash in the database.
8) Integrity Verification: The function return TRUE if the hashes match, indicating client integrity is maintained or FALSE, if there is a mismatch indicating possible tampering.
9) The verification result is sent to the Sub-Manager for integration with Nagios.

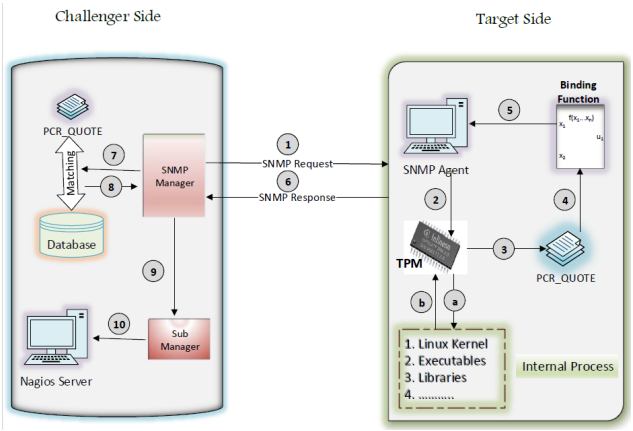


Fig. 2. Proposed architecture.

### B. Entities in the Proposed Architecture

The proposed architecture includes entities responsible for measuring and verifying system integrity, as outlined below.

- Challenger: The challenger is an administrative system that verifies the integrity of clients. It initiates SNMP-GET request to retrieve integrity-related information from managed device.
- Target: The target refers to a TPM enabled client device whose integrity is being verified and monitored.
- SNMP Manager: This module, installed on the challenger system, serve as the main SNMP component that sends the request to the SNMP agent on the client device.
- SNMP Agent: SNMP agent is a module installed on managed/client systems that interact with Management Information Base (MIB). For this research, the agent MIB has been extended with a custom Object Identifier (OID) to support integrity reporting
- Trusted Platform Module: In this research, TPM generates integrity measurements by hashing system

critical files. These hashes are bounded to SNMP custom Object Identifier for reporting.

- Nagios Server: An open-source monitoring tool, is configured to graphically display system integrity Reports for system administrator.

### C. Target System Configuration

The TPM enabled system measures the integrity of selected binaries, executable, and libraries and stores it in Platform Configuration Registers (PCRs). The PCR_EXTEND command updates a PCR by appending a new calculated hash value. The PCR_QUOTE command signs the value using Attestation Identity Key (AIK) [44], allowing verification on the challenger side using AIK public key. This aggregate value is then associated with reserved OID through Agent MIB to be matched at the challenger side for verification. A timestamp and nonce ensures response validity and prevent reply attacks.

- Hash-OID Binding Module: A python-based function 1 is designed that maps TPM generated integrity values to designated SNMP-OID. This allows the integrity status to be queried remotely from challenger without impacting the protocol performance or altering its core functionality.

### D. Challenger Side

This section details challenger system and its configuration, and custom-developed functions. The initial integrity status of each registered node is in hashes database as trusted hashes or signatures. Incoming new hashes from clients are matched and verified against these trusted hashes to determine system integrity.

- Hash Appraisal Function: A Python module validates the integrity status by comparing the incoming hash from clients against stored hashes the database. A match confirms the target's integrity while a mismatch indicates potential compromise. The function then forwards the result to SNMP Sub-Manager.

### E. Nagios Plugin for Monitoring

Nagios is an open-source network monitoring tool that oversees overall network. It provides real-time alerts on integrity violation or system failures allowing administrator to respond promptly [45–49]. In this research a plugin is developed to retrieve and visualize client integrity status using SNMP. The plugin interact with SNMP Sub-Manager within Nagios core. Ensuring continuous monitoring and timely detection of integrity breaches.

## VIII. Implementation and Experimental Results

This section is structured into two parts: The first outlines the operating system setup and configuration, including network environment setting, installation and configuration of net-SNMP on both server and client sides. The second part presents the testing output obtained during implementation validating the proposed architecture's effectiveness.

### A. Hardware and Software Stack

The proposed architecture is deployed on Linux due to its open-source nature, allowing kernel level modification to meet specific requirements. TPM2.0 tools provide command line utilities for to fetch PCR values, Net-SNMP is deployed to extend and enhance SNMP functionalities for integrity verification. The hardware and software supporting this implementation is detailed in Table 2.

