Secure Management of Hyperscale Cloud Network Accelerators

Faye Yang
Senior Hardware Systems Engineer
Microsoft
fayeyang@microsoft.com

Xiling Sun
Principal Firmware Engineer
Microsoft
xilingsun@microsoft.com

***Abstract*—**Ensuring cloud computing security and robustness is increasingly vital for all stakeholders in the digital age, particularly against the backdrop of escalating cyber threats. Network accelerators are extensively utilized to enhance network performance within Hyperscale cloud infrastructure. The cloud server platforms housing the network accelerators serve as the backbone and simultaneously become the appealing targets for attackers aiming to compromise the server. Creating a secure and resilient management platform for network accelerators is fundamental to ensuring cloud security from the ground up. This paper presents an innovative approach to designing such a platform tailored for 400Gb network and storage accelerators and beyond. The design incorporates layered security strategies, including enhancing hardware security at device level, strengthening platform integrity, and fortifying remote access protocols with strong certificate-based authentication. It also proposes an advanced hardware interface design, which serves as the physical foundation for secure communication. These measures work together to protect the network accelerator platform from attacks and build trust with users and customers by providing reliable cloud infrastructure.

Keywords—Network Accelerator Cards (NACs), Baseboard Management Controllers (BMCs), hardware security, platform integrity, advanced interfaces

I. INTRODUCTION

Network Accelerator Cards (NACs) and Baseboard Management Controllers (BMCs) are crucial components for enhancing network performance and managing cloud server platforms within hyperscale cloud infrastructure effectively. As illustrated in Fig.1, cloud server platform relies heavily on BMCs for out-of-band (OOB) managing server hardware, including CPUs and NACs. BMCs employ unique privileged hardware interfaces to interact with various devices and peripherals on the platform, enabling remote monitoring, system recovery, and firmware updates among other functions. Due to their critical role, BMCs are prime targets for security vulnerabilities, which can result in unauthorized access and control over the whole server platforms. Such vulnerabilities can go undetected and unresolved for years, as highlighted by the "Pantsdown" [1] exposure in 2019. Remote access makes BMCs appealing targets for orchestrating large-scale attacks. Likewise, Network Acceleration Cards (NACs), which enhance the speed of network traffic processing and transmission, are often the focus of attackers looking to intercept or alter data streams. Current devices, interfaces, and communication protocols within NAC platform management frequently fail to address these security issues adequately.

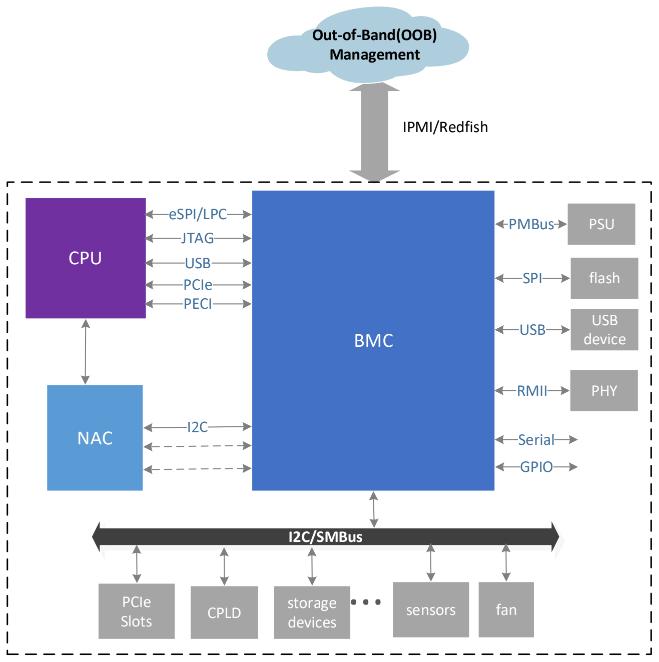

Fig. 1. High-level cloud server platform architecture

This paper presents an innovative approach that combines layered security strategies and sophisticated hardware interfaces to create a robust and efficient NAC management system. In line with Microsoft's Secure Future Initiative [2], we incorporate security into the design stage to enhance the resilience of the platform within a hyperscale cloud environment. This approach not only tackles existing vulnerabilities but also equips cloud server infrastructure for upcoming security threats.

