A Comprehensive Framework for Enhancing Security in InfiniBand Architecture

Manhee Lee, *Student Member*, *IEEE*, and Eun Jung Kim, *Member*, *IEEE Computer Society*

Abstract—The InfiniBand Architecture (IBA) is a promising communication standard for building clusters and system area networks. However, the IBA specification has left out security aspects, resulting in potential security vulnerabilities, which could be exploited with moderate effort. In this paper, we view these vulnerabilities from three classical security aspects—confidentiality, authentication, and availability—and investigate the following security issues. First, as groundwork for secure services in IBA, we present *partition-level* and *queue-pair-level* key management schemes, both of which can be easily integrated into IBA. Second, for confidentiality and authentication, we present a method to incorporate a scalable encryption and authentication algorithm into IBA, with little performance overhead. Third, for better availability, we propose a stateful ingress filtering mechanism to block denial-of-service (DoS) attacks. Finally, to further improve the availability, we provide a scalable packet marking method tracing back DoS attacks. Simulation results of an IBA network show that the security performance overhead due to encryption/authentication on network latency ranges from 0.7 percent to 12.4 percent. Since the stateful ingress filtering is enabled only when a DoS attack is active, there is no performance overhead in a normal situation.

Index Terms—Cluster security, InfiniBand Architecture, Galois/Counter Mode, authentication, encryption, availability, DoS.

1 Introduction

Computer clustering is a popular trend in academia, as well as in the industry for high-performance and high-availability computing. Widespread use of cluster systems in a diverse set of applications has spurred significant interest in designing such servers, considering performance, scalability, and quality of service. However, cluster computer developers have paid more attention to performance and cost efficiency than to security. As a result, numerous security loopholes of cluster servers have been revealed and, consequently, the design of secure clusters has recently surfaced as a critical issue.

The easiest way to protect clusters is to isolate them physically, like some government and military agencies do, but this cannot be a general solution. Instead, reinforcing firewalls can be a common method, but it is well known that firewalls cannot prevent all possible threats. First of all, firewalls are useless at preventing inside attacks [1]. Also, firewalls scan only certain layers, therefore, there always exist potential attacks using upper layers. In addition, firewalls themselves have security vulnerabilities or can be misconfigured to operate improperly. The following are such examples. Geer reported that several products of well-known security companies such as Check Point Software Technologies, Symantec, and Zone Labs had potentially dangerous flaws that could allow hackers to gain control of the systems, disable the computers, or cause other problems [2]. Wool quantified configuration errors of firewalls installed on several sites. He found that many sites had serious configuration errors, and for example, almost 80 percent of firewalls allowed *any* service and insecure access to firewalls, which are not desirable for high security [3].

Besides improving the strength of firewalls, the security of cluster systems can be enhanced in many other ways. For example, using secure operating systems (OS) like SELinux, preventing buffer overflows of server applications, or partitioning cluster nodes into several logical groups with different root privileges can improve cluster security. Among many possible approaches, we focus on secure communication by encrypting/authenticating communication data. Protecting communication data prevents potential eavesdropping on data and passwords, and unauthorized accesses to cluster nodes. This approach will help build secure cluster systems, especially when cluster interconnects (which are previously used to connect enclosed cluster nodes) are being extended to connect external systems located outside the cluster, as demonstrated in the Supercomputing Conference 2005, where multiple clusters and storage systems were connected through a cluster interconnect [4]. In this environment, the cluster nodes are liable to security attacks through the extended cluster interconnects.

We can provide secure communication in two ways: software-based and hardware-based. However, software-based security enhancements cannot be proper solutions in cluster systems where high performance is a primary goal because such schemes will deteriorate the overall cluster performance by taking up a great amount of computing resources of host processors. Furthermore, if the OS or a user application needs to be deeply involved in the security,

- *M. Lee is with the Department of Computer Science, Dwight Look College of Engineering, Texas A&M University, H.R. Bright Building, Room 427D, College Station, TX 77843-3112. E-mail: manhee@cs.tamu.edu.*
- *E.J. Kim is with the Department of Computer Science, Dwight Look College of Engineering, Texas A&M University, H.R. Bright Building, Room 427C, College Station, TX 77843-3112. E-mail: ajkim@cs.tamu.edu.*

Manuscript received 25 Sept. 2006; revised 2 Feb. 2007; accepted 9 Feb. 2007; published online 6 Mar. 2007.
Recommended for acceptance by M. Singhal.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0298-0906.
Digital Object Identifier no. 10.1109/TPDS.2007.1079.

1045-9219/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society

then significant code modifications are necessary, especially posing great difficulties to legacy applications. However, hardware-supported security measures impose little performance overhead, since most security operations will be done in the hardware, not consuming the host processor's computing power. In addition, the secure cluster communication provided in the cluster interconnect protocol level will be transparent to applications, resulting in seamless adoption of secure communication. Therefore, our challenge is to minimize the performance degradation and maximize the security transparency while providing secure communication in cluster.

There are several widely used cluster interconnects. Myrinet, proposed in 1995, is a switching network with low-latency cut-through switches [5]. As of November 2006, 15.8 percent of the top 500 supercomputers use Myrinet. Quadrics has been the interconnect of choice for high-end supercomputers. As of November 2006, 2.8 percent of supercomputers are using products from Quadrics. Following the success of Ethernet as a local area network, Gigabit Ethernet is making another success in the high-performance computing area. Currently, 42.6 percent of supercomputers are using Gigabit Ethernet. The InfiniBand Trade Association, an industry consortium, is leading the specification of the InfiniBand Architecture (IBA) [6]. IBA is a promising I/O communication standard positioned for building system area networks that are used in clusters, multiprocessor systems, and storage area networks. User-level communication mechanisms in IBA improve the overall system performance significantly by reducing the communication overhead of the operating system in message transfer. Among these cluster interconnects, we investigate the security of IBA, since it has potential security vulnerabilities due to lack of security considerations in the current specifications. In addition, an increasing number of sites are constructing IBA cluster systems. As highlighted in the recent top-500 supercomputer list, the number of IBA clusters has been doubled from 36 to 78 in six months. This trend is expected to continue for IBA to overtake Myrinet.

The main contribution of this paper is to investigate the following security issues for providing a comprehensive framework for security enhancement in IBA clusters. First, we point out the security vulnerabilities of IBA and its potential threats. By simulating a denial-of-service (DoS) attack inside an IBA cluster, we show that DoS attacks can degrade a cluster's communication performance by up to two orders of magnitude. Second, we present secret key management schemes, which can be tightly coupled with the existing IBA key management, and describe how we can distribute the keys in IBA securely. Third, we introduce the scalable adoption of an authenticated encryption scheme, Galois/Counter Mode (GCM), into IBA, with only minor modifications to the IBA specifications [7]. Finally, for better availability, we propose a stateful ingress filtering (SIF) to block one kind of DoS attack with invalid IBA Keys and scalable packet marking algorithms to identify the location of the attackers. Since our scheme filters ports related to only active DoS attacks instead of all ports, it incurs no performance overhead in a normal situation. This dynamic filtering scheme is further enhanced by making it robust to spoofing attacks with fake source addresses. Our scalable packet marking algorithms can trace real attackers, regardless of routing algorithms in huge regular networks.

We use a comprehensive cycle-accurate simulation testbed for IBA [8]. Simulation results of a 16-node IBA network show that the security performance overhead due to encryption/authentication on network latency is 12.4 percent for 64-byte-long packets. When the packet length increases, the overhead decreases to as low as 0.7 percent, with 1,024-bytes-long packets.

The remainder of the paper is organized as follows: We briefly introduce related work and IBA in Sections 2 and 3, respectively. In Section 4, we elaborate on security vulnerabilities of the current IBA after introducing security background. Section 5 presents a comprehensive framework to enhance IBA security, and its security and performance are analyzed in Sections 6 and 7, respectively, followed by the concluding remarks in Section 8.

2 Related Work

There has been some research to improve the security inside clusters. Yurcik et al. introduced a new concept called emergent security properties to identify the security characteristics unique to clusters, which can be used to develop a unified monitoring tool for clusters [9]. Pourzandi proposed a new security model called the distributed security infrastructure [10]. It supports a fine-grained clusterwide security enforcement on distributed applications by providing a process-level resource and access control [11]. Foster et al. proposed a communication library allowing programmers to communicate securely in geographically distributed computing environments [12], which suitably provides well-organized security in Grid. Connelly and Chien focused on incorporating confidentiality in remote procedure calls in tightly coupled applications. They applied traditional security functions such as transposition, substitution, and data padding on the marshalling layer [13]. Their performance analysis showed that encryption can be used in clusters with a minimal performance impact. Dimitrov and Gleeson presented security enhancement methods at three levels: network host interfaces, system area networks, and communication protocols [14]. Their approach is systematic enough to be considered as a guideline for enhancing the security of Myrinet or VIA-based clusters, but it is not compatible with the most recent IBA specification. An overview of various cluster interconnects and their security issues are well described in [15].