Table 2. Hardware and software stack specification for the proposed research

| S.No | Component | Role and Specification | |
|---|---|---|---|
| 1 | Processor | Server - Intel Core i7 (8-Core, 3.0 GHz ), Client - Intel Core i5 (4-Core, 2.5 GHz) | Hardware Specifications |
| 2 | Memory (RAM) | 16 GB (Server), 8 GB (Client) | |
| 3 | Storage | 512 GB SSD (Server), 256 GB SSD (Client) | |
| 4 | TPM Chip | TPM 2.0 (Hardware-based), computes PCR values for integrity verification. | |
| 5 | Network Bandwidth | 1 Gbps Ethernet. | |
| 1 | CentOS 8.0 | Linux OS with TPM 2.0 support. | Software Stack |
| 2 | TPM 2.0 Tools | Command-line utilities used for fetching PCR values for integrity verification. | |
| 3 | net-SNMP 5.8 | Implements SNMP with support for custom MIB extensions. | |
| 4 | Nagios 4.3 | Network monitoring tool with a custom plugin. | |
| 5 | Python 3.11 | Programming language for Hash-Appraisal and SNMP automation scripts. | |

### B. Implementation Steps

The implementation phase follows a structured approach to ensure secure, lightweight, and scalable integrity verification for distributed systems. The process involves setting up the necessary environment, configuring required services, and ensuring secure and reliable communication between the entities. The following sections provide a detailed breakdown of the implementation architecture.

### C. Configuration of SNMP

Net-SNMP is an open-source network management suit that provides essential utilities for management and monitoring of network devices. It is freely available and can be downloaded from its official repository at SourceForge.net.

- Creating SNMP V3 users: Users must be defined for authentication in SNMPv3. This is done using the; net-snmp-create-v3-user command, which configures users with AES encryption and SHA authorization in compliance with SNMPv3 policies.

  Note: SSL is enabled for secure communication.
- Verifying SNMPv3 Security: To verify SNMPv3 security, some dummy requests are sent to the client, and its reply is captured. These packets are analyzed and proved encrypted using the network monitoring tool "wire-shark".
- Extending SNMP: SNMP is extended using SNMP MIB (Management Information Base) modules. MIB modules define the objects and variables that can be accessed using SNMP. The author created an OID in agent's MIB and then assigned TPM generated hash value with this OId, Heo *et al.* [47] have also utilized Extended-SNMP to manage digital convergence devices. A new Object ID (1.3.6.1.4.1.61096) is reserved from registration authority https://www.iana.org. The MIB file is edited and compiled to assign the reserved OID. snmpd service is restarted to update the MIB file. PCR Quote value is

associated with this extended reserved object (OID) which can be requested from the challenger side.

### D. Hash-OID Binding Function

Algorithm 1 outlines the binding process between TPM-generated hash measurements with custom SNMP Object ID (1.3.6.1.4.1.61096). These integrity measurements are later requested by the server and matched against the referenced hashes stored in the server side database. In Line 1, the algorithm imports the necessary Operating System and SNMP libraries to enable communication with the Operating System and SNMP configuration files. The Get() function retrieve the TPM generated hashes from Platform Configuration Registers and present it to the main function. In Step 6, the hash is assigned to a variable, which is then bound with SNMP OID in Steps 7 and 8, making it accessible to remote integrity queries.

Mathematical Eq. (1) formally defines the module, where OIDbound represents the customized OID within the extended MIB, mapped to integrity measurements.

Hagg denotes the aggregate hash value obtained from PCR. OIDreserved refers to the predefined OID allocated storing integrity-related measurements. The function $f(\cdot)$ defines the binding operation that associates the computed aggregated hash with the designated OID, ensuring seamless integration within the SNMP framework.

$$0ID_{bound} = f(H_{agg}.OID_{reserved}) \quad (1)$$

**Algorithm 1. Hash OID binding Function**

```
1 Input  PCR quote generated by TPM at agent side.
2 Output Bind and handover the hash to snmp.
3 Import: os.sys, net.snmp;                    /*imports os and
snmp */
4 Function: hash-quote ();    /*user function*/
5  Get: pcr_quote;                    /*getting pcr value from
agent*/
6  Read: pcr_quote;    /*open the quote*/
7  Return: quote;    /*return the hash quote*/
8 var ← hash ();                    /*assign the hash to a variable*/
9 Set: oid ← var;        /*Bind variable with Oid*/
10 Get: snmpPDU (hash);            /*snmp command for server
side*/
```

### E. Hash Appraisal Function

The preceding section described the module responsible to retrieve the aggregate value from target system to the challenger. To evaluate whether the integrity of a given client is upheld or compromised, the hash appraisal function perform a comparison between recently retrieved hash and the referenced hash stored in the server side database.

