Security Enhancement in InfiniBand Architecture

Manhee Lee*   Eun Jung Kim*   Mazin Yousif †

*Department of Computer Science
Texas A&M University
College Station, TX-77840
manheelee@tamu.edu, ejkim@cs.tamu.edu

†Corporate Technology Group
Intel Corporation
Hillsboro, OR 97124
mazin.s.yousif@intel.com

Abstract

The InfiniBand^{TM} Architecture (IBA) is a new promising I/O communication standard positioned for building clusters and System Area Networks (SANs). However, the IBA specification has left out security resulting in potential security vulnerabilities, which could be exploited with moderate effort. In this paper, we view these vulnerabilities from three classical security aspects: availability, confidentiality, and authentication. For better availability of IBA, we recommend that a switch be able to enforce partitioning for data packets for which we propose an efficient implementation method using trap messages. For confidentiality, we encrypt only secret keys to minimize performance degradation. The most serious vulnerability in IBA is authentication since IBA authenticates packets solely by checking the existence of plaintext keys in the packet. In this paper, we propose a new authentication mechanism that treats the Invariant CRC (ICRC) field as an authentication tag, which is compatible with current IBA specification. When analyzing the performance of our authentication approach along with other authentication algorithms, we observe that our approach dramatically enhances IBA's authentication capability without hampering IBA performance benefit. Furthermore, simulation results indicate that our methods enhance security in IBA with marginal performance overhead.

1. Introduction

To achieve high performance and high availability computing, computer clustering is a popular trend observed in academia as well as in the industry. 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 come to the forefront and subsequently the design of secure clusters have recently surfaced as a critical issue.

To defend against attacks, clusters usually rely on *firewalls* or Intrusion Detection Systems (IDS), which although can defend against a considerable number of attacks, but they cannot prevent all possible threats for the following two reasons that were recently reported [8, 27, 30]. The first is that firewalls or IDSs, themselves, could have vulnerabilities. Geer reported that several products of well-known security companies such as Check Point Software Technologies, Symantec, and Zone Labs have "potentially dangerous flaws that could let hackers gain control of systems, disable computers, or cause other problems" [8]. Wool quantified configuration errors of Check Point FireWall-1 installed in several sites, where he surprisingly found that most sites have many errors [27]. Almost 80 percent of firewalls allow both "Any" service and insecure access to firewalls, meaning that they could have been broken into easily. The second reason is that if hackers somehow get legitimate user passwords (Note that many still use simple passwords for authentication), they can enter a system with no difficulty. For example, a hacker recently infiltrated and compromised many computers in the supercomputer networks at Stanford University after getting a legitimate user password [30].

Once a hacker breaks into a cluster, the impact of attack within the cluster would be severe. That is because one infected system, believed to be trustworthy, may instantly paralyze the whole cluster through the high speed network. In addition, high bandwidth connections, extensive computational power, and massive storage capacity of a cluster can be used for another attack and as a repository for illegal contents [28]. Therefore, to prevent or mitigate these vulnerabilities, it is imperative to incorporate more security functionalities into clusters to harden them against security attacks.

Some researchers have proposed new methods to improve security inside clusters. Yurcik *et al.* introduced a new concept called "emergent security properties" to identify security characteristics unique to a cluster, and

which can be used to develop a unified monitoring tool for a cluster [28]. Foster *et al.* proposed a communication library allowing programmers to communicate securely in geographically distributed computing environments [11], which suitably provides well-organized security in Grid. Connelly and Chien focused on incorporating confidentiality in remote procedure calls in tightly coupled components applications. They applied traditional security functions such as transposition, substitution, and data padding on the marshalling layer [4]. Their performance analysis showed that encryption can be used in clusters with minimal performance impact. However, since we focus on IBA's authentication vulnerability, a new security design that can be deployed in IBA environments is researched here. Dmitriov and Gleeson presented security enhancement methods at three levels: network host interfaces, SANs, and communication protocols for interconnecting SANs [6]. Their approach is systematic enough to be considered a guideline for enhancing security of Myrinet- or VIA-based clusters, however, it is not compatible with the most recent IBA specification. To our knowledge, there has been no research on rectifying security vulnerabilities on IBA. In this paper, we attempt to first identify potential security threats to IBA and then suggest means to remove those threats. Considering that IBA is expected to be widely used for designing high performance clusters, identifying and removing such threats is necessary. In particular, we attempt to address the following questions.

Q.1 *What are the most serious security issues in IBA?*
A.1 Denial of Service (DoS) attacks deplete network or system resources, thereby preventing these resources from properly functioning due to the large amount of data such attacks dump. IBA specifies a mechanism, referred to as *partitioning*, for grouping nodes to limit access control [36]. Partitioning enables several nodes to exclusively share some resources, forbidding other nodes not in the same partition to access them. However, an attacker on a compromised InfiniBand node can easily trigger a DoS attack by flooding packets with random partition keys in the InfiniBand network. Destination nodes will block those packets because they do not have legitimate partition keys. However, they have already gone through the network, incurring a significant delay to other legal traffic.