II. LAYERED SECURITY STRATEGIES

This section dives into the three essential aspects of our layered security strategies: Firstly, enforcing hardware and firmware security at the device level by integrating Root-of-Trust (RoT) subsystem into device's System-on-Chip (SoC) and establishing a trusted execution environment (TEE) in the SoC application subsystem. Secondly, strengthening platform integrity through attestation with platform RoT. Thirdly, securing remote access with robust certificate-based authentication.

A. Enforcing hardware and firmware security at device SoC level: RoT security subsystem integration and TEE establishment.

Modern BMC and NAC SoC usually contain application processors, high-speed interconnect fabric, RAM, peripherals and corresponding firmware stack that form the application subsystems responsible for primary operations. To build highly secure devices, it is crucial to have a dedicated security subsystem that remains physically separated from the application domain to handle security-critical functions. Fig. 2 shows a simplified model of a BMC or NAC System-on-Chip (SoC) with an integrated RoT security subsystem.

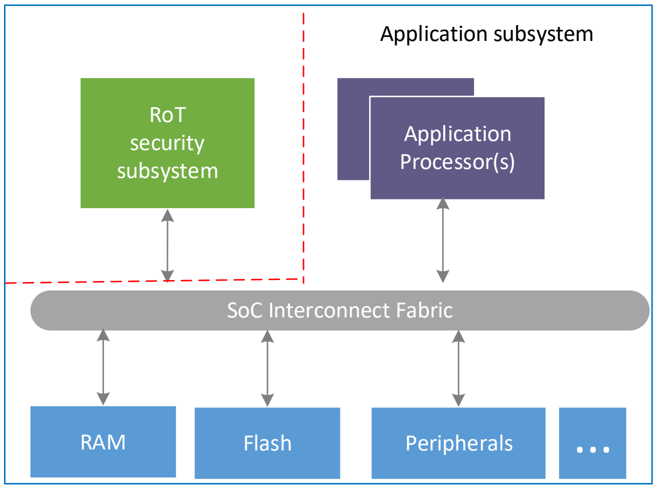

Fig. 2. Simplified model of SoC with RoT security subsystem

The RoT security subsystem typically consists of key security hardware building blocks and firmware stack, as illustrated in Fig. 3. At the center there is a dedicated security processor performing critical security operations. It also includes cryptographic engines such as a True Random Number Generator (TRNG), AES, SHA, and ECDSA to facilitate modern cryptographic algorithms. The secure interconnect fabric safeguards data transfers within SoC. Additionally, it features one-time-programmable (OTP) memory such as eFuses to save crucial unchangeable configuration settings, encryption keys, security polices etc.