As defined in Eq. (2), I denotes the Integrity status, where $I = 1$ = indicates the integrity is maintained, and $I = 0$ signifies a compromised state. Hreceived represents the hash value received from the client, while Hstored refers to the pre-stored reference hash in the database, used for integrity verification.

$$I = 1, \text{ if } H_{received} = H_{stored} \quad (2)$$

$$I = 0, \text{ otherwise}$$

### F. Flow of the Protocol

The complete protocol workflow is depicted in Fig. 3, illustrating the interaction between challenger and target system. The challenger initiate an SNMP request to verify The integrity of a client by communicating with the SNMP agent through the SNMP Manager. The agent processes requests only from authenticated users, verifying its credentials, security level, and encryption setting as per SNMP v3 configuration. The TPM generated aggregate value extended from Platform Configuration Registers (PCRs). This value is then associated with custom reserved SNMP OID through the hash-Oid binding function. On receiver side hash-appraisal function match this received hash with already stored reference hashes in the database on the server side. Based on this comparison, a True/False Boolean result is returned. Finally, the Nagios server as shown in Fig. 4, graphically display the real time integrity status across the network, providing administrators with centralized visibility and alerting capabilities.

**Algorithm 2. Algorithm for Hash Appraisal Function**

```
1 Input Gets two hashes in the form of text file
2 Output Returns “matched” or “Un-matched”
3 Import: os.sys, net.snmp ;              /*imports os and snmp*/
4  fopen(stored_hashes) ;              /*reading good hashes */
5  while(eof) ;   /*repeat till end*/
6  explode(stored_hash);/*read complete file*/
7  file1←stored_hash[] ;              /*assign hash to var */
8  End loop ; /*loop completed*/
9  fclose(stored_hash) ;  /*closing file*/
10     fopen(fetched_hash) ;          /*opens newly fetched hash
by snmp*/
11 while(eof) ;              /*repeat reading the til end */
12 explode(fetched_hash);   /*read fetched hash*/
13 file2←fetched_hash[] ;       /*assign hash to var*/
14 End loop ; /*loop completed*/
15 fclose(fetched_hash) ;/*close file*/
16 if file1[] == file2[] ;          /*appraise both files*/
17 Display “Hahses Matched” ;       /*Files are same*/
18 Display “Hahses Not Matched”;          /*Files are different*/
```

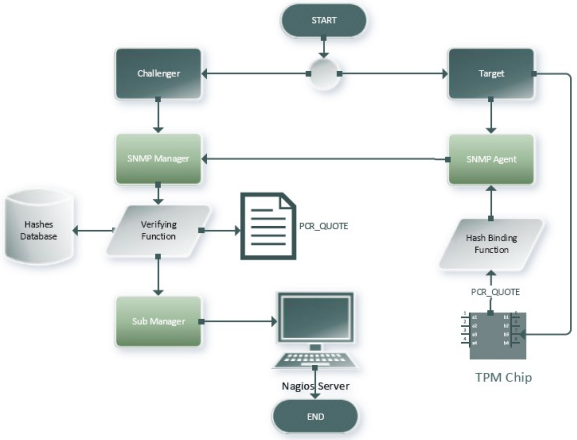


Fig. 3. Flow of the protocol.

### G. Configuring Display Interface

The final stage of the proposed architecture involved presenting the client integrity reports in a graphical user interface. As previously discussed, Nagios has been selected as the monitoring and display interface for the system

administrator. This section outlines the steps taken to configure the Nagios plugin for integrity reporting.

### H. Nagios Core Configuration

Nagios core version 4.3.1 was selected as a stable and reliable monitoring platform, to enable full functionality, the NRPE plugin (Nagios Remote Plugin Executor) and all supporting packages were installed. These included a C-compiler, C standard Library, development tools, SSL libraries and extended internet service daemon (xinetd). These components ensure that Nagios core operates securely with client system to collect and display real time integrity status.