3 InfiniBand Architecture

In this section, we introduce some features of IBA that are relevant to our security enhancement in IBA. IBA is a high-speed switched interconnect connecting end nodes such as processor nodes, I/O units, and routers to build clusters and system area networks. An IBA consists of one or more subnets, and each subnet is a set of end nodes, switches, and its common subnet manager (SM). A channel adapter (CA), similar to a network interface card, connecting a processor node or an I/O node to the IBA network is called a host CA or a target CA. One or more ports in a CA can be

connected to the IBA fabric through serial links. IBA links provide various speeds—from 2.5 up to 120 Gbps.

A queue pair (QP) is a pair of work queues—a send queue and a receive queue—for a consumer (or a process). When a consumer wants to send data to another consumer, a send request is queued up to the send queue, and its CA executes the request by getting data from the user memory and sending them to the other consumer's receive queue. According to connection and acknowledgment modes, a QP is categorized as one of these five types: reliable connection, unreliable connection, reliable datagram, unreliable datagram, and raw datagram. Although a connection-oriented QP can communicate with its connected QP only, a datagram QP can communicate with multiple datagram QPs. A reliable QP provides the acknowledgment service, guaranteeing the in-order communication without error or duplication. Since the raw datagram is designed to interact with non-IBA components, it is not our focus.

Among the many features of IBA, *Management*, *Partition*, and *Key* are closely related to our design for security enhancement in IBA. First, an SM manages a subnet by configuring and managing routers, switches, and CAs in the subnet. Each node should have a subnet management agent to communicate with the SM. Second, IBA Keys are used to provide isolation and protection. Currently, five types of IBA Keys are specified as follows: Management Key (M_Key), Baseboard Management Key (B_Key), Partition Key (P_Key), Queue Key (Q_Key), and Memory Key (L_Key and R_Key). The use of each Key is summarized in [16].

A packet carries one or more IBA Keys, and the receiving CA compares the delivered Keys with the stored Keys in the CA. If the two are identical, then the packet can access the node's resource, but if not, then the packet is discarded. Third, a partition is defined as a collection of CA ports that can communicate with each other to provide the exclusive resource sharing. The partition is managed by a partition manager, but since the partition manager is usually a function of the SM, in this paper, we consider that the SM manages the partitioning.

4 Security Threat in IBA

In this section, we describe the basic security background and assumptions of our research and point out IBA security vulnerabilities and their potential attacks, which motivated our research.

4.1 Security Background

Confidentiality involves a restricted access to the data sent by a sender only to a designated receiver. Encryption scrambles an original plaintext into a ciphertext by using an encryption key, whereas decryption recovers the original plaintext from the ciphertext by using a decryption key. *The Counter (CTR)* mode is a commonly used encryption mode [17]. An input of the encryption function is neither a plaintext nor a ciphertext, but just a counter. An encryption function encrypts this counter to generate an encrypted counter, often called a *keystream*, which is XORed with a plaintext to make a ciphertext. Decryption should use the same counter to recover the original plaintext by generating the same keystream.

Authentication allows two parties to agree that a received message is authentic and not modified or forged. In this research, we use GCM, which combines encryption and authentication by using Galois Field (GF) multiplication and CTR mode encryption for a faster authenticated encryption [7]. Its detail operation will be presented in Section 5.2.

Availability refers to the "timely and reliable access to data and information services for authorized users [18]." Availability has become more important with respect to the Internet because of the increase of DoS attacks. In our research, we will propose a scalable and dynamic scheme to block DoS attacks only when necessary. A more extensive introduction to security can be found in [19].

4.2 Basic Assumptions

Before identifying and solving the security vulnerabilities of IBA, we assume the following constraints:

- The CA/switch/router can be compromised.
- Each of those devices has a tamper-resistant storage, the contents of which cannot be read or modified from outside the device. The contents are only accessible by its SM with a legitimate key.
- Attacking the traffic can have spoofed source addresses.

4.3 Vulnerabilities and Threats

4.3.1 Confidentiality

Since IBA does not provide any encryption and decryption method, eavesdropping attacks, with help from compromised switches and CAs, will succeed in acquiring all communicating data. We call this a *Type I attack*. This vulnerability is critical, especially when a cluster processes classified data.

4.3.2 Authentication

A resource to be protected has its own Key such as P_Key or L_Key and R_Key. Any captured packets that carry legitimate Keys will expose these plaintext Keys and allow a hacker to generate illegal traffic with the Keys, referred to as a *Type II attack*. If this illegal traffic is using fake source addresses, then it will be hard to find in which port the attacker is attached. We classify this attack as a *Type III attack*.

4.3.3 Availability

An attacker who does not even know any legitimate Key can stage a DoS attack by sending tremendous traffic to a victim node. Although the traffic will be discarded at the victim nodes because of the illegal Key, the traffic can affect other normal traffic when it passes through the network. We define this attack as a *Type IV attack*.

4.4 A Simulation of a DoS Attack

To show the impact of the DoS attack, we simulate a DoS attack on an IBA testbed. We partition 16 nodes of the IBA network into four partitions and choose the attacker nodes randomly. The attacker nodes randomly select victim nodes and generate illegal traffic at their full speed (2.5 Gbps, 1X IBA link). Other nonattacker nodes communicate with each other in the same partition. Our simulation uses two kinds

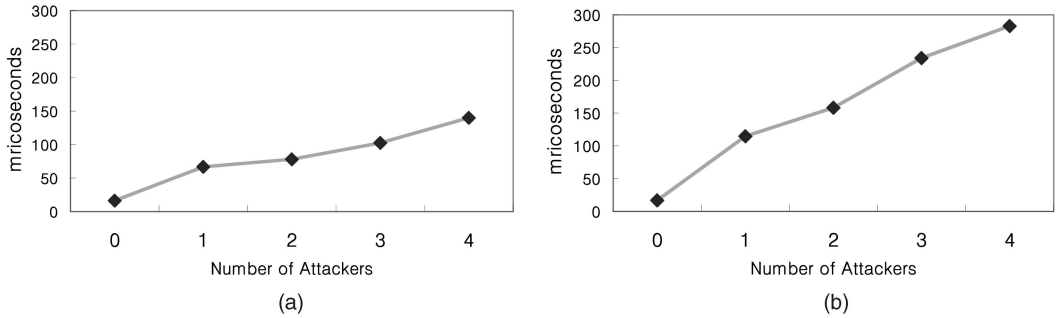

Fig. 1. Average end-to-end latency under DoS attacks. We simulate DoS attacks on our IBA testbed to estimate their effect on the overall network performance. The latencies of real-time and best-effort traffic are increased by 6 and 18 times, respectively. This shows that as the number of attackers increases, the average network latencies will increase significantly. (a) Real-time traffic. (b) Best-effort traffic.

of traffic: real time and best effort. Real-time traffic is injected at a fixed rate and has a higher priority than best-effort traffic in intermediate switches. By using a best-effort attacking traffic, we can see its effect on real-time traffic and on other best effort traffic.

Fig. 1a shows how a real-time traffic is affected by an illegal best effort traffic. With no attacker, the average end-to-end latency is around 20 μs. The figure clearly shows that the latency could increase by as much as 50 μs when even one attacker dumps traffic. As more attackers are added, the average latency increases up to 120 μs. Fig. 1b shows how the illegal best effort traffic degrades other legal best effort traffic. Its average latency increases more dramatically than that of the real-time traffic. This is because the real-time traffic has a higher priority than best-effort traffic.

5 Security Enhancement in IBA

Security vulnerabilities are inevitable, as long as IBA packets continue to carry plaintext IBA Keys for their access control. To remove such vulnerabilities, we propose that IBA use traditional cryptographic methods. Our main goal is to integrate encryption and authentication into IBA not only with minor modifications to the IBA specification, but also with only marginal performance overhead. For this, new cryptographic keys, in addition to IBA Keys, are necessary. Therefore, we first propose new key management schemes to manage and distribute the keys. Based on these key management schemes, we describe how GCM can be integrated into IBA and then propose an efficient mechanism to block DoS attacks and a scalable packet marking scheme to trace back DoS attacks that use fake source addresses.

5.1 Secret Key Management in IBA

To provide any security service, it is mandatory to have an efficient and secure key management scheme. Since the existing key management of the current IBA is not designed for security, we propose two new key management schemes: partition-level and QP-level key managements. The partition-level key management enforces that all communications inside a partition use the same secret key. This scheme ensures that data communications are secure against attacks by a hacker located in a different partition. This is simple to implement because each CA only has to maintain the same number of secret keys as partition keys. However, this scheme is vulnerable to security attacks originating from the same partition. To remove this problem, we propose to use the QP-level key management for finer-grained security. Since QP is the smallest communication entities, the QP-level key management will guarantee confidentiality and integrity for all QP communications. Both management schemes can be adopted into IBA without significant changes in the specification. We choose the symmetric key mechanism for better performance.

5.1.1 Partition-Level Key Management

In this key management, a secret key will be created and assigned to each partition. All components in each partition, like switches and CAs, should have its secret key. The secret key is used to make a message authentication code (MAC) and encrypt every packet transferred within the partition. Even though P_Keys are still available in packets, the exposure of the P_Keys does not pose any further threats, since the encryption/authentication relies on its secret key and not on the P_Keys. Therefore, even if an attacker captures IBA packets, he cannot make legitimate packets with the captured P_Keys because the attacker does not know their secret keys. This partition-level key management removes the aforementioned Type I and Type II attacks coming from different partitions.