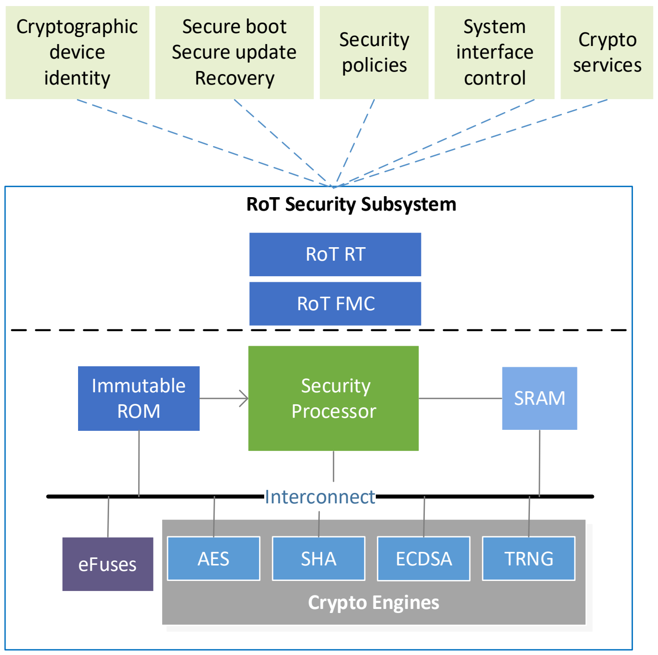

Fig. 3. RoT security subsystem architectural view

RoT firmware stack running on security processors acts as the brain for handling security-critical operations. It generally comprises at least three layers: immutable ROM, RoT First-Mutable-Code (FMC) and RoT Run-time (RT). The immutable ROM contains hardwired logic and minimal bootstrapping code that is fixed at the time of System on Chip (SoC) manufacturing. This immutable nature ensures that the code cannot be altered, providing a secure foundation for the boot process. We will discuss FMC and RT shortly.

Below are the vital roles that RoT firmware performs to ensure the security of SoC.

1) Generates and protects an unforgeable cryptographic device identity embedded in hardware. Hardware cryptographic device identity is significant for authenticating the device and ensuring secure communication with other devices or systems. It forms the basis of supply chain security via securely harvesting and provisioning the device identity. Device Identifier Composition Engine (DICE) [3], as defined by the Trusted Computing Group (TCG), is a widely adopted model that combines silicon characteristics with software techniques to produce such a unique device identity.

2) Compliant with NIST 800-193 platform firmware resiliency guidelines.[4]

Secure boot: a mechanism that ensures the integrity of every piece of code being loaded before it is allowed to execute by verifying the firmware's digital signature using a securely anchored root of trust public key, maintaining a chain of trust throughout the boot process, and revoking previously signed firmware if it is found to be compromised.

Secure update: a mechanism that ensures that firmware updates are authenticated and authorized, with the firmware image signed by a key cryptographically bound to the device's immutable root key. The signature is verified by RoT before programming non-volatile storage, providing rollback protection and integrity protection to prevent unauthorized modifications.

Secure Recovery: a mechanism that ensures a device can return to its normal operational state by obtaining operational images from a trustworthy source when corruption, tampering or an unknown state is detected. The recovery image must be authenticated by RoT before being programmed to non-volatile storage.

3) Manage system configurations, security policies, peripheral and interface control and hardware lifecycle management. For instance, it can limit debugging to authorized personnel, secure GPIO pins, set up secure boot options, and handle memory partitioning with access controls to protect data. This is especially important for mitigating risks linked to permissive interfaces on BMCs.

4) Provide cryptographic services for the application domain e.g. signing request, ensuring crypto keys remain secure and are never exposed to the application. This will be further discussed in later session.

One approach to implement the RoT security subsystem on an SoC is to adapt the open-source Project Cerberus [5] RoT solution to construct the RoT firmware stack on SoCs equipped with all the mentioned security hardware components. Project Cerberus provides a strong, platform-agnostic implementation for TCG DICE, covering firmware image validation, key manifest management, secure firmware updates, attestation, flash management, and more.

The main challenge with this approach lies in the substantial investment needed from silicon vendors to develop and implement security hardware components within an SoC from the ground up. Additionally, the significant effort required to port and integrate Cerberus RoT onto a particular SoC presents a challenge for scalability across multiple diverse SoCs.

To tackle these issues, *we strongly recommend the integration of the industry-standard open-source RoT solution Caliptra into BMC and NAC SoCs as security subsystem during design phase*. Caliptra [6] is fully open source, providing transparency from the hardware RTL to the firmware stack. This initiative and community are progressing steadily, enabling state-of-the-art features such as streaming boot, post-quantum cryptography (PQC) support.