Fig. 4. Client integrity status.

- To specify which systems Nagios should monitor, the configuration file nagios.cfg must be configured. The path for the configuration files and the host definition configuration file is defined as: cfg_dir=/etc/nagios/servers.
- NRPE Configuration Nagios Remote Plugin Executor (NRPE) is installed on the client side to collect system level metrics and transmit them to Nagios Core on the server. NRPE use TCP port 5666 and it must be open to allow connections from server via allowed_hosts=SERVER_IP in nrpe.cfg.
- Integrity check Plugin Nagios being open-source and modular allows for extensive customization. A custom plugin was developed in python and integrated in Nagios. This plugin queries the hash-appraisal function and get an appropriate Nagios exit code (0 for OK, 1 for WARNING, 2 for CRITICAL, and 3 for UNKNOWN).

Once configuration is completed, save the files and restart the NRPE and Nagios core services.

- After refreshing Nagios Core, the integrity monitoring result for the DNS server is displayed in Fig. 4. The top-level message under the group, "Host Integrity Status Information", is "Verified", indicating that no integrity violations has been detected for the corresponding client system.
- If the integrity of the client is compromised, Nagios will display "Compromised" in place of "Verified". This status serves as an alert, indicating that the client measured hash value do not match the trusted value stored in the server database.

## IX. Results

The proposed architecture is evaluated from both security and performance perspectives. As is often the case in secure systems, enhancing security mechanism can introduce a degree of performance overhead. Since features such as cryptographic operations, hash binding and comparison consumes additional CPU cycles, memory and network bandwidth. Conversely, optimizing for performance by reducing these measures may reduce security posture of the system. Therefore, a trade-off must be made between security and performance based on the specific needs and requirements of the system.

### A. Performance with Integrity

The integration of integrity verification into SNMP framework introduces minimal impact on overall performance. However, there is a computational cost associated with increased security. During initial stages of system booth, when only a limited number of applications are active, the response time remains fast, as the number of active application increases, the integrity measurement process becomes more time-consuming, leading to increased response latency as illustrated Fig. 5. during initial measurements, both standard SNMP and the proposed Extended SNMP responded in comparable latency, however in the later half of the experiment when a bulk of requests were queried, a marginal delay if approximately 50–60 ms was observed in Extended-SNMP. This difference is considered negligible and may be attributed to transient network congestion and increased system load. To test the impact on performance, snmpget and snmpget-next requests for the integrity-specific OID (1.3.6.1.4.1.61096) commands in net-snmp are used, and the latency is noted.

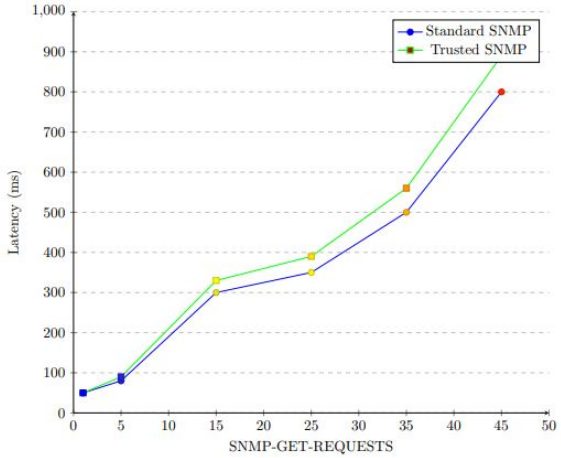


Fig. 5. Latency graph shows 0.5% packet loss.

While the impact on performance remains small under normal network conditions, it could become more pronounced in environments characterized by high packet loss, network congestion, or low throughput. The effect of bulk SNMP requests on latency showing that bulk operations introduce additional delays compared to individual queries. This behavior is expected due to the increased processing and transmission load associated with bulk data transfers.

### B. Performance with Frequency

The frequency of requests for integrity status can significantly impact the amount of network traffic generated over time. A high frequency polling rate–such as querying the integrity status every minute–can introduce significant traffic overhead. Especially in bandwidth-constrained or high

utilization network environment. Conversely, A low frequency polling strategy, such as querying once per hour, considerably reduce traffic but may compromise the system responsiveness to integrity violation.