There are several advantages of this scheme. First, every IBA packet should carry a partition key, whereas other IBA Keys may not be carried, so it is easy to enforce security throughout the network. Second, since the number of partitions is often small, the number of secret keys will not be so large, thus requiring a small amount of space for the key management. Last, this key management makes use of the well-defined key management mechanisms of IBA by adding one more column into the existing Key tables. The current IBA key management, therefore, can successfully manage new secret keys along with its IBA Keys. However, this key management has a disadvantage in that it cannot block attacks originating from the same partition. This is because CAs automatically decrypt and authenticate incoming packets, as long as the packets have the same P_Key as the CA.

5.1.2 QP-Level Key Management

In order to remove aforementioned threats inside the partition, we alternatively propose the QP-level key management scheme: two connection-oriented QPs share

a temporary secret key, referred to as a *session key*, and a connectionless QP has its own secret key that can be sent to other connectionless QPs. There are two benefits of implementing the QP-level key management. One is that even in the same partition, Type I and Type II attacks are not viable without a legitimate QP secret key. The other benefit is that it also removes additional threats resulting from the exposure of other IBA Keys. For example, even if an R_Key is exposed to a hacker, he cannot access the corresponding remote memory without the QP's secret key. Therefore, the QP-level key management provides a fine-grained key management enough to secure all communications in IBA.

5.1.3 Initial Key Distribution

To use the aforementioned key management schemes, it is necessary to distribute secret keys securely. We first define two basic keys, a CA secret key (CA_SK) and an SM secret key (SM_SK), on which subsequent key distributions depend. A CA_SK is a unique secret key assigned to each CA by its manufacturer and stored in each CA's tamper-resistant storage.[1] A manufacturer passes this key offline, assumed to be secure, to an IBA cluster administrator. Therefore, since the administrator has all the secret keys of the CAs in the cluster, he can access any CA by using the CA's secret key. The other basic key SM_SK is assigned to each subnet by the administrator and stored in SM's CA. Then, the SM distributes its secret key to all the CAs in its subnet as follows:

1. SM sends SM_SK to all CA_i in the subnet:

$$\mathrm{E}_{CA\_SK_i}(\mathrm{SM\_SK}).$$

2. CA_i decrypts SM_SK:

$$\mathrm{SM\_SK} = \mathrm{D}_{CA\_SK_i}(\mathrm{E}_{CA\_SK_i}(\mathrm{SM\_SK})).$$

Note that MAC is not described for simplicity, and $\mathrm{E}_K(\mathrm{T})$ and $\mathrm{D}_K(\mathrm{T})$ represent that a plaintext T is encrypted and decrypted by a secret key K, respectively. Since the SM_SK is encrypted using the receiving CA's secret key, the distribution is secure. The decrypted SM_SK is stored in each CA. One concern in storing the SM_SK in CAs is that a non-SM node might use the SM_SK inappropriately. Therefore, each CA needs to be hardwired so as not to decrypt or authenticate normal packets by using the SM_SK. If there are multiple subnets in an IBA network, then each subnet has its own SM_SK, and SMs should communicate securely with each other. For this purpose, the administrator defines a shared SM secret key SSM_SK to be shared by all SMs.

5.1.4 Partition-Level Key Distribution

In the partition-level security, a partition secret key P_SK is created and distributed as follows:

1. SM sends P_SK to all CAs in the subnet:

$$\mathrm{E}_{SM\_SK}(\mathrm{P\_SK}).$$

2. CA_i decrypts P_SK:

$$\mathrm{P\_SK} = \mathrm{D}_{SM\_SK}(\mathrm{E}_{SM\_SK}(\mathrm{P\_SK})).$$

When a partition is created across several subnets, one SM transfers the P_SK to other SMs to distribute it inside their own subnets:

1. SM sends P_SK to all SMs in other subnets:

$$\mathrm{E}_{SSM\_SK}(\mathrm{P\_SK}).$$

2. SM_j decrypts P_SK:

$$\mathrm{P\_SK} = \mathrm{D}_{SSM\_SK}(\mathrm{E}_{SSM\_SK}(\mathrm{P\_SK})).$$

3. SM_j sends P_SK to CA_i in the subnet:

$$\mathrm{E}_{SM_j\_SK}(\mathrm{P\_SK}).$$

4. CA_i decrypts P_SK:

$$\mathrm{P\_SK} = \mathrm{D}_{SM_j\_SK}(\mathrm{E}_{SM_j\_SK}(\mathrm{P\_SK})).$$

From now on, all communications inside the partition can be encrypted and authenticated by P_SK.

5.1.5 QP-Level Key Distribution

For QP-level security, there are two secret key setup processes, depending on the type of communication. When a consumer wants a connection-oriented communication, the source CA creates a session key randomly and then sends a request message (REQ) containing the session key to a destination CA. If the destination CA accepts the request, then the CA creates a connection-oriented QP, decrypts the session key, and then replies with a reply message (REP) containing the destination QP's information. The session key is used to encrypt and authenticate all the following messages between the two QPs.

To protect the session key distribution, we assume that every pair of CAs in a partition has a unique secret key. Let $\mathrm{CA\_SK}_{ij}$ be a secret key for communication between CA_i and CA_j and let $\mathrm{CA\_SK}_{ij} = \mathrm{CA\_SK}_{ji}$. Therefore, an REQ from CA_i to CA_j is encrypted using $\mathrm{CA\_SK}_{ij}$. The SM generates all n-to-n keys and distributes the keys to all CAs beforehand. Based on this secret key, the session key distribution is described as follows:

1. CA_i sends an REQ message to CA_j containing a session key SK_l:

$$\mathrm{REQ}(\ldots, \mathrm{E}_{CA\_SK_{ij}}(\mathrm{SK}_l)).$$

2. CA_j decrypts SK_l:

$$\mathrm{SK}_l = \mathrm{D}_{CA\_SK_{ji}}(\mathrm{E}_{CA\_SK_{ij}}(\mathrm{SK}_l)).$$

3. CA_i sends data to CA_j: $\mathrm{E}_{SK_l}(\mathrm{Data})$.
4. CA_j sends data to CA_i: $\mathrm{E}_{SK_l}(\mathrm{Data})$.

After step 2, CA_i and CA_j can send data using SK_l at any time in steps 3 and 4.

1. All the following secret keys will be stored in this storage too.

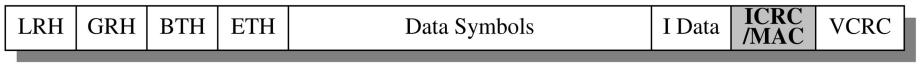

Fig. 2. IBA Packet format with MAC. ICRC covers all invariant fields from LRH to I Data to detect errors on the data communication. Since ICRC does not change end to end, we propose to use this field as the MAC location that will carry MAC in the security enhanced IBA (LRH, GRH, BTH, Extended Transport Header (ETH), ICRC, and VCRC).

In contrast to a connection-oriented QP, a datagram QP can communicate with more than one QP. To control accesses to this QP, each datagram QP carries the destination QP's Q_Key, and an access to this QP is allowed only if the Q_Key in the requesting packet is the same as the one in the receiving QP. Therefore, before two datagram QPs send data, they need to exchange packets to know each other's Q_Key. As noted earlier, the Q_Key in the packet is also plaintext, so it is vulnerable to the capturing attack. To prevent this problem, we propose that a secret key be assigned to each Q_Key and exchanged. The following explains a reliable datagram QP's communication establishment steps. In an unreliable datagram communication, SIDR_REQ and SIDR_REP are used instead of REQ and REP:

1. CA_i sends an REQ message to CA_j containing a session key SK_l:
$$\mathrm{REQ}(\ldots, \mathrm{Q\_Key}_{local}, \mathrm{E}_{CA\_SK_{ij}}(\mathrm{SK}_l)).$$
2. CA_j decrypts SK_l:
$$\mathrm{SK}_l = \mathrm{D}_{CA\_SK_{ji}}(\mathrm{E}_{CA\_SK_{ij}}(\mathrm{SK}_l)).$$
3. CA_j sends an REP message to CA_i containing another session key SK_r:
$$\mathrm{REP}(\ldots, \mathrm{Q\_Key}_{remote}, \mathrm{E}_{CA\_SK_{ji}}(\mathrm{SK}_r)).$$
4. CA_i decrypts SK_r:
$$\mathrm{SK}_r = \mathrm{D}_{CA\_SK_{ij}}(\mathrm{E}_{CA\_SK_{ji}}(\mathrm{SK}_r)).$$
5. CA_i sends data to CA_j:
$$\mathrm{E}_{SK_r}(\mathrm{Data}), \mathrm{Q\_Key}_{remote}.$$
6. CA_j sends data to CA_i:
$$\mathrm{E}_{SK_l}(\mathrm{Data}), \mathrm{Q\_Key}_{local}.$$

The space to store a 128-bit session key in request and reply packets is available in the PrivateData field in the packet formats. The space overhead for storing 1-to-n secret keys, CA_SK_{ij}, in each CA is 128 bits times the number of CAs in its partition. If a partition consists of 2^{10} CAs, then the total space is 20,480 bytes, including 32 bits of MAC for each secret key. If the tamper-resistant storage cannot hold all secret keys, then they can be stored in normal memory in encrypted form.