Another device-level security strategy is to establish a Trusted Execution Environment (TEE) within the SoC application subsystem. This can be achieved by utilizing hardware-based security features like Arm® TrustZone® [7] with open-source ARM trusted firmware stack to create a secure, isolated environment for running trusted applications (TAs) separately from the main operating system and other applications. One popular TA is the firmware-based Trusted Platform Module (fTPM) that implements TPM functionality within the TEE. An open-source reference implementation for fTPM [8], along with integration examples, is available. This method is ideal for both BMC and NAC SoC application subsystems to execute sensitive tasks like cryptographic key management, secure storage, attestation measurements, and user authentication.

B. *Strengthening platform integrity through attestation with platform RoT.*

Platform RoT is a foundational security component that ensures the integrity and authenticity of the cloud server platform. It is powered on before any other system components and needs to be independent and tolerant of resets, enabling updates and resets without impacting the platform.

A key aspect of maintaining platform integrity is having the platform RoT verify all devices during both boot and runtime, regularly attesting to all devices in the system. Fig. 4 shows the simplified platform attestation workflow in the cloud infrastructure. A device subject to attestation by the platform RoT is referred to as an Active Component Root-of-Trust (AC-RoT). For each AC-RoT enabled on the platform, the Platform RoT performs the following steps:

1) Validate the identity: platform RoT validates the device identity of the AC-RoT using the device's identity certificate chain.

2) Authentication challenge: it confirms that each device can use the private authentication key linked to the verified certificate chain by issuing a challenge to each device.

3) Measurement verification: platform RoT extracts the digitally signed measurements reported by the AC-RoT and verifies them against reference measurements.

4) Acceptance or remediation: based on the verification results, the Platform RoT either accepts the device or initiates a remedial action.

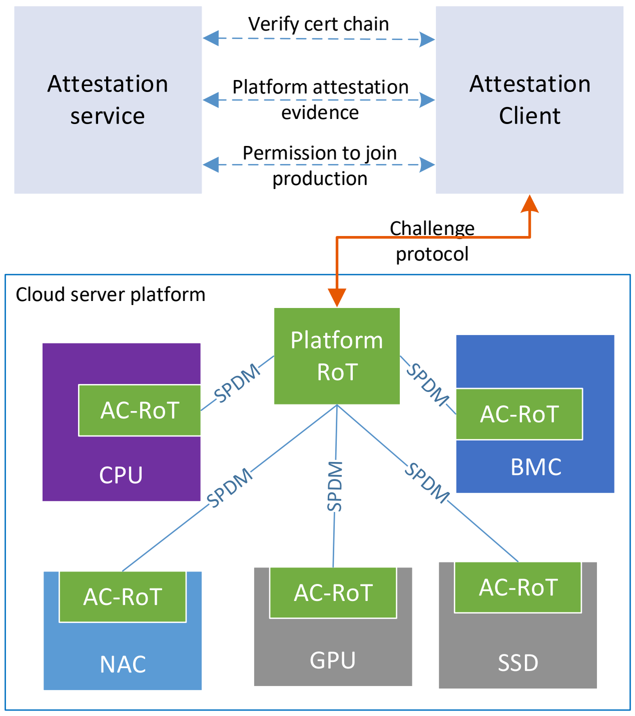

Fig. 4. Simplified platform attestation workflow

Platform RoT supports multiple attestation protocols to enable the remote attestation of the AC-RoTs, such as Project Cerberus Challenge protocol [9] and the DMTF's Secure Protocol and Data Model (SPDM) protocol [10] Upon successful verification, the Platform RoT securely conveys platform attestation evidence to the Data Center Management plane. This evidence is used to permit server node entry into the production environment, ensuring compliance with established security policies and thereby enhancing the overall integrity of the platform.