Therefore, selecting an appropriate query interval involve a trade-off between detection timeline and communication overhead, considering the specific requirements and constraints of the network being monitored. This scenario is illustrated in Fig. 6.

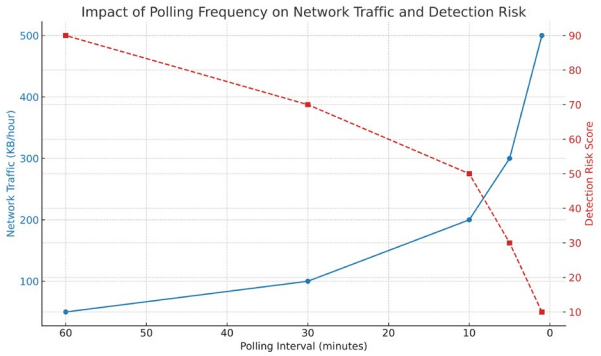


Fig. 6. Frequency graph.

### C. Security

The proposed architecture enhances SNMP by introducing integrity measurement capabilities without altering its core security mechanisms. SNMPv3 is selected due to its robust support for authentication, privacy and message integrity, making it a suitable base for building a trusted network monitoring solution.

- Confidentiality: The security subsystem in this architecture uses the SNMPv3 module, which ensures complete confidentiality employing hashing techniques and Transport Layer Security (TLS) encryption to protect data in transit. It is worth noting that no modifications have been made to the security subsystem of SNMP during this study. Therefore, the architecture ensures complete confidentiality of data at both the system and network levels.
- Integrity: This research's primary focus and contribution is the implementation of integrity verification through the utilization of the TPM chip by the Extended SNMP module. By leveraging this technology, any alteration to even a single bit of a file can result in a complete hash change, thereby enabling comprehensive integrity checks.
- Availability: The proposed architecture has been designed in such a way that it does not affect the availability of the overall network. The integrity information is made available, along with other network status information, through SNMP on a functioning network.

## X. Conclusion

This research extends the capabilities of SNMP by integrating TPM based integrity verification alongside traditional system monitoring. The MIB of the SNMP is extended to associate the hash of concerned files from PCRs. This hash is assigned to a reserved SNMP OID and sent to the challenger, where this value is compared with some known good and trusted hashes of the same files. The integrity of these systems is maintained by comparing hashes. A Nagios plugin is configured to periodically report the integrity status to the system administrator regarding targeted clients. The research focuses on integrity verification based on TPM. However, not all IP-based nodes have TPM chips installed, which means that the integrity of those machines cannot be measured without TPM. Therefore, the architecture proposed in the research is limited to devices with installed TPM chips.

This design can be further improved by incorporating machine learning techniques to automate real-time detection and response to integrity attacks. Machine learning modules can enhance decision-making by providing insights into the root cause of integrity breaches and suggesting appropriate mitigation measures.

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

MA analysed the data, wrote and reviewed the paper, conducted the research and experiments; FW supervised, reviewed and proofread the article; AS analysed the data, wrote and reviewed the paper; MK conducted the research and experiments; MIS proofread the article. All authors had approved the final version.

## References

[1] M. Riek and R. Böhme, "The costs of consumer-facing cybercrime: An empirical exploration of measurement issues and estimates," *Journal of Cybersecurity*, vol. 4, 2018.

[2] T. Ali, J. Ali, T. Ali, M. Nauman, and S. Musa, "Efficient, scalable and privacy preserving application attestation in a multi stakeholder scenario," in *Proc. International Conference on Computational Science and Its Applications*, Springer, 2016, pp. 407–421.

[3] W. Charoenwet, P. Thongtanunam, V.-T. Pham, and C. Treude, "Toward effective secure code reviews: An empirical study of security-related coding weaknesses," arXiv preprint, arXiv:2311.16396, 2024.

[4] S. Semenov, V. Davydov, N. Kuchuk, and I. Petrovskaya, "Software security threat research," in *Proc. 2021 XXXI International Scientific Symposium Metrology and Metrology Assurance (MMA)*, 2021, pp. 1–4.

[5] S. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics*, vol. 12, no. 6, 2023.

[6] P. Sun, Y. Wan, Z. Wu, and Z. Fang, "A survey on security issues in IoT operating systems," *Journal of Network and Computer Applications*, 103976, 2024.