IBA provides an isolation and protection service using Keys. There are five types of Keys in IBA: Management Key (M_Key), Baseboard Management Key (B_Key), Partition Key (P_Key), Queue Key (Q_Key), and Memory Keys (L_Key and R_Key). However, since Keys in IBA are available in the message as plaintext, their security value is diminished. Once a hacker gets any of the above Keys, especially the M_Key, the hacker can cause great harm such as reconfiguring the network, disconnecting communicating nodes, etc.

Q.2 *Which mechanisms could mitigate DoS?*
A.2 One such mechanism is to incorporate a stateful partition enforcement mechanism in switches using trap messages. This will eliminate DoS attacks that violate partition policies. Furthermore, since it is enabled only when a partition violation occurs, it will incur minimal performance overhead on IBA.

Q.3 *Can we use security protocols such as IPSec to prevent the Key exposure problem? If not, is there an alternative?*
A.3 IPSec's encryption and authentication mechanism may prevent this problem [12]. However, to adopt IPSec, IBA will need to change its protocol format. We propose a new authentication mechanism with no modification to IBA packet formats and with marginal performance overhead. Our main idea is to use the Invariant CRC (ICRC) field in the IBA protocol as a Message Authentication Code (MAC). Since MAC provides message authentication as well as error detection, it can successfully remove the Key exposure problem in IBA.

Q.4 *If authentication is to be added, authentication keys are necessary. How can IBA generate and manage new keys?*
A.4 We propose two key management methods: Partition-level and Queue Pair (QP)-level. Both methods are compatible with IBA Key policy while providing finer security granularity than static key management.

Q.5 *How secure will IBA be after adding the proposed authentication functions and what is implementation overhead?*
A.5 The 32-bit authentication tag can improve security dramatically, up to 2^{-30} of forgery probability. It degrades performance to some degree, but analysis shows that it is possible for authentication functions to operate at IBA link speed.

The remainder of the paper is organized as follows: Section 2 explains security background and IBA security. We show a DoS attack simulation result and propose an efficient partition enforcement method in section 3. In section 4, we enumerate authentication vulnerabilities and suggest two efficient key management methods. Section 5 describes an authentication mechanism and its performance analysis. Simulation results are presented in Section 6. Finally we discuss our algorithm's limitation and conclusions in section 7 and 8, respectively.

2. Background

2.1. Availability

Generally any computer security architecture needs to offer three basic goals: availability, confidentiality, and

authentication (or integrity). Availability refers to the "timely and reliable access to data and information services for authorized users" [32]. Availability has become more important in the Internet age because of the increase in DoS attacks. First generation DoS attacks dump the huge number of packets on a specific target system greatly slowing down that system & network connected to it [5, 7]. Second generation DoS attacks are caused by worms or viruses [18, 22, 24, 31]. Even though these attacks do not target a specific system, they can tie up system or network resources. Defense against a DoS attack is processed as follows: First a monitoring function detects a DoS attack. Then if the source address is spoofed, the source identification function identifies the true source system. Finally, a blocking function blocks incoming traffic or stops the source system from sending traffic.

There has been little research on defending against DoS attack in clusters. One example is Yurcik's emergent security properties established to enumerate security characteristics that are unique to clusters [28]. Manhee *et al.* proposed a deterministic distance packet marking scheme to identify DoS attackers in cluster interconnects [16]. In this paper, we simulate a DoS attack that results from the IBA partitioning enforcement policy and explain how to block the attack.

2.2. Confidentiality

Confidentiality refers to restricting access to data sent by a sender only to the designated receiver. Encryption and decryption will be needed to enforce confidentiality. Encryption scrambles the original plaintext into ciphertext using an encryption key; while decryption recovers the original plaintext from the ciphertext using a decryption key. If the two keys are identical, this is called a symmetric or secret key mechanism. Otherwise, it is called an asymmetric or public key mechanism, where anyone can encrypt a message using a public key, but only the person with the private key can decrypt it [13].

In this paper, we propose data encryption only for key distribution in IBA, which will be discussed in section 4. Even though recently proposed cryptography functions aim to reduce overhead in terms of computation time and power, they still decrease network and system performance. We can infer the expected performance overhead on IBA by looking at IPSec. Elkeelany *et al.* and other researchers analyzed IPSec's encryption overhead and found it to be significant [9, 19]. To improve on software performance, there has been a trend to develop security processors such as CryptoManiac and other commercial products [17, 29, 35, 34]. For now, we limit our discussion to authentication.

2.3. Authentication

Authentication allows two parties to agree that a received message is authentic, not modified or forged. Message Authentication Code (MAC) is commonly used, which is also called a keyed one-way hash function [23]. The hash function maps a variable size message into a fixed length of digest. By checking the digest, the receiver can verify the authenticity of the message. To use the hash function as the authentication function, a secret key should be used as an input. By digesting the message along with a secret key, the MAC function makes an authentication tag. If the receiver has the same secret key, the receiver can make the same authentication tag. Well-known MACs are keyed MD5, HMAC-MD5, and HMAC-SHA1 [14, 26].