C. *Securing remote access protocols with robust certificate-based authentication*

OpenSSH and Redfish are crucial remote access protocols for cloud server platforms in manufacturing setups and for out-of-band (OOB) management across fleets. However, SSH public key-based authentication presents several limitations. The main issues include scalability challenges in managing a large number of keys and connections. Key management becomes complicated due to the difficulties in distributing, rotating, and revoking keys, along with the need to secure numerous keys, which introduces vulnerabilities. Moreover, the Trust-On-First-Use (TOFU) model is insecure as it establishes trust on the initial connection, increasing risks if the first key is compromised. Enhancing these protocols with certificate-based authentication substantially improves security and efficiency. Employing certificates issued by a trusted Certificate Authority (CA) mitigates the vulnerabilities associated with password-based authentication and reduces the risk of unauthorized access.

We developed a prototype enabling OpenSSH certificate-based authentication backed by BMC RoT subsystem, providing an additional layer of security to the standard process. The integrated architecture is shown in Fig. 5. The highlight is the isolation of the private key within the RoT subsystem, where the signing request for key exchange hash during OpenSSH session establishment is routed to the hardware RoT subsystem via the PKCS#11 provider library. In this case, RoT subsystem is the cryptographic service provider to application subsystem mentioned earlier. Furthermore, the SSH certificate is anchored to an organizationally trusted CA. When certificate authentication is configured on both the client and server, it allows for password-less login, effectively addressing the security risks related to the username/password logins to BMC and TOFU model.

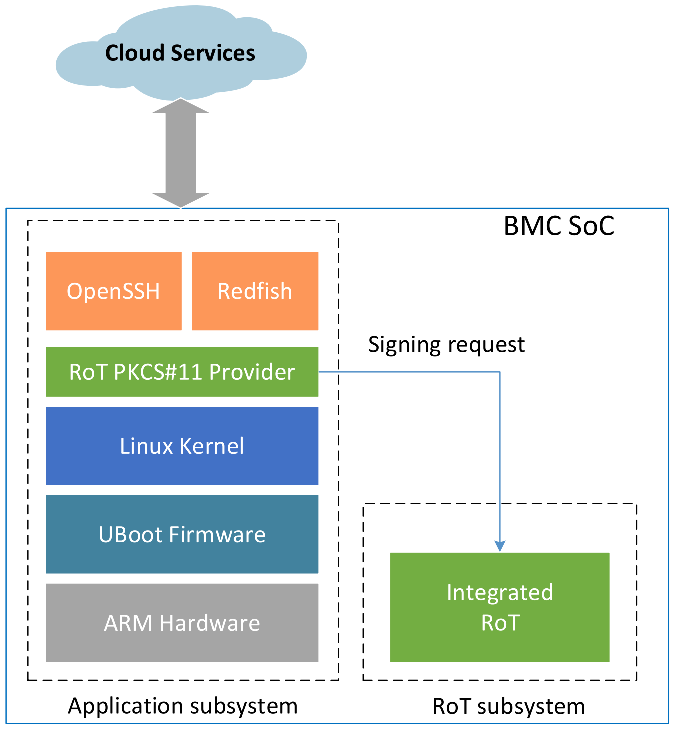

Fig. 5. Integrated architecture of certificate-based auth

The prototype is scalable and extendable to NAC and other devices with RoT subsystem integrated. Enabling certificate-based authentication for OpenSSH and Redfish enhances the security of remote access to the systems, bolstering their defense against potential threats.

III. ADVANCED HARDWARE INTERFACES DESIGN

In current modern server design, BMC is usually physical on the DC-SCM (standards-ready secure control module) plug-in to server motherboard as shown in Fig. 6. NAC usually connects to the server motherboard via a PCIe slot. There are interfaces like UART/I2C/USB between BMC and NAC for dedicated remote control, system monitoring, firmware updates and recovery. BMC also has the 1G ethernet PHY connecting to management switches to the rack manager, which can push NAC's telemetry data to relevant databases and support out of band control capabilities to corresponding APIs or CLIs.