[7] O. Demigha and R. Larguet, "Hardware-based solutions for trusted cloud computing," *Computers & Security*, vol. 103, 2021, 102117.

[8] Trusted Computing Group. TC trusted computing. [Online]. Available: http://www.trustedcomputinggroup.org/

[9] Y. Adıgüzel and S. B. Yalçın, "Secure boot design for a RISC-V based SoC and implementation on an FPGA," in *Proc. 2024 32nd Signal Processing and Communications Applications Conference (SIU)*, 2024, pp. 1–4.

[10] T. Morris, "Trusted platform module," in *Encyclopedia of Cryptography, Security and Privacy*, Springer, 2024, pp. 1–5.

[11] A. Muñoz, A. Farao, J. R. C. Correia, and C. Xenakis, "ICITPM: Integrity validation of software in iterative continuous integration through the use of Trusted Platform Module (TPM)," in *Proc. Computer Security: ESORICS 2020 International Workshops*, Springer, 2020, pp. 147–165.

[12] S. Tyata and A. Barsoum, "Network management protocols: Analytical study and future research directions," *Journal of Network and Information Security*, vol. 9, no. 2, pp. 9–13, 2021.

[13] J. Liu, C. Qu, and T. Zhou, "A novel cloud computing platform monitoring system based on Nagios," in *Proc. 2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*, IEEE, 2023, pp. 169–172.