The initial overhead for generating and distributing all CA_SKs is analyzed as follows: Those secret keys are random numbers, and they can be generated by the Advanced Encryption Standard (AES) function in the CTR mode [20]. In a partition with 2^{10} CAs, the total number of SA_SKs is 2^{18}. Since a recent hardware implementation of AES can encrypt at $30 \sim 70$ Gbps, it can generate all the secret keys within a second, assuming that the length of a secret key is 128 bits long. An SM needs to distribute all the secret keys sequentially. If an IBA cluster has a 2.5-Gbps link, assuming a 1,024-byte-long packet is carrying a secret key, then it will take also less than 1 second to send out all the packets. Considering a fairly long operation time of cluster systems, our approach is scalable to even larger networks, since such a small initial overhead will be easily amortized.

5.2 Message Encryption and Authentication in IBA

Now, we will explain where an MAC is to be located in each packet and show an example of how an authenticated encryption algorithm GCM is integrated into IBA.

Cyclic Redundancy-Check (CRC) codes are widely used for error detection on data communication. IBA defines three types of CRC—Invariant CRC (ICRC), Variant CRC (VCRC), and Link Packet CRC (LPCRC)—as shown in Fig. 2 (only ICRC and VCRC are depicted). ICRC covers all invariant fields from the Local Route Header (LRH) to I Data, not covering the variant fields that the switches or routers can change. In contrast, VCRC covers the LRH up to the last byte before the VCRC. If some fields are changed in a switch or a router, then the VCRC is calculated again. LPCRC is present at the end of all Link packets. The only Link packet in the current IBA specification is the flow-control packet, so we do not consider LPCRC.

Our main idea is to use the ICRC field as the MAC location, as depicted in Fig. 2. We can gain two advantages in using the ICRC field. First, ICRC does not change from end to end. It only covers static fields, so it can be considered as a transport-level CRC. Since the authentication is an end-to-end transport feature, the ICRC field is a nice fit for the MAC location. The second advantage is that we do not have to change the IBA packet format, which is extremely important. By having ICRC as a default and MAC as an option, the original drivers that use ICRC do not have to be changed. Therefore, our idea will become fully compatible with the current IBA.

To accommodate various authentication algorithms, the *Reserved* field of Base Transport Header (BTH) is used to identify authentication algorithms. BTH is a basic header that every transport packet carries. If the value is zero, then the packet is using the original ICRC. A nonzero value indicates that an authentication algorithm is in use, and an MAC is stored in the ICRC field. This can be exploited to provide an on-demand authentication service. For instance, suppose that in a partition, a very important job is running. The administrator can enable the authentication for that partition by using MAC instead of ICRC. Since the

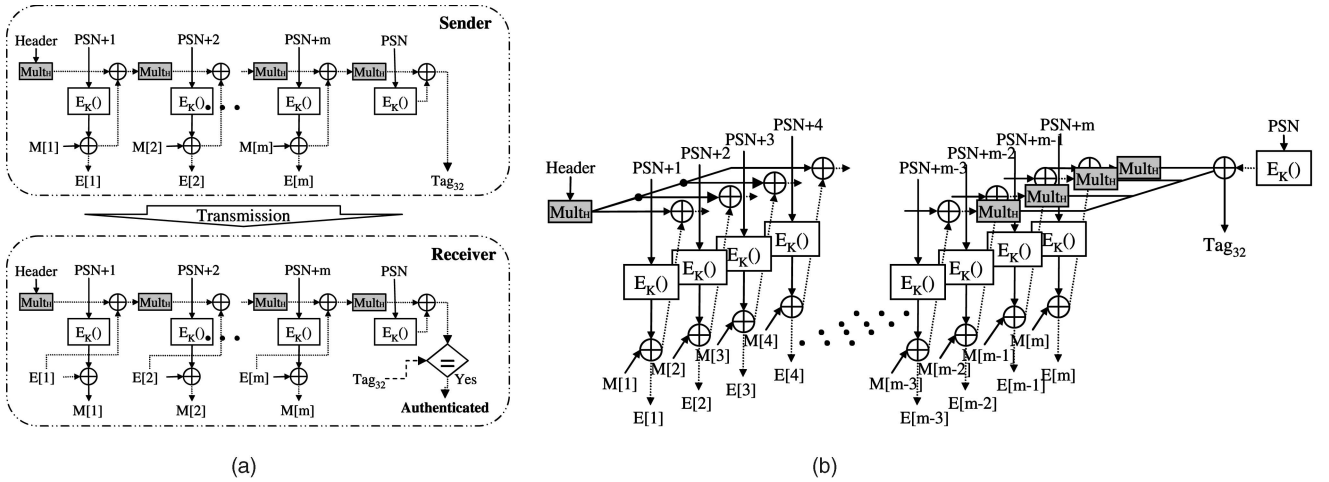

Fig. 3. GCM architecture using AES. In the sender side, an original data M is divided into multiple 128-bit-long messages, $M[1], M[2], \ldots, M[m]$. This substring is encrypted by XORing the result of the encryption of $\mathrm{PSN} + i$. The first input of $Mult_H$ is the header of the packet, and the subsequent inputs are the encrypted substrings $E[i]$. Note that due to the pipelining capability of recent AES and $Mult_H$ hardware, adjacent substrings are encrypted and authenticated in parallel with a small pipelining delay. The receiver's architecture is almost similar, except that $E[i]$ and $M[i]$ are switched. GCM 4X utilizes four AES hardware to keep up with 4X InfiniBand speed. Since there is no dependence between AES blocks, it can get a nearly linear speedup. (a) 1X GCM. (b) 4X GCM.

authentication can be disabled and enabled at any time, our mechanism provides the flexible authentication service.

Although any encryption/authentication algorithm can be used for a secure IBA, it will be very helpful to take one algorithm and show its performance because we can estimate the performance impact of our approach in real systems. We choose GCM, since it has two main advantages in terms of security and speed. The security strength of GCM is the same as the strength of its block cipher [21], [7]. Since we use AES in this research, and it is considered to be secure without any serious weakness until now, the confidentiality of each IBA packet can be improved greatly. With duplicate hardware, the whole authentication can be done in parallel in several additional cycles after the AES computation [7].

Fig. 3 depicts the architecture of GCM to encrypt and authenticate packets by using AES. A Packet Sequence Number (PSN) and its sequential numbers are encrypted and XORed with plaintexts to generate ciphertexts. The header of a packet and ciphertexts are multiplied in a GF to generate an MAC. This architecture can fully utilize the pipelined AES, as shown in Fig. 3a. The parallelism can be further exploited by the use of the multiple encryption units, as depicted in Fig. 3b. A 12X IBA network can be extended in the same way. Note that the Tag calculation only requires the previous results. The performance overhead coming from this delay will be analyzed in Section 7.

5.3 Stateful Ingress Filtering

Until now, we described how we can provide confidentiality and authentication services in IBA. In this section, we explain how we can mitigate the DoS (Type IV) attack problems for better availability.

This vulnerability stems from the fact that IBA Key checking is done in the destination nodes. A straightforward remedy is to make all switch ports filter out the packets carrying illegal Keys. We call this method the Duplicate Table (DT) scheme. However, this is very inefficient in terms of memory usage and network performance because every switch port has to maintain a partition table, and all packets should be checked at every hop. For example, when a network consists of n nodes and s switches with r ports, and all nodes join the same number of p partitions, where k is the length of a partition key in bytes, each switch will need $r \cdot p \cdot k$ bytes to store partition tables, which means that $r \cdot p \cdot s \cdot k$ bytes will be used in total for the whole network, as shown in Table 1. Let $f(i)$ be the time that a lookup table function f takes to search an entry from a table having i entries. DT will take $f(p)$ at every hop, thus requiring $h \cdot f(p)$ to route a packet to a destination with h, an average of hop counts.

To remove the high overhead of DT, Ingress Filtering (IF), commonly used on the Internet, can be adopted in IBA. In IF, the packet filtering is applied to ingress ports that are directly connected to nodes. If all invalid P_Key packets are filtered out at ingress ports, then it is not necessary to check at intermediate hops. Therefore, IF can increase the memory and network efficiency by using less memory and taking shorter time, as shown in Table 1, where $\frac{n}{s}$ is the average number of nodes directly connected per switch.

Still, however, there are redundant operations in IF because it is necessary to look up P_Key tables in all ingress ports, regardless of whether a DoS attack is occurring or not. To remove such redundancy, we propose SIF, which filters attacking traffic only when attacks are active. To decide when and where to enable filtering, we propose to use a *trap* message. In IBA, when an incoming packet's Key

TABLE 1
Partition Enforcement Overhead

	DT	IF	SIF
one switch mem	$r \cdot p \cdot k$	$\frac{n}{s} \cdot p \cdot k$	$\frac{n}{s} \cdot Min(p', p) \cdot k$
all switches mem	$r \cdot p \cdot s \cdot k$	$n \cdot p \cdot k$	$n \cdot Min(p', p) \cdot k$
Look-up/pkt	$h \cdot f(p)$	$f(p)$	$Pr(n) \cdot f(Min(p', p))$

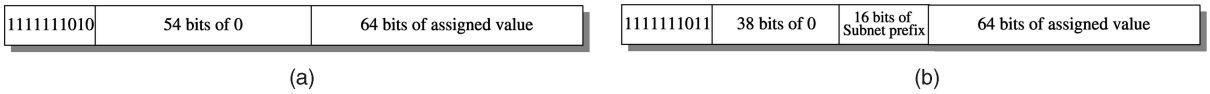

Fig. 4. GRH address format. GRH can be in use both within a subnet and between subnets by using link-local prefix 1111111010 and site-local prefix 1111111011, respectively. Sixty-four bits are used for addressing inside a subnet, and an additional subnet address (16 bits) is necessary for addressing between subnets. The commonly unused 38 bits of two formats can be used as the MF. (a) Link-local address format. (b) Site-local address format.

does not match the receiver's Key, the receiver optionally sends a trap message to its SM. We suggest that this trap message convey the invalid Keys and the packet's source address. When the SM receives the trap message, it will find the switch port connected to the attacking node and enable the filtering of the port.