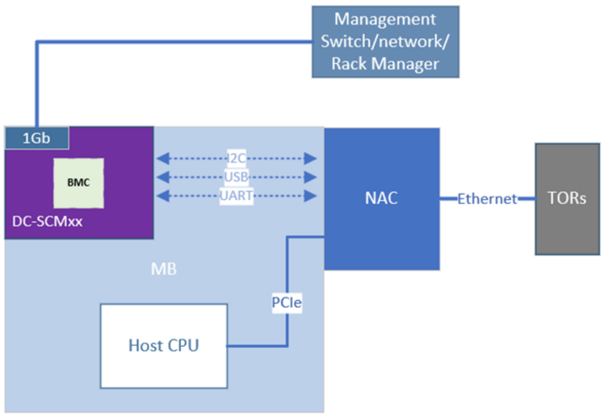

Fig. 6. System view of simplified cloud server connectivity

The UART interface usually serves as a serial port providing out of band admin console to the SoC on the NAC. Mainly used for debugging and diagnostics purposes. The I2C interface is usually used for BMC to collect critical telemetry data and monitor various parameters/logs events from key components on the NAC, which is crucial for thermal power and error detection in the data center. This interface is also used for out-of-band firmware update and recovery as well as remote management to NAC's RoT via utility commands designed for secure and efficient management. The BMC interacts with NAC's RoT to enforce these security policies, ensuring that only authorized firmware is executed.

The USB 2.0 interface usually has two functions: one as storage allowing the BMC/Rack manager to transfer files to and from the NAC. The other is serving as high-speed USBvNIC communication running at 480Mbps. Such as pushing NAC's SEL (system event log) from SOC to BMC, configuration file transfers, diagnostics and telemetry data collection. Fig. 7 illustrates the detailed interfaces between BMC and NAC on the management platform.

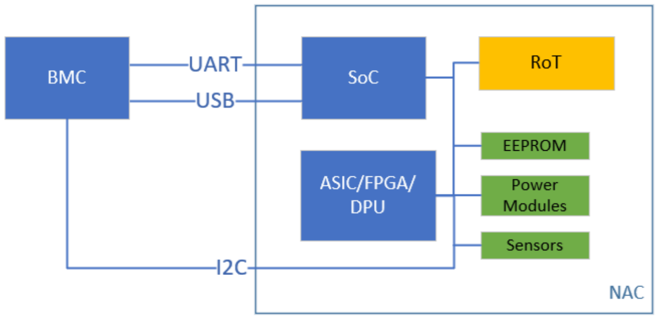

Fig. 7. NAC-BMC interfaces on management platform

A. Changing USB to SGMII

As NAC can support more and more offloading functionalities and become increasingly crucial to platform operations, there are various demands for BMC to communicate with NAC cards. A SGMII interface would be more suitable than USB 2.0 interface for next generation server design between new DC-SCM and NAC for the following reasons:

1) Improved speed and signal integration: Effective bit rate of SGMII is 1.0 Gbps while USB 2.0 max data transfer speed is only 480Mbps. SGMII uses differential signaling, which provides better noise immunity and signal integrity compared to USB. This makes SGMII less susceptible to electromagnetic interference, which can be a security advantage in environments with high electrical noise.

2) Protocol overhead and complexity: USBvNIC involves more protocol overhead and complexity, which can introduce additional attack surfaces. SGMII, being a simpler and more direct interface, reduces the potential for vulnerabilities.

3) Data transfer security: SGMII's use of differential signaling and 8B/10B encoding provides inherent security advantages by reducing the likelihood of data corruption and interception during transmission than USB vNIC.

4) More flexibility to system design: since not all servers support USB interface dedicated to NAC PCIe slot and the future trend of 1 to N (NAC to hosts) or N to 1 design, SGMII could provide better performance as high bandwidth and low latency.