[14] Y.-C. Tian and J. Gao, "Network management architecture," in *Network Analysis and Architecture*, Springer, 2023, pp. 321–359.
[15] W. Stalling. SNMP architecture. [Online]. Available: http://www.net-snmp.org/docs/mibs/
[16] E. Gamess and S. Hernandez, "Performance evaluation of SNMPv1/2c/3 using different security models on Raspberry Pi," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, issue 11, 2021. doi: 10.14569/IJACSA.2021.0121101
[17] J. Schönwälder and V. Marinov, "On the impact of security protocols on the performance of SNMP," *IEEE Transactions on Network and Service Management*, vol. 8, no. 1, pp. 52–64, 2011.
[18] R. Jatothu and G. Narasimha, "Enhancement in SNMP services with improved security with the impact of SSH, TLS and DTLS protocols," in *Proc. 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, 2017, pp. 888–895.
[19] IETF, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," Tech. Rep. RFC 3414, Dec. 2002.
[20] H. Handschuh, "SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)," in *Encyclopedia of Cryptography, Security and Privacy*, Springer, 2024, pp. 1–5.
[21] IETF, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)," Tech. Rep. RFC 3415, Dec. 2002.
[22] P. C. Zatko and D. Rizzo, "Trusted computing," U.S. Patent 9,569,638, Feb. 14, 2017.
[23] S. Hosseinzadeh, B. Sequeiros, P. R. Inácio, and V. Leppänen, "Recent trends in applying TPM to cloud computing," *Security and Privacy*, vol. 3, no. 1, 2020.
[24] S. Wesemeyer, C. J. Newton, H. Treharne, L. Chen, R. Sasse, and J. Whitefield, "Formal analysis and implementation of a TPM 2.0-based direct anonymous attestation scheme," in *Proc. the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 784–798.
[25] M. Alam, T. Ali, S. Khan, S. Khan, M. Ali, M. Nauman, A. Hayat, M. Khurram Khan, and K. Alghathbar, "Analysis of existing remote attestation techniques," *Security and Communication Networks*, vol. 5, no. 9, pp. 1062–1082, 2012.
[26] R. A. Popa, "Confidential computing or cryptographic computing?" *Communications of the ACM*, vol. 67, no. 12, pp. 44–51, 2024.
[27] W. Arthur, D. Challener, and K. Goldman, *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*, Springer Nature, 2015.
[28] Trusted Computing Platform. (2020). TPM profile specification for TPM 2.0. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG-PC-Client-Platform-TPM-Profile-for-TPM-2.0-Version-1.06-Revision-32_5April24.pdf
[29] P. G. Wagner, P. Birnstill, and J. Beyerer, "Establishing secure communication channels using remote attestation with TPM 2.0," in *Proc. the Conference Security and Trust Management (STM 2020)*, 2020.
[30] N. Ashraf, A. Masood, H. Abbas, R. Latif, and N. Shafqat, "Analytical study of hardware-rooted security standards and their implementation techniques in mobile," *Telecommunication Systems*, vol. 74, pp. 379–403, 2020.
[31] A. A. Kuznetsov, O. V. Potii, N. A. Poluyanenko, Y. I. Gorbenko, and N. Kryvinska, "Comparative analysis of determined generators for random bits, defined in the NIST special publication 800-90A," in *Stream Ciphers in Modern Real-time IT Systems: Analysis, Design and Comparative Studies*, 2022, pp. 165–179.
[32] K. Shah, A. Bhadauria, P. Thakkar, J. Shah, and H. Kaur, "Advancements in elliptic curve cryptography: A review of theory and applications," in *Proc. the 2024 Parul International Conference on Engineering and Technology (PICET)*, 2024, pp. 1–6.
[33] F. Bohling, T. Mueller, M. Eckel, and J. Lindemann, "Subverting Linux integrity measurement architecture," in *Proc. the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–10.
[34] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proc. 13th USENIX Security Symposium*, 2014, vol. 13, pp. 223–238.
[35] T. Jaeger, R. Sailer, and U. Shankar, "PRIMA: Policy-reduced integrity measurement architecture," in *Proc. the Eleventh ACM Symposium on Access Control Models and Technologies*, 2006, pp. 19–28.
[36] R. Du, W. Pan, and J. Tian, "Dynamic integrity measurement model based on vTPM," *China Communications*, vol. 15, no. 2, pp. 88–99, 2018.
[37] M. Faisal, I. Ali, M. S. Khan, S. M. Kim, and J. Kim, "Establishment of trust in Internet of Things by integrating trusted platform module: To counter cybersecurity challenges," *Complexity*, 2020.
[38] D. Lu, R. Han, Y. Wang, X. Dong, X. Ma, T. Li, and J. Ma, "A secured TPM integration scheme towards smart embedded system based collaboration network," *Computers & Security*, vol. 97, 101922, 2020.
[39] B. Xing, Z. Han, X. Chang, and J. Liu, "OB-IMA: Out-of-the-box integrity measurement approach for guest virtual machines," *Concurrency and Computation: Practice and Experience*, 2014.
[40] S. Thom, R. Aigner, M. Kapadia, S. H. Schaefer, and R. K. Spiger, "Utilizing a Trusted Platform Module (TPM) of a host device," U.S. Patent 10,212,156, Feb. 19, 2019.
[41] P. Matoušek, O. Ryšavý, and L. Polčák, "Unified SNMP interface for IoT monitoring," in *Proc. the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2021, pp. 938–943.
[42] M.-S. Kim, D. P., J. Yu, and H. Lee, "Traffic flooding attack detection with SNMP MIB using SVM," *Computer Communications*, 2008.
[43] L. X. N. Wang, "Network operation administration and maintenance (OAM) method, apparatus, and system," U.S. Patent 10,237,124, Mar. 2019.
[44] L. Chen and M. Young, *Trusted Systems: Second International Conference, INTRUST 2010, Beijing, China, December 13-15, 2010, Revised Selected Papers*, Springer Science & Business Media, 2011.
[45] G. Katsaros, R. Kübert, and G. Gallizo, "Building a service-oriented monitoring framework with REST and Nagios," in *Proc. the 2011 IEEE International Conference on Services Computing (SCC)*, IEEE, 2011, pp. 426–431.
[46] F. Wahab, S. Ma, X. Liu, Y. Zhao, A. Shah, and B. Ali, "A ranked filter-based three-way clustering strategy for intrusion detection in highly secure IoT networks," *Computers and Electrical Engineering*, vol. 127, 110514, 2025.
[47] G. Heo, E. Kim, and J. Choi, "An extended SNMP-based management of digital convergence devices," in *Proc. the 2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*, IEEE, 2010, pp. 2540–2547.
[48] F. Wahab, S. Ma, Y. Zhao, and A. Shah, "An explainable three-way neural network approach for intrusion detection in IoT ecosystem," *Internet of Things*, 101722, 2025.
[49] A. Shah, N. Azam, B. Ali, M. T. Khan, and J. Yao, "A three-way clustering approach for novelty detection," *Information Sciences*, vol. 569, pp. 650–668, 2021.