IBA provides an isolation and protection service using Keys. If a legitimate Key is stored in a packet, a receiver verifies that it is authentic. However, the Key can be exposed, which can cause a dangerous situation because once a hacker gets a Key, he can do much harm to IBA. To our knowledge, there has been no prior research on authentication in IBA. We propose an authentication mechanism for IBA in section 4 and 5.

3. Partition Enforcement in Switch

IBA specifies a partitioning scheme for access control. A partition includes Channel Adapters (CAs), routers, and switches' ports. Besides, a port may belong to one or more partitions at the same time. Only ports within the same partition can communicate with each other. The Host Channel Adapter (HCA) must implement a partition table containing a set of legally allowable P_Keys to enforce access control. A switch port, on the other hand, may optionally support partitioning enforcement. Note that although IBA uses partition enforcement, an attacker on a compromised IBA node can easily trigger a DoS attack by flooding packets. In this section, we show simulation results to demonstrate the harmfulness of DoS attack in IBA and then we propose a stateful partition enforcement mechanism in switches using trap messages.

3.1. Simulation Testbed

Our IBA testbed includes packet-level switches and HCAs compliant with the IBA specification. Table 1 shows the main parameters used for simulation. For our experiments, we simulated a 16-node mesh network designed using 5-port switches and an HCA. Two different workload traffics are used: realtime and best-effort. Realtime traffic is a continuous stream of packets with a higher priority than best-effort traffic. Since realtime traffic has minimal bandwidth requirements, an

application does not send any packet when the current network status cannot support the application's bandwidth requirement, and it also does not send faster than its predefined sending rate. Best-effort traffic is generated with a given injection rate and generally with Poisson distribution, which is similar to scientific workloads. Unlike realtime traffic, best-effort does not take current network conditions into considerations when sending packets. If the injection rate is too high relative to the current IBA network conditions, generated best-effort traffic will wait a long time before transmitted. In IBA, best-effort and realtime traffics do not interfere with each other because separate virtual lanes (VL) are allocated to different classes of traffic.

Table 1. IBA simulation testbed parameters

| Physical Link Bandwidth	2.5 Gbps
Number of Physical Links	5
Number of VLs/Physical Link	16
Realtime, Best-effort MTU	1024 Bytes

We simulate DoS attacks attempting to dump tremendous traffic into the IBA network. To make our simulation more realistic, we partition the IBA network into four random groups and allow the attacker to choose random nodes to attack. For simplicity, we assume each node has only one port. All nodes, other than the attacker node, send packets at a predefined rate to nodes in the same partition. The attacker node chooses destinations randomly and generates traffic at full speed (2.5Gbps since we simulated a 1x IBA link). This simulation is repeated increasing the number of attackers each time. The parameter of significance here is the queuing time, which is the waiting time that a packet waits before it gets served in HCA. In the real Internet, a packet is immediately transmitted but may suffer a long delay en-route, where queuing may happen in intermediate routers. However, since IBA uses credit-based flow control, the IBA network accepts a new packet only when there is available buffer. When the IBA network becomes congested, queuing time increases significantly while network latency increases marginally before a saturation point.

3.2. Simulation Result

Figure 1 (a) shows the difference in the average queuing time and network latency relative to the number of attackers. With no attacker, average queuing time is about five microseconds and network latency is about 20 microseconds. The Figure clearly shows that queuing time could increase by as much as 50 microseconds when only one attacker dumps traffic. As more attackers are added, the average queuing time increases up to 100

microseconds. However, network latency suffers marginal performance degradation.

With best-effort traffic and as the number of attackers increases, the average queuing time also increases, but more dramatically. Note that DoS attacks affect best-effort traffic more than realtime traffic because IBA's VL arbitration gives higher priority to realtime traffic.

(a) Realtime traffic   (b) Best-effort traffic
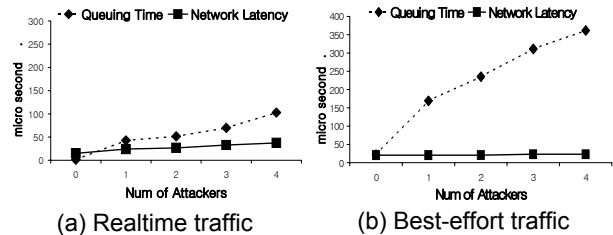
Figure 1. Average queuing time & network latency under DoS attacks

3.3. Stateful Partition Enforcement

To mitigate the problem discussed in section 3.2, we suggest extending the current partition enforcement in switches to filter all types of packets with an invalid P_Key. Three implementations are possible: *Duplicate Partition Table (DPT)*, *Ingress Filtering (IF)*, and *Stateful Ingress Filtering (SIF)*.

DPT is a straightforward approach, which refers to incorporating in each switch a partition table for all possible P_Keys that it might process and allow it to filter all traffic. However, this is very inefficient in terms of network performance and memory usage. The maximum amount of memory for the partition table and table lookup operations are shown in Table 2. The network consists of *n* nodes and *s* switches, and all nodes join the same number of partitions, *p*. We assume that a node is connected to only one switch for simplicity. $f(i)$ is the number of operations that table lookup function *f* needs to search one table which has *i* number of table entries.