Note that this filtering needs invalid Keys and not valid Keys. Since the current IBA switches have a table of valid P_Keys designed for the partition enforcement, we introduce a table of invalid P_Keys, referred to as *Invalid_P_Key_Table*. When the SM receives a trap message, it registers the invalid P_Key to its Invalid_P_Key_Table. To disable this filtering when attacks end, we define *Ingress P_Key Violation Counter* in switch ports. It counts the number of invalid P_Keys sent from the filtered node. If the Ingress P_Key Violation Counter does not increase for some time, then the contents of Invalid_P_Key_Table will be flushed, and the IF will be disabled. Consequently, since SIF will be activated only when a node is injecting packets with invalid P_Keys and only where the attacking traffic is coming from, it will eliminate all the redundant filtering operations of DT and IF.

One concern is that the Invalid_P_Key_Table might grow bigger than the valid P_Key_Table, as an attacker uses many invalid P_Keys. In this case, to prevent the long lookup table time, the IF needs to look up P_Key_Table to pass the legitimate traffic with valid P_Keys. The Ingress P_Key Violation Counter still needs to be counted because if it does not change for some time, then the switch's port has to return to the normal operation. When disabling filtering, it is necessary to flush Invalid_P_Key_Table and reset Ingress P_Key Violation Counter. Due to this, any attempts to stage a DoS attack on the SM by triggering multiple trap messages will also be blocked soon. This is because once the number of invalid P_Keys becomes larger than the valid P_Key_Table, only packets with valid P_Keys will be passed.

SIF's overhead depends on how often DoS attacks are occurring and how many invalid P_Keys are used for the attacks. Let $Pr(n)$ be the probability that one node joins a P_Key attack and let p' be the number of invalid P_Keys used in the P_Key attacks. The memory overhead for one switch is $\frac{n}{s} \cdot Min(p', p) \cdot k$ bytes, and the lookup table operation will be $Pr(n) \cdot f(Min(p', p))$, where $Min(p', p)$ is the minimum between p' and p. By limiting the maximum p', the memory overhead will be smaller than or equal to that of DT or IF. Furthermore, considering that DoS attacks are not occurring often, the low $Pr(n)$ will make the lookup table overhead of SIF negligible.

5.4 Source Identification Scheme in IBA

One more concern to implement the SIF is that a compromised node may use fake source addresses, referred to as *spoofing*. In this case, the SIF would block wrong ports. Since IBA allows the software to choose which source address will be used in an outgoing packet, spoofing is possible if a hacker successfully manipulates the address information in a CA.

To remove this problem, a source identification scheme capable of tracing back the real attackers is necessary. In [22], Aljifri categorized traceback methods into four categories: link testing, logging, ICMP-based traceback, and packet marking. Since link testing and ICMP-based traceback should generate additional traffic, they will degrade the overall performance of the cluster. Logging requires additional storage and computing power in routers/switches, which is not desirable due to additional costs. In contrast, the marking schemes proposed in [23], [24], [25] simply mark (or write) some information in packets while routing the packets to help the receiving nodes investigate where the attacking traffic comes from. Since marking schemes do not cause additional overhead, except additional marking time in each switch, we choose to develop a marking scheme in this study.

5.4.1 Marking Field in IBA

The first requirement to use any marking scheme is an additional space in an IBA packet to mark on, referred to as *Marking Field* (*MF*). An IBA packet uses different headers relying on where a destination node is located. If a destination and a source nodes are located in the same subnet, then LRH is enough to switch inside a subnet. However, IBA specifies that a Global Route Header (GRH) can be used within a subnet by setting the link-local prefix to 1111111010, as depicted in Fig. 4a. If the two communicating nodes are located in different subnets, then the GRH is necessary to route these packets properly. For this, the destination address should set the site-local prefix to 1111111011, as described in Fig. 4b. Therefore, the 10-bit prefix indicates whether a packet's destination node is inside the site. If a packet is leaving a site, then we do not mark the packet. Otherwise, 54 and 38 bits of a destination address are set to zeroes. We suggest using the common 38 bits as the MF.

5.4.2 Deterministic Packet Marking

In a deterministic marking algorithm, each switch (or router) writes its own information such as switch indexes in every packet's MF. When a cluster is not so big, end nodes and switches are usually connected in a tree structure. For example, the lowest level switches equipped with tens of ports are connecting the end nodes, and those

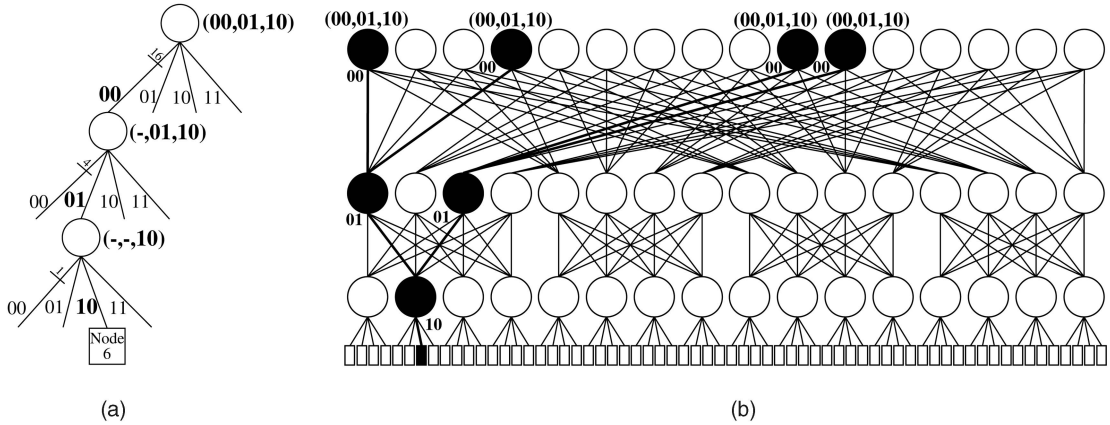

Fig. 5. Deterministic packet marking on a three-level fat tree. As a packet goes up, each switch marks an incoming port's number in the packet's MF. Marking completes when a packet reaches the top-level switch. This marking scheme works for a fat tree built by a large number of minimal basic switches, since this scheme marks only port numbers and not switch indexes. (a) Logical topology. (b) An implementation using eight-port basic switches.

switches are connected to upper level switches hierarchically. In this environment, all hops can be most likely recorded in the MF. If there are n switches in a cluster, then $\lfloor 38/\log n \rfloor$ hops can be recorded in the 38-bit MF. Alternatively, each switch's port index can be recorded instead of the switch index. In this method, with the 38 bits MF, $\lfloor 38/\log p \rfloor$ hops will be traced back, where p is the maximum number of ports in switches.

We show an example of a deterministic packet marking on a fat tree, which is the most popular network topology in IBA clusters. Fig. 5a depicts a three-level fat tree in a logical view. As the level goes up, the bandwidth of each link is squared to provide nonblocking switching. Before marking packets, each port is first numbered in its switch by using $\log p$ bits, where p is the number of ports in the switch. In Fig. 5a, 2-bit port numbers are shown next to links. Accordingly, an address of a leaf node is the concatenation of the port numbers from the top-level switch to the lowest level switch. For example, the address of node 6 is (00,01,10).

To identify a source node, when a packet traverses to upper levels, each switch writes down the index of the port that the packet comes through. For example, the lowest switch marks 10 on the packets coming from node 6, so their MF will be $(-, -, 10)$. Then, 01 and 00 are written at the upward switches. As a result, the MF field will have (00,01,10) at the top-level switch, which is exactly the same as node 6's address. A real implementation of the fat tree using a simple small-sized switch is depicted in Fig. 5b. Even though packets coming from one leaf node can take different upward paths, the recorded MF values are always the same because each switch records a port number and not a switch index. As described in Fig. 5b, although the packets from node 6 take four different paths, all the recorded values are (00,01,10). When the maximum number of ports is p, a 38-bit MF will record at most $\lfloor 38/\log p \rfloor$ hops. Assuming that all intermediate nodes are switches, and leaf nodes are computing nodes or storage nodes, the maximum size of cluster that the 38-bit MF supports is $p^{\lfloor 38/\log p \rfloor}$, as summarized in Table 2.

However, when a cluster consists of tens or hundreds of thousands of nodes with regular interconnects such as mesh and hypercube, the average number of hops becomes so large that deterministic packet marking schemes cannot work. For example, the diameter of a 20×20 mesh network is 39, where *diameter* is the longest hop in the shortest paths between any pair of nodes. Furthermore, when the routing is not static but adaptive, recording all hops becomes impossible because a routing path can be much longer than the shortest path, depending on the current network condition. Therefore, the deterministic marking schemes will be inapplicable to large networks with regular topologies or adaptive routing schemes.