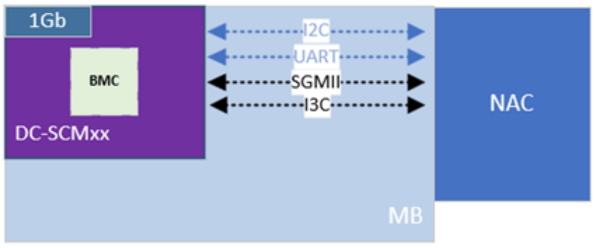

Fig. 8. Advanced interfaces on NAC management platform

Overall, changing USB 2.0 to SGMII will make connections between DC-SCM/BMC to NAC faster, more reliable and secure. Which provides the foundation for more offloading enablement on the NAC SoC side and maintaining high quality monitoring and manageability at system level.

B. Adding I3C interface

The traditional I2C interface between BMC and NAC has carried over a lot of important tasks such as telemetry and RoT communication between multiple endpoints on the Network accelerator card. However, the speed of I2C begins to show limitations for these out of band controls. There is a need to switch or add to much higher speed I3C interfaces. Compared to I2C, I3C can support data rates up to about 40Mbps at a master clock of 12.5 MHz which is significantly higher than I2C's max speed of 400kbps. I3C also supports dynamic addressing which allows for more flexibility for NAC integration into different server platforms. The other benefit of I3C is hot-join feature which allows the responder devices to be added or removed from the system without interrupting the bus, which enhances the scalability of the system and reliability when one of the responder devices goes into error stage and potentially could hang the whole bus for the I2C application. I3c is also backward compatible with I2C for legacy design and devices.

Overall, I3C provides significant improvements in speed, flexibility and compatibility, which suits more modern applications like BMC to NAC design.

IV. EMERGING TRENDS

The advanced hardware interfaces design is the foundation for secured management between BMC and NAC from various aspects of performance, security and reliability. Firmware, software and upper agents rely on these interfaces to do essential OOB security updates, remote control and recovery. It will also be used for more complex and high-density next generation server and rack system design like 1 to N / multi-host or N to 1/multi-NIC concepts to achieve better TCO and efficiency.

When it comes to security, Post-Quantum Cryptography (PQC) is on the rise, aiming to defend against quantum computer threats by creating new cryptographic algorithms that can withstand quantum attacks. This shift requires updates to both software and hardware components. The transition also means tackling performance issues, such as heightened computational demands and larger key sizes, which may impact system efficiency. Moreover, ensuring compatibility with existing protocols and systems is vital for a smooth transition. Adopting PQC into BMC and NAC security architecture design will enhance the long-term security of network accelerators, protecting data and communications from future quantum threats and making the platform resilient and future-proof.

ACKNOWLEDGMENT

We sincerely thank Shawn Swilley, Andrew Putnam, Vishal Soni and Bryan Kelly for their invaluable guidance and support. Our gratitude also extends to Parvathi Bhogaraju for her contribution on prototyping OpenSSH certificate authentication and platform attestation.

REFERENCES

[1] "Pantsdown" BMC attack: https://eclypsium.com/blog/qct-pantsdown-an-executive-summary/
[2] Secure future initiative: https://www.microsoft.com/en-us/trust-center/security/secure-future-initiative
[3] TCG DICE: https://trustedcomputinggroup.org/work-groups/dice-architectures/
[4] NIST800-193: Platform Firmware Resiliency Guidelines: https://csrc.nist.gov/pubs/sp/800/193/final
[5] Project Cerberus: https://github.com/Azure/Project-Cerberus
[6] Caliptra RoT: https://www.opencompute.org/documents/caliptra-silicon-rot-services-09012022-pdf
[7] Arm® TrustZone®: https://www.arm.com/technologies/trustzone-for-cortex-a
[8] Microsoft fTPM 2.0 reference implementation: https://www.microsoft.com/en-us/research/publication/ftpm-a-firmware-based-tpm-2-0-implementation/
[9] Project Cerberus Challenge Protocol: https://github.com/opencomputeproject/Project_Olympus/blob/master/Project_Cerberus/Project%20Cerberus%20Challenge%20Protocol.pdf
[10] DMTF SPDM:https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.0.1.pdf