Clearly, the DPT method incurs too much overhead. IF can increase the network and memory efficiency. The ingress switch that is directly connected to the end node only maintains a necessary & sufficient partition table. All switches do not have to have a partition table because if ingress switches filter all packets with the invalid P_Key, there will be no invalid packets in intermediate switches. Network performance and memory efficiency can be significantly increased as shown in Table 2. However, $f(p)$ is still necessary for every packet in IF, resulting in redundant operations if the valid P_Key are used most of the time.

To remove such redundant operations, we propose Stateful Ingress Filtering (SIF). More efficiency is obtained if filtering is enabled only when necessary. To

decide when to enable ingress filtering, our mechanism uses a trap message. In IBA, when an incoming packet's P_Key does not match with the receiver's P_Key, the receiver may send a trap message to the Subnet Manager (SM). Here, we suggest to use this trap message to find the right timing for ingress filtering. For this, we introduce an **Invalid_P_Key_Table** in switches. When the SM receives a trap message, it knows who sent the invalid P_Key packets and locates the switch it is connected to. SM can register the invalid P_Key to the Invalid_P_Key_Table of the switch, and then enable the switch's filtering function.

To disable this filtering when attacks end, we define **Ingress P_Key Violation Counter** in switches. This is similar to the P_Key Violation Counter defined in an IBA channel adapter for counting the number of invalid P_Key packets arriving at a node. Instead, the Ingress P_Key Violation Counter counts the number of invalid P_Key packets sent from the connected node. If this counter does not increase for some time, the switch disables ingress filtering by itself. Therefore, SIF is activated only in the case of invalid P_Key. Consequently, it can remove redundant filtering operations. The Invalid_P_Key_Table can grow bigger than its partition table as an attacker uses too many P_Keys. The Invalid_P_Key_Table table lookup time will become longer than the normal partition table lookup time. To prevent this situation, the Invalid_P_Key_Table should be used as long as the number of entries is smaller than the partition table.

SIF's overhead is shown in Table 2. $Pr(n)$ is the probability that one node joins a P_Key attack, and $Avg(p)$ is the average number of entries in the Invalid P_Key Table. In view of memory efficiency, it appears to double memory usage because it needs one more table. However, since DoS attacks are usually rare and result in a small number of compromised nodes, memory usage does not increase fast. Note that the table lookup operation per packet for SIF depends on the attack probability. IF and SIF show similar memory overhead, but SIF incurs practically no overhead on the table lookup time.

Table 2. Partition enforcement overhead

	DPT	IF	SIF
Memory for one switch	$n \times p$	p	$p + Pr(n) \times \mathrm{MIN}(Avg(p), p)$
Memory for all switches	$n \times p \times s$	$p \times n$	$p \times n + Pr(n) \times \mathrm{MIN}(Avg(p),p) \times n$
Table lookup operations/packet	$f(n \times p)$	$f(p)$	$Pr(n) \times f(\mathrm{MIN}(Avg(p), p))$

4. Authentication Key Management

4.1. InfiniBand Key Vulnerability

Even though IBA limits access to Keys, it may not be always successful because a packet can be captured on the link or CA or it is possible that a switch crashes and leaks Keys. Plaintext Keys in the packet might be exposed causing following vulnerabilities.

Table 3. IBA Key vulnerability

Key	Vulnerability
M_Key	Since M_Key controls almost everything in a subnet, leaking M_Key becomes a serious problem. Although SM assigns different M_Key for CA ports, it cannot prevent an M_Key exposure problem fundamentally.
B_Key	Since B_Key controls hardware of nodes and switch, a malicious user having B_Key can change hardware configuration.
P_Key	Any user acquiring a P_Key of a partition can break membership restriction of the partition. Some partition's information may be highly classified, so an exposure of P_Key itself can be a serious security problem.
Q_Key	If a Q_Key is exposed, the communication between two QPs may be disrupted or information may be ruined because the existence of Q_Key authenticates the packet. This is possible only when the partition's P_Key is available. In contrast to datagram service, connection–oriented service does not have Q_Key because its QPs are created to communicate with each other, so it is not necessary to differentiate packets from other QPs with Q_Key.
L_Key, R_Key	If R_Key is available, the memory can be read or written without any intervention of destination QP. That is possible because destination QP does not intervene Remote Direct Memory Access (RDMA) operations to increase performance. It can change memory contents of destination QP. Note that this is possible when both P_Key and Q_Key are available for datagram service. In case of connection-oriented service, this can happen when only P_Key is available.

4.2. Partition-Level Key Management

To remove the previously mentioned vulnerabilities, we propose that each packet has a MAC. To generate MACs, an authentication Key is necessary for each communicating entity. The authentication Key can be assigned to each partition: *Partition-Level Key Management*. When the SM creates a partition, it generates a secret key for that partition, and transfers it securely to the CA. Here, we assume SM knows public keys of all CAs and each CA can decrypt the secret key encrypted by the SM. Any QPs in the same partition can share a secret key, so they use the secret key to authenticate further communication among them. P_Key

is used to look up a secret key in the key table. For example, in Figure 2, node A and node B are in the same partition, Partition I, and node A and node C in the same partition, Partition II. P_K1, P_K2 are partition Keys and S_K1, S_K2 are their secret keys for partition I & II, respectively. If QP1, QP2 in Node A and QP1, QP2 in Node B want to communicate, they use S_K1. S_K2 is used between QP3 in Node A and QP1 in Node C. The P_Key exposure is not a problem since MACs are generated by its secret key not by P_Key. Therefore, the partition key threat is eliminated. In addition, since this key management does not use formal key exchange protocols such as Internet Key Exchange (IKE) protocol, its implementation will be simple [10]. One drawback of partition-level key management is that it trusts all QPs in the same partition, which means a hacker who gets a secret key of a partition can do harm to the partition.