Inputs:	Destination D $(d_0, d_1, ..., d_{n-1})$
	Current X $(x_0, x_1, ..., x_{n-1})$
Output:	Next Y $(y_0, y_1, ..., y_{n-1})$
	Source S $(s_0, s_1, ..., s_{n-1})$
Variables:	Distance V $(v_0, v_1, ..., v_{n-1})$
	New distance $V\prime$ $(v_0\prime, v_1\prime, ..., v_{n-1}\prime)$
	Difference Δ $(\delta_0, \delta_1, ..., \delta_{n-1})$
Procedure:	
	if X = D then
	V := Extract_MF();
	S := X - V;
	exit;
	endif
	Y := Routing(V);
	V:= Extract_MF();
	Δ := Y - X;
	$V\prime$:= V + Δ;
	Store_MF(V');
	exit;

Algorithm 2. Deterministic Distance Packet Marking

TABLE 2
Scalability of Deterministic Packet Marking Scheme

Topology	Max ports per switch	Max Cluster Size
fat-tree	p	$p^{\lfloor 38/\log p \rfloor}$

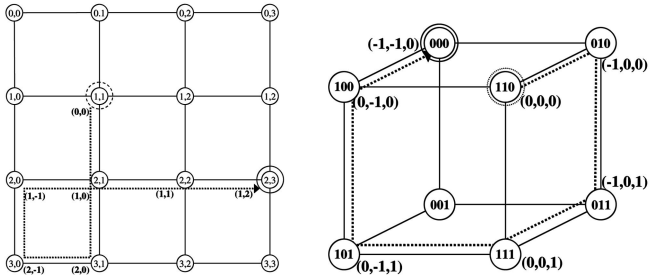

Fig. 6. An example of DDPM. When a packet traverses through switches, each switch calculates the difference between its own coordinate and a neighboring switch passing the packet. The difference is added up to the current distance vector to update the MF. The receiving node easily identifies the source node by subtracting (or XORing) the distance vector from the receiving node's position. Note that this marking algorithm is robust to any adaptive routing algorithms that allow looping, as shown in (a). (a) A 4 × 4 mesh. (b) A 3-hypercube.

5.4.3 Deterministic Distance Packet Marking

To alleviate this problem, we propose the Deterministic Distance Packet Marking (DDPM) scheme. This scheme records only the relative distance from the current position of a packet to the source node, without keeping track of intermediate paths. For example, in an n-dimensional mesh, we locate a node X by using its coordinate $(x_0, x_1, \ldots, x_{n-1})$. A relative distance between two nodes $X\,(x_0, x_1, \ldots, x_{n-1})$ and $Y\,(y_0, y_1, \ldots, y_{n-1})$ can be defined as a vector $V\,(v_0, v_1, \ldots, v_{n-1})$, where $v_i = y_i - x_i$. This vector is the coordinate difference. There is a unique distance vector between any two nodes in the network. That is, with a distance vector V, node Y identifies a unique node X.

In each hop, each packet's distance vector is updated. V is set to a zero vector when a packet first enters a switch from a node, which is hardcoded in the switch or CA. After the switch decides the next hop, it updates the distance vector. In regular networks, a packet moves by one hop in one dimension at each switching. Therefore, at each hop, one v_i is added or subtracted according to the routing direction. After a switch stores the new distance vector V in the MF, the switch transmits the packet to the next switch. The full algorithm is described in Algorithm 2. $Routing()$ is a function that returns the next node coordinate. $Extract\_MF()$ returns a distance vector from the GRH header, and $Store\_MF()$ stores it in the header. Regardless of the intermediate routing paths, the final distance vector V is the exact difference from the source to the destination node. Therefore, the destination can identify the source node instantly with only one packet.

Fig. 6a shows an example that a packet traverses a 2D mesh adaptively from (1,1) and (2,3). The distance vector changes as follows: (0,0), (1,0), (2,0), (2,-1), (1,-1), (1,0), (1,1), and (1,2). When (2,3) node receives the distance vector (1,2), it can subtract (1,2) from (2,3) and quickly identify the source (1,1). In the hypercube in Fig. 6b, the distance vector changes as follows: (0,0,0), (1,0,0), (1,0,1), (0,0,1), (0,1,1), (0,1,0), and (1,1,0). (0,0,0) can identify the source (1,1,0) by XORing its coordinate (0,0,0) and the distance vector (1,1,0). Note that in hypercube, XORing is used instead of subtraction.

TABLE 3
Scalability of Deterministic Distance Packet Marking Scheme

Topology	Required Field	Max Cluster Size
$n \times n$ mesh, torus	2 log n - 2	$2^{18} \times 2^{18}$
$n \times n \times n$ mesh, torus	3 log n - 3	$2^{11} \times 2^{11} \times 2^{11}$
n-cube hypercube	$\log 2^n$	2^{38}

The DDPM can support a larger number of nodes as compared to the deterministic marking algorithm, regardless of the routing policies. To support a $n \times n$ 2D mesh and torus, half of the 38-bit MF contains the distance in one dimension. The distance can be negative, so $2^{18} \times 2^{18}$ 2D mesh and torus networks can be marked by the DDPM. For a 3D mesh and torus, the DDPM can mark $2^{11} \times 2^{11} \times 2^{11}$ networks by splitting the MF into three parts (2^{33}-node cluster). For the hypercube, the whole MF can be used for the distance vector, so the DDPM can mark a 38-cube hypercube (2^{38}-node cluster). Table 3 summarizes these facts.

6 Security Analysis

6.1 Defending Attacks

Type I attack. As we showed in an example in Section 5.2, GCM encrypts by using AES, with the CTR mode considering a PSN as a counter. Since AES is considered to be secure without any serious weakness until now, the brute-force attack is believed to be the most effective attack. When a 128-bit key is used, on the average, 2^{127} trials are needed to find the key, which is infeasible in the foreseeable future. Another concern is the multiple use of the same PSN, since it will expose partial information of original plaintexts even though the amount of information is not much. Suppose E_1 and E_2 are two different ciphertexts using the same counter C and the same secret key K. If a hacker gets these two ciphertexts and XORs them, then he will get an XOR of two original plaintexts by clearing out the keystream.

In the QP-level key management, since a new session key is assigned to a new communication session, the multiple use of the same PSN will not cause this problem. However, it does in the partition-level key management, since multiple consumers will use the same key. To avoid the problem, we propose that a 128-bit random number be created and shared during a communication setup phase. The use of a PSN, along with a random number, as an input to AES will remove the concern, except when any two random numbers happen to be the same, called *collision*. The probability $P_{collision}$ that any two random numbers out of given n integers with range [1, d] are the same is approximated as follows [26]:

$$P_{collision} = 1 - e^{-(n(n-1))/2d}. \quad (1)$$

Suppose 2^{10} nodes keep sending packets at their full speed in 1X IBA (2.5 Gbps) for 10 years, and the average length of packets is 1,024 bytes. During the time, around 2^{61} random numbers will be generated. Therefore,

$$P_{collision} = 1 - e^{-(2^{61}(2^{61}-1))/2^{129}} \approx 0.0078.$$

Considering the normal amount of traffic, $P_{collision}$ will be far less than this approximation. Therefore, the proposed encryption scheme will be secure in the foreseeable future.

Type II attack. Any MAC algorithms using a 32-bit MAC can be used to remove a Type II attack. It is known that the security strength of GCM is the same as the strength of its block cipher AES [21], [7]. As Hellekalek et al. concluded that pseudorandom numbers generated by AES are indistinguishable from real random numbers [20], a 32-bit tag generated by GCM using AES is a near-random number. Therefore, on the average, after 2^{31} trials, a hacker can successfully make an authentic MAC without a legitimate key. In an IBA cluster with a 2.5-Gbps link, assuming that all packets are 1,024 bytes long, 2^{18} packets can be generated in 1 second. This means that it will take the hacker about less than 3 hours to spoof a packet. However, note that the attacker should send wrong MACs 2^{31} times, on the average, before he succeeds. These attacks can be detected by issuing an alert in case of consecutive authentication errors.

Type III attack. The DDPM, as described in Section 5.4, can trace back the original location of an attacking node, which is robust to fake addresses.

Type IV attack. The SIF blocks DoS attacks that are using illegal Keys.

6.2 Replay Attacks

In replay attacks, a hacker captures a packet carrying a legitimate MAC and keeps sending it to disrupt the network. Our authentication scheme is vulnerable to this attack because the proposed scheme is a per-packet authentication method. Therefore, repeated packets will be considered as authentic due to the legitimate MAC. Among possible approaches, time stamps can be deployed in IBA clusters [27]. In the time stamps scheme, a sending node inserts a time stamp into each packet, and a receiving node checks whether this time stamp is sufficiently close to the local time. For sure, the time stamps should be included for generating MACs to prevent false time stamps. There are two basic requirements to use this scheme. One is that the communication nodes should be synchronized, and the other is that each packet should have a space to carry a time stamp. In a cluster system, the global time synchronization is rather easy because time synchronization packets can be broadcast periodically with a very short delay. Concerning the space, as we mentioned in Section 5.4, the GRH can be used in every packet, and some bits out of an MF can be used to store a time stamp, as long as the rest of the MF bits support the network. For example, a 4,096-node mesh network requires 14 bits as an MF. Then, 24 bits can be used as a time stamp space, and it will wrap around in 194 days when the time granularity is second. If the network is huge, and the administrator is more concerned about the replay attack than Type III attack, then the source identification scheme may be disabled, and the whole 38 bits can be used for time stamps, which will make the wrap-around time thousands of years.