4.3. Queue Pair-Level Key Management

For further security enhancement, an authentication Key can be assigned to each QP: *Queue Pair-Level Key Management*. QP management including Q_Key management takes place in the CA independently. This is much finer than partition-level because QP is the smallest communication entity in IBA. Here is an overview of QP and its transport service. When two consumers or user processes communicate in IBA, usually two (or possibly more) QPs communicate. QPs can communicate in one of two ways: connection-oriented or datagram (connection-less). In connection-oriented service, two QPs only communicate between each other. Since they cannot communicate with other QPs, packets only carry a P_Key; no Q_Key is included here. In datagram service, a QP can communicate with many other QPs. Therefore datagram packets should carry a Q_Key as well as P_Key
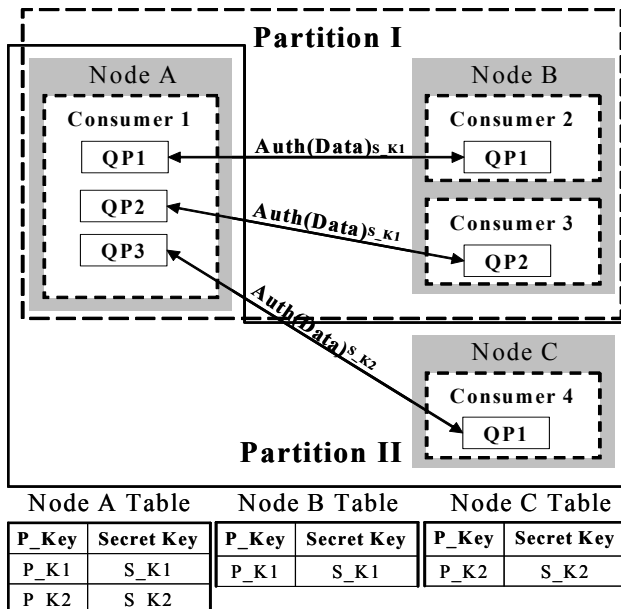
to show that the packet has a rightful privilege.

For two connection-oriented QPs to share a secret key, a QP that initiates the connection creates a secret key and sends it to a destination QP. Since we assume that each node has a table of public keys of other nodes, the key can be transferred securely. Further communication is signed and authenticated with the secret key. Note that secret key is generated and managed at the QP-level, but the key is distributed at the node level because it uses node-level encryption keys.

For datagram transport services, since a QP can communicate with multiple QPs, one QP may have many secret keys. Note that IBA enforces access right for datagram service using Q_Key. A datagram QP only accepts packets that have a legitimate Q_Key. If a QP wants to communicate with another datagram QP, it first sends a packet to request destination QP's Q_Key and receives it. Then it writes the Q_Key in every packet it sends. In our mechanism, a secret key is generated at every Q_Key request, which gets encrypted by the requester's public key before sending it. Indexing a secret key is a little bit different from partition-level key management because the receiver QP cannot index a secret key with only Q_Key. That is because the QP may issue multiple secret keys. In Figure 3, QP2 issues two secret keys, S_K2 and S_K3, to QP4 and QP5, respectively. Therefore in node A's table, Q_K2 has two entries of S_K2 and S_K3. Therefore, to index a secret key, both Q_Key and source QP are necessary.

Since QP-level key management has much finer granularity than partition-level key management, it helps remove the Memory Key threat in addition to eliminating a P_Key threat. That is because even if R_Key is exposed, QP-level key management guarantees authentic communication among legitimate QPs.
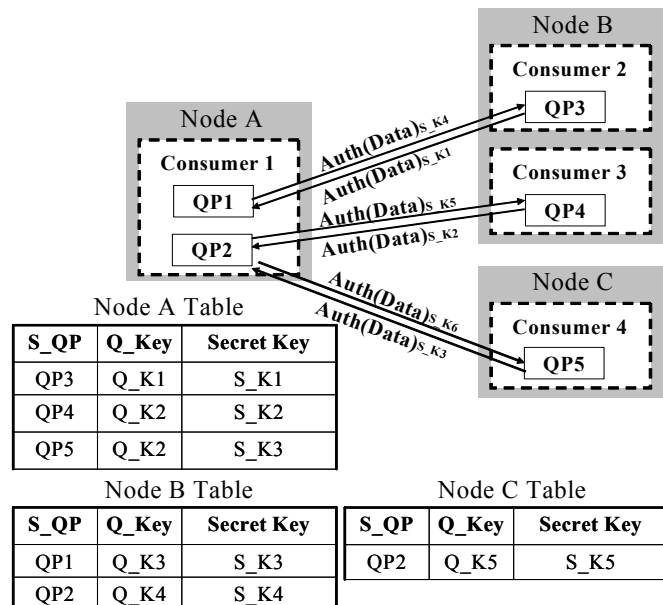

Figure 2. Partition-Level Key management

Node A Table

P_Key	Secret Key
P_K1	S_K1
P_K2	S_K2

Node B Table

P_Key	Secret Key
P_K1	S_K1

Node C Table

P_Key	Secret Key
P_K2	S_K2

Figure 3. QP-Level Key management

Node A Table

S_QP	Q_Key	Secret Key
QP3	Q_K1	S_K1
QP4	Q_K2	S_K2
QP5	Q_K2	S_K3

Node B Table

S_QP	Q_Key	Secret Key
QP1	Q_K3	S_K3
QP2	Q_K4	S_K4

Node C Table

S_QP	Q_Key	Secret Key
QP2	Q_K5	S_K5

5. Message Authentication in InfiniBand

5.1. Authentication Tag Field in InfiniBand

Each packet should carry an authentication tag called, Message Authentication Code (MAC). We propose to use the Invariant CRC (ICRC) field to store the Authentication Tag (AT). We only have to change the function to process the ICRC field.

IBA defines three types of CRC: Invariant CRC (ICRC), Variance CRC (VCRC), and Link Packet CRC (LPCRC) as shown in Figure 4 (a) (Only ICRC and VCRC are depicted). Cyclic Redundancy-Check (CRC) codes are widely used for error detection on the data communication [20]. ICRC covers all invariant fields from LRH to I Data. It does not cover variant fields that switches or routers can change. VCRC is present in all data packets and covers from LRH to the last byte before the VCRC. If some fields are changed in a switch or a router, VCRC is calculated again. LPCRC is present at the end of all Link packets. The only Link packet in current IBA specification is the flow control packet, so we do not consider LPCRC.
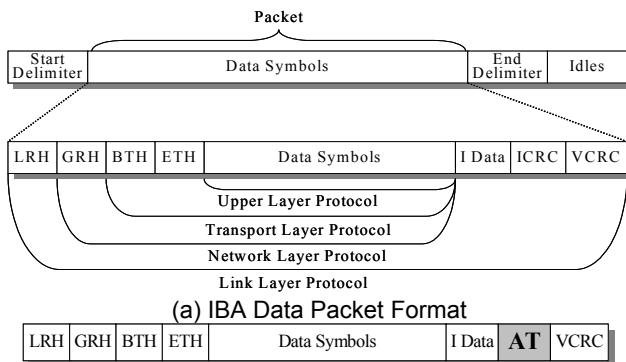

(a) IBA Data Packet Format

(b) Data Packet Format with authentication tag

Figure 4. Packet format & authentication tag [36]

Our main idea is to use the ICRC field for MAC as shown in Figure 4 (b). There are couple reasons for that. The first is ICRC does not change from end to end. It only covers unchangeable fields. Therefore, we can think of ICRC as transport-level CRC. Since authentication is an end-to-end transport level feature, the ICRC field is a nice fit for the AT. The second reason is that we do not have to change IBA packet format, which is extremely important. By leaving ICRC as the default option, our idea will become fully compatible with the current IBA. We use the authentication function only when necessary. Only the name of the ICRC field changes to AT. To accommodate various authentication functions, we can use *Reserved* field of Base Transport Header (BTH) for identifying which authentication function is used. BTH is a basic header that every transport packet carries. If the value is zero, the packet is using original ICRC. Non-zero value means an authentication function is in use and an AT is in the ICRC field. A predefined authentication function noted in Reserved field authenticates the message with an appropriate secret key indexed by P_Key or Q_Key and the source QP number.

Another advantage is that it is possible to provide an on-demand authentication service. For instance, let us assume that in some partition a very important job is running. The administrator can enable authentication only for that partition. Since the authentication can be disabled and enabled anytime, our mechanism provides very flexible authentication service.

5.2. Authentication Performance Analysis

A concern of enforcing authentication is performance. In this section, we describe several ways to implement authentication algorithms and compare their performance.

■ **Message Authentication Codes**

Here we compare four algorithms: CRC-32, HMAC-MD5, HMAC-SHA1, and UMAC. CRC-32 is an error-detecting algorithm used in IBA. The reason we compare MACs with CRC-32 is to demonstrate how much delay MACs incur to networking. HMAC-MD5 and HMAC-SHA1 are conventional MACs adopted in IPSec. Both of them are designed to provide high security. Even though they suffer from a performance problem, we include them because they are most widely used and IBA nodes may communicate with IPSec systems. Among many MAC algorithms, we chose UMAC due to its speed and proved security [2, 21]. Its speed comes mainly due to relying on Single Instruction Multiple Data (SIMD) parallelism from the Multi-Media eXtensions (MMX) instruction set. For security it uses universal hash-function family[1]. It is not easy to compare the above algorithms since their implementations vary. Nevertheless, by comparing some implementation results, we could get an idea of our authentication performance.