6.3 Tamper-Resistant Storage

As we assumed before, each device has a tamper-resistant storage. This storage is usually built by using battery-backed RAM (BBRAM) [28]. Its contents will be instantly zeroed by cutting off the power supply once a tamper is detected. FIPS 140-1 recommends BBRAM to store sensitive data [29]. The contents of BBRAM should be toggled periodically because if the same data is stored for a long time, then the contents can be imprinted into RAM. In addition, each device should be hardwired to keep any secret information on BBRAM only accessible through its SM. Finally, all the secret keys stored in the device's BBRAM will be protected against any software and physical attacks.

7 Performance Analysis

7.1 Simulation Testbed

Our IBA testbed includes packet-level switches and host CAs compliant with the IBA specification [8]. For our experiments, we simulated a 16-node network designed using five-port switches. To simulate the Single Data Rate (SDR) and Double Data Rate (DDR), we use high-bandwidth physical links of up to 60 Gbps (DDR 12X). For the performance analysis, we use the best effort traffic only.

7.2 Performance Slowdown by Encryption and Authentication

To see the impact of encryption/authentication scheme, we compare the end-to-end latency. All the delays of the security operations are added up to the total latency. A recent design of AES can encrypt $30 \sim 70$ Gbps by using 0.18 μm complementary metal-oxide semiconductor (CMOS), with 80 ns of initial latency [30], [31]. We call this initial latency the *AES latency*. This means that at the sender in Fig. 3a, it takes one AES latency to get $E[1]$ at the first encryption. The subsequent encryptions till $E[m]$ are pipelined with no delay. According to [7], a parallel implementation of $mult_H$ can keep up with any pipelined implementation of AES. Calculating Tag requires one more $mult_H$ latency because it should wait until the completion of $E[m]$ in our simulation. Therefore, we set the total additional overhead to 320 ns per packet. We assume that the lookup table time for the partition or QP-level secret key is negligible. If there are a huge number of QP-level secret keys, then it will be necessary to take the lookup time into account.

Fig. 7a shows the overall network performance slowdown incurred by encryption and authentication. *No Security* represents the original IBA configuration, without any security function enabled. In the *Security* configuration, all encryption/authentication are enabled. As the packet length increases from 64 bytes to 1,024 bytes, the average latency decreases proportionally from 12.4 percent to 5.5 percent, 2.6 percent, 1.2 percent, and 0.7 percent, since the additional delay is constant to the packet length.

To see the effect of our scheme on very high speed networks, we simulate up to 60 Gbps (DDR 12X IBA). To rule out the network congestion effect, we scaled down the injection rate in high-speed networks. The packet length is set to 256 bytes as default. As shown in Fig. 7b, the proportion of security overhead is increasing because the 320-ns AES latency is not changing, whereas the average network latency decreases. Even though our scheme incurs more performance overhead proportionally in high-speed IBA networks, the total additional latency is shorter than 1 μs, hopefully making our scheme still practical.

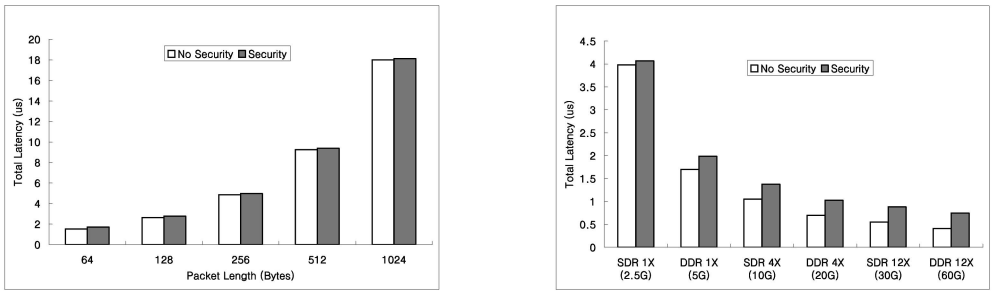

Fig. 7. Performance slowdown due to security operations. (a) Security configuration where both encryption and authentication are enabled incurs a relatively small overhead, ranging from 0.7 percent to 12.4 percent, compared to no security configuration. (b) As IBA network speed increases, the performance overhead of the security configuration also increases proportionally. However, since the additional overhead is still nanoseconds scale, the security configuration will be practical.

7.3 SIF Simulation

To investigate the effectiveness of the SIF, we compare the SIF with the DT and the IF. When DoS attacks are very rare or not happening, the SIF is definitely the most efficient because it induces no delay, whereas the DT and the IF require constant additional delays, regardless of the occurrence of DoS attacks. In the DT, all switch ports compare each packet's Key with the valid Keys stored in switches, whereas in the IF, only ingress switch ports do so. According to the IBA specification, each port can have at most 32,768 P_Keys, so the maximum size of memory to store all the P_Keys is 64 Kbytes because one P_Key is 16 bits long. Considering that the QP-level key management can be enabled, and the number of QPs is much larger than that of P_Keys, we assume that the size of storage for holding all Keys is 1,024 Kbytes in our simulation. According to a cache access model [32], a 1,024-Kbyte SRAM memory can be accessed within 5 ns. Therefore, the DT has a 5-ns additional delay at every switching port, whereas the IF has the delay only at ingress ports.

We simulate a DoS attack in the middle of the simulation and measure the average end-to-end delay in four configurations: Original, DT, IF, and SIF, as shown in Fig. 8a. The simulation consists of four sets of simulations, varying the input load from 40 percent to 70 percent. The first four bars in the figure are tested with an input load of 40 percent, and so forth. The first bar of each set shows the average end-to-end latency under a DoS attack in the original configuration. As the input load increases, the network performance is further deteriorating. This is because the DoS attack saturates the already heavily loaded network by injecting more traffic, thus resulting in significant delays of normal traffic. The second and third bars of each set show the average delay of the DT and the IF. Since they block all illegal traffic, their latencies are not affected by the attack. However, redundant P_Key lookup table operations incur some amount of delay. Especially, under the heavy traffic,

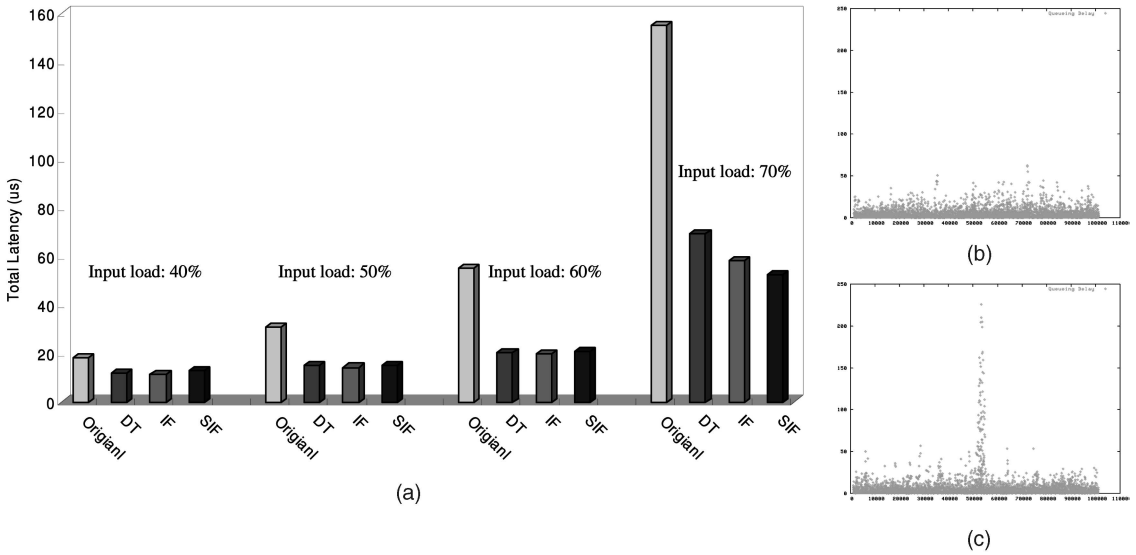

Fig. 8. Effect of SIF. (a) DT, IF, and SIF block DoS attacks successfully and show little difference in terms of total latency, except at the input load of 70 percent, where SIF shows the best performance. Note that since SIF is enabled only when there are active DoS attacks, SIF will show the best performance in normal situations. (b) and (c) show the distribution of end-to-end latency. Since IF blocks all DoS traffic from coming into the IBA network, there is no difference in the end-to-end latency, as shown in (b). However, SIF allows DoS traffic for a short time. Due to this, there are high spikes of the end-to-end latency in (c). All static methods, DPT and IF, have to incur a constant overhead, regardless of the occurrence of DoS attacks.

their filtering overhead results in a longer delay than the SIF. The fourth bar is showing the performance of SIF. Note that the SIF enables port filtering only when a DoS attack is active. Therefore, with SIF, there is a delay to register an illegal Key in appropriate ingress ports after detecting the illegal key. During the time, the attack affects other normal traffic even though its overhead is small. Fig. 8c shows that the SIF blocks the traffic instantly, but the delay results in high spikes of end-to-end latency in the middle of the simulation. In contrast, the IF in Fig. 8b does not have such variations. This explains that the average latencies of the SIF for the input loads with 50 percent and 60 percent are slightly longer than those of the DT and the IF.