■ **Time Complexity**

A commercial product can generate CRC-32 codes at a 10Gbps rate using 32-bit multistage technology running at 312MHz as clock speed [33]. According to [2], SHA1 can be implemented at a 12.6 cycles/byte using a 250 MHz Pentium II PC. Since HMAC-SHA1 uses SHA1, the result can be used as the upper bound for HMAC-SHA1. Based on [3], Adcock estimates HMAC-MD5's upper bound would be 5.3 cycles/byte [1]. As follow-up research, Rogaway posted new implementation results for UMAC on his web page [21]. To generate a 4-byte authentication tag from a 1500 bits message, it takes 0.7 cycles/bytes with 32-bit technology in 700MHz Pentium

---

[1] This function is out of our scope. But further consideration is discussed in section 7.

III PC. Table 4 compares these results, which are normalized to 350 MHz. We assume their performance is proportional to clock speed. Even though UMAC is slower than CRC, it can be authenticated at Gbps speeds, enabling it to be used in IBA.

Table 4. Time & forgery complexity

Algorithm	Cycles/byte	Gbits/sec	Forgery Prob.
CRC	0.25	11.2	1
HMAC-SHA1	12.6	0.22	$\sim 2^{-32}$
HMAC-MD5	5.3	0.53	$\sim 2^{-32}$
UMAC-2/4	0.7	4.00	2^{-30}

■ **Forgery Probability**

In spite of performance disadvantage, adopting MAC is worthwhile because it can provide message authenticity as well as message integrity. CRC is just for checking transmission error, so forgery probability is virtually one. The security strength of HMAC depends on the underlying hash function. Since there has been no successful attack on MD5 and SHA1, the forgery probability of MD5 and SHA1 can be projected to 2^{-120} and 2^{-160} using their original tag size of 120-bit and 160-bit, respectively [1]. We assume that the security strength of two algorithms is proportional to their authentication tag sizes. The forgery probability of a 32-bit AT of two algorithms may be up to 2^{-32}, while UMAC provides provable security strength of 2^{-30} with a 32-bit AT [21]. This means that even if our authentication method uses a 32-bit tag, it can improve security strength dramatically. Therefore, using the ICRC field as an authentication tag is definitely worthwhile. The forgery analysis is also shown in Table 4.

6. Simulation Results

In section 3, we introduced our IBA testbed and presented one simulation result. That showed that the overall average queuing and network delay increased dramatically as attackers started to inject traffic into the IBA network at a high speed. In this section, we present several simulation results showing that SIF and authentication with Partition-level or QP-level key management can be implemented with marginal performance overhead.

The first simulation focuses on the effectiveness of SIF. According to IBA specification, each port can have at most 32768 P_Keys, and the maximum size of memory for storing all the P_Keys is 64KB because one P_Key is 16 bits long. According to the CACTI model, 1024KB SRAM memory can be accessed within 5ns [38]. Since this access time is similar to the current system bus speed, we can conservatively estimate that P_Key table access

time, $f(p)$, is one clock cycle. In DPT, P_Key table lookup operation should occur at every hop. To simulate DPT, we add one additional cycle at all ports in switches. In IF, only ingress ports need one more cycle for ingress filtering. In SIF, ingress filtering is enabled when an attack is present. One more cycle is added to the ports that are involved in an active DoS attack.

Figure 5 shows four sets of performance comparison among No filtering, DPT, IF, and SIF. Each set has a different input load from 40% to 70%. The first bar of each set is the average queuing and network delay of non-attacking traffic under a DoS attack by four attackers. The second bars of sets show the average delay when DPT is enabled. Since DPT blocks all illegal traffic, attacking traffic cannot get into the IBA network, but redundant P_Key table lookup operations incur some amount of delay. IF in the third bars show a better performance than DPT by removing those redundant operations from DPT.

The fourth bar is showing the performance of SIF. Since SIF is activated only when a DoS attack is active, SIF ideally should provide better performance than IF. However, in this simulation, it is slightly better than IF, and even at input load 40% and 50%, it is worse than IF. The reason is that we conservatively set the probability of DoS attack to 1%, which is significantly high considering current DoS attack duration in cluster. During the attacking period, SIF allows a DoS attack in the IBA network for a subnet manager to register the invalid
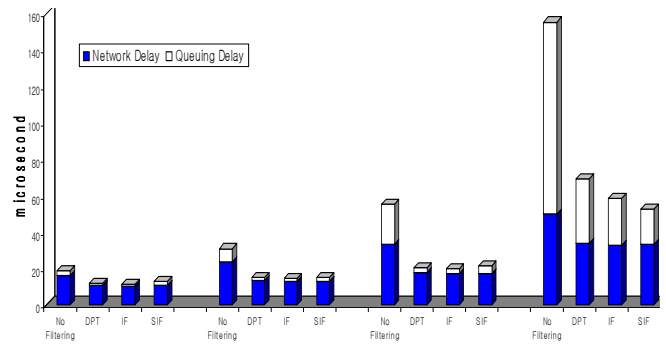
Figure 5. Performance comparison among No Filtering, DPT, IF, and SIF
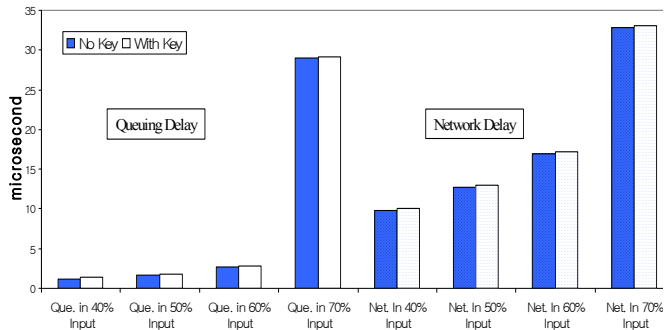
Figure 6. Message authentication overhead with Key Initialization