However, even with this effect, the SIF in the 70 percent input load shows the best performance. This implies that the ingress filtering in the DT and the IF schemes can make a tangible effect on heavily loaded networks. Standard deviations of the SIF at 40 percent and 50 percent input loads are around 14 and 11, standing higher than those of the DT and the IF, around 5 and 8, due to the short-lived DoS attack. However, at the high load of up to 70 percent, standard deviations for all methods become much bigger because of high traffic.

8 Conclusion

This paper addresses a security provisioning framework for a secure IBA. We have argued that the security vulnerability inside IBA can be a serious problem and proposed a comprehensive framework to enhance IBA security in view of confidentiality, authentication, and availability. For confidentiality and authentication, we proposed the partition-level and QP-level secret key management schemes and showed how IBA accommodates GCM, with minor modifications to the IBA specification. For better availability, we proposed a SIF scheme, which is enabled only when there is a DoS attack by using invalid IBA Keys. The packet marking algorithms further improve the availability by identifying source nodes instantly.

The important conclusions of this work are the following. First, we elaborated on the IBA Key exposure problem and simulated possible DoS attacks inside IBA cluster systems. Simulation results showed that the overall performance can be affected significantly. Second, we also showed that the overall performance overhead by adopting an encryption and authentication algorithm into IBA clusters can be as low as 0.7 percent. Considering the security strength of GCM with AES, our scheme improves the security of IBA significantly with marginal performance degradation. Third, our SIF incurs a low performance overhead by being active only when necessary. Furthermore, our source identification algorithm successfully identifies real attackers in large irregular networks and huge regular networks having 2^{36} nodes with a 2D mesh and torus topology or 2^{38} nodes with a hypercube topology.

We are currently examining a number of possible extensions of this work. First, we would like to implement our ideas and build a real testbed. Second, we will extend our research to other cluster interconnects like Quadrics, Myrinet, and Gigabit Ethernet. Since they have common constraints such as the speed limitation on security operations, the hardware approach used in this research will be applicable to other interconnects. Third, in a very large-sized IBA cluster, the number of QP-level secret keys become so big that its lookup time is expected to become longer. For this, we are investigating a fast secret key management with architectural support. Last, since the OS can get much benefit from the proposed secure IBA for better cluster security, we plan to investigate this possibility.

References

[1] W.R. Cheswick, S.M. Bellovin, and A.D. Rubin, *Firewalls and Internet Security; Repelling the Wily Hacker,* second ed. Addison-Wesley, 2003.

[2] D. Geer, "Just How Secure Are Security Products," *Computer,* vol. 37, no. 6, pp. 14-16, 2004.

[3] A. Wool, "A Quantitative Study of Firewall Configuration Errors," *Computer,* vol. 37, no. 6, pp. 62-67, 2004.

[4] *HPC Wire,* http://news.taborcommunications.com/msgget.jsp?mid=506904, 2005.

[5] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and W.-K. Su, "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro,* vol. 15, no. 1, pp. 29-36, 1995.

[6] *InfiniBand Architecture Specification,* vol. 1, Release 1.1, InfiniBand Trade Assoc., 2002.

[7] D. McGrew and J. Viega, *The Galois/Counter Mode of Operation (GCM),* NIST Modes of Operation Process, 2004.

[8] E.J. Kim, K.H. Yum, C.R. Das, M.S. Yousif, and J. Duato, "Performance Enhancement Techniques for InfiniBand Architecture," *Proc. Ninth Int'l Symp. High-Performance Computer Architecture (HPCA '03),* pp. 253-262, 2003.

[9] W. Yurcik, G.A. Koenig, X. Meng, and J. Greenseid, "Cluster Security as a Unique Problem with Emergent Properties: Issues and Techniques," *Proc. Eighth LCI Int'l Conf. High-Performance Clustered Computing (Linux Revolution '04),* 2004.

[10] M. Pourzandi, "A New Distributed Security Model for Linux Clusters," *Proc. 2004 Usenix Ann. Technical Conf.: Extreme Linux Special Interest Group (Usenix '04),* pp. 231-236, 2004.

[11] Distributed Security Infrastructure, http://disec.sourceforge.net/, 2002.

[12] I. Foster, N. Karonis, N. Kesselman, and C. Tuecke, "Managing Security in High-Performance Distributed Computing," *High-Performance Distributed Computing Cluster Computing,* vol. 1, no. 1, pp. 95-107, 1998.

[13] K. Connelly and A.A. Chien, "Breaking the Barriers: High Performance Security for High Performance Computing," *Proc. Workshop New Security Paradigms (NSPW '02),* pp. 36-42, 2002.

[14] R. Dimitrov and M. Gleeson, "Challenges and New Technologies for Addressing Security in High Performance Distributed Environments," *Proc. 21st Nat'l Information Systems Security Conf. (NISSC '98),* pp. 457-468, 1998.

[15] M. Lee, E.J. Kim, K.H. Yum, and M. Yousif, "An Overview of Security Issues in Cluster Interconnects," *Proc. Second Int'l Workshop Cluster Security (Cluster-Sec '06)/Sixth IEEE Symp. Cluster Computing and the Grid (CCGrid '06),* p. 25, 2006.

[16] M. Lee, E.J. Kim, and M. Yousif, "Security Enhancement in InfiniBand Architecture," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* 2005.

[17] H. Lipmaa, P. Rogaway, and D. Wagner, "CTR-Mode Encryption," *Proc. NIST Workshop Symmetric Key Block Cipher Modes of Operation,* 2000.

[18] "National Information Assurance Glossary," Committee on Nat'l Security Systems, http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf, June 2006

[19] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C,* second ed. John Wiley & Sons, 1995.

[20] P. Hellekalek and S. Wegenkittl, "Empirical Evidence Concerning AES," *ACM Trans. Modeling and Computer Simulation,* vol. 13, no. 4, pp. 322-333, 2003.

[21] D. McGrew and J. Viega, *Flexible and Efficient Message Authentication in Hardware and Software,* http://www.zork.org/gcm/gcm-paper.pdf, 2003.

[22] H. Aljifri, "IP Traceback: A New Denial-of-Service Deterrent?" *IEEE Security and Privacy,* vol. 1, no. 3, pp. 24-31, 2003.

[23] D.X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM '01*, pp. 878-886, 2001.
[24] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM '00*, pp. 295-306, 2000.
[25] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proc. IEEE Symp. Security and Privacy (SP '03)*, p. 93, 2003.
[26] W. Feller, *An Introduction to Probability Theory and Its Applications*, third ed., vol. 1. John Wiley & Sons, 1968.
[27] D.E. Denning and G.M. Sacco, "Timestamps in Key Distribution Protocols," *Comm. ACM*, vol. 24, no. 8, pp. 533-536, 1981.
[28] J. Dyer, R. Perez, S. Smith, and M. Lindemann, "Application Support Architecture for a High-Performance Programmable Secure Coprocessor," *Proc. 22nd Nat'l Information Systems Security Conf. (NISSC '99)*, Oct. 1999.
[29] *Security Requirements for Cryptographic Modules*, NIST Standard FIPS 140-1, 1994.
[30] A. Hodjat and I. Verbauwhede, "Minimum Area Cost for a 30 to 70 Gbits/s AES Processor," *Proc. IEEE CS Ann. Symp. VLSI*, pp. 83-88, 2004.
[31] Y. Zhang, L. Gao, J. Yang, X. Zhang, and R. Gupta, "SENSS: Security Enhancement to Symmetric Shared Memory Multiprocessors," *Proc. 11th Int'l Symp. High Performance Computer Architecture (HPCA '05)*, 2005.
[32] S. Wilton and N. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677-688, 1996.

Manhee Lee received the BE and ME degrees from the Department of Computer Engineering, Kyungpook National University, Korea, in 1995 and 1997, respectively. He is currently a PhD student in the Department of Computer Science, Texas A&M University. He worked as a network researcher at the Korea Institute of Science and Technology Information for seven years. His research interests include computer architecture, parallel/distributed systems, secure computing architecture, secure cluster computing, Internet security, and computer networks. He is a student member of the IEEE and the ACM.

Eun Jung Kim received the BS degree in computer science from the Korea Advanced Institute of Science and Technology, Korea, in 1989, the MS degree in computer science from the Pohang University of Science and Technology, Korea, in 1994, and the PhD degree in computer science and engineering from the Pennsylvania State University in 2003. She is an assistant professor in the Department of Computer Science, Texas A&M University. From 1994 to 1997, she worked as a member of the technical staff in the Korea Telecom Research and Development Group. Her research interests include computer architecture, parallel/distributed systems, low-power design, secure computing, performance evaluation, and fault-tolerant computing. She is a member of the IEEE Computer Society and the ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.