P_Key to a switch. Therefore, non-attacking traffic is affected by DoS traffic and an amount time is averaged throughout the simulation time. Due to this, standard deviations of SIF at 40% and 50% of input load, around 14 and 11, stand higher than those of DPT and IF, which are around 5 and 8. When input load increases up to 70%, standard deviations of other methods become bigger than that of SIF. This is because longer delay by high traffic and filtering causes high variance of queuing and network delay. If we exclude the attacking period, the overall delays of IF and SIF are 14.19μs and 13.65μs, respectively. If we increase the simulation time while decreasing the attack probability, we would get better performance. Additionally, in terms of power efficiency, SIF is better because more active stages in DPT and IF will definitely increase the power usage of switches.

The next simulation result in Figure 6 shows a performance overhead in Key management and authentication inside IBA network. If the Partition-level Key management is in use, Key distribution overhead is virtually zero because the SM distributes P_Keys and their secret keys first, and QPs do not have to receive them again until the partition is changed. However, in the QP-level Key management, Key exchange is frequent because a Q_key and its secret key are dynamically generated and should be available to the other side to authenticate messages. To simulate this, we add one round trip time delay for each pair of communicating QPs. A QP uses the same Q_Key for multiple messages, so its overhead is insignificant as shown in Figure 6. We assume its overhead is one clock cycle for each message even though it varies upon implementations. Like the previous figure, while standard deviations are as low as around 4 and 8 when input load is 40% and 50%, they increase significantly when input load increases to 60% and 70%. As analyzed in Table 4, UMAC can generate 1.4 bytes per cycle, which means that if we use 200MHz, UMAC can authenticate messages at the similar speed with IBA. This incurs one additional stage at each end node per message and pipelining can make this overhead negligible.

7. Discussions

■ **Limitations of UMAC & Fast Authentication**
We assume that UMAC can be implemented in the CA of IBA. When its inventors analyzed time complexity, they exploited SIMD parallelism available in the MMX instruction set of Intel processors. Therefore it may not be easy to implement UMAC in IBA directly. There are several ways to overcome this difficulty. First method is trading-off of security strength and MAC computing speed [1]. The idea is to digest a small part of the message to make the authentication tag. This will increase forgery

probability, but it will be better than CRC. Another is using Parallelizable Message Authentication Code (PMAC) under research [21]. Even though they do not provide a detailed analysis, it is expected to show good performance. NIST selected PMAC as one of the authentication modes of operation [37]. Another possible approach is to use a stream cipher MAC where MAC can be made while transferring data [15, 25]. Thereby, it can improve the network performance significantly.

Instead of using above techniques, the conventional MAC algorithms can be used for faster InfiniBand. [39] recently proposed a security processor which can encrypt/decrypt at 30 to 70 Gbps. Even though implementing the security processor in CA is not easy, its speed is comparable to IBA with regard to speed. We will study more on fast MAC.

■ **More DoS Attacks in InfiniBand**
Since the following attacks are still possible in IBA, more research is necessary.
- Dumping traffic only with a valid P_Key. Since this attack uses a valid P_Key, any ingress filtering is useless.
- DoS attack on the SM by dumping management messages and trap messages. Since a management packet can reach SM regardless of its partition, the attacker can dump management packets to slow down the SM and network.
- Replay attack. Attackers may capture a valid packet and replay the packet disrupting communications. This can be avoided by using timestamps or sequence numbers, referred to as nonce. Consecutive packets use different nonce, so the replayed packets will be found illegal. However, creation and management of nonce will be another overhead.

8. Conclusion

In this paper, we have argued that security vulnerability inside InfiniBand is a serious problem and proposed mechanisms to enhance InfiniBand security in view of availability, confidentiality, and authentication. First, we described availability threats and simulated DoS attack in InfiniBand. Surprisingly, the result shows that even one attacker can decrease network performance significantly; implying partition enforcement in switches can help to prevent DoS attack from exploiting the InfiniBand partition key. Then we proposed an efficient partition enforcement implementation method using stateful ingress filtering in switches. IBA can exert partition enforcement in switches with marginal performance degradation.

Second, we explained the current authentication problem and its threat. To eliminate this vulnerability, we also proposed an authentication mechanism. For managing authentication keys, we described two key

management methods: Partition-level and Queue Pair-level. Both methods are easy to implement in InfiniBand because they are well matched to current InfiniBand key policy. In addition, our method uses the ICRC field as authentication tag field without changing the IBA protocol format, making our method fully compatible with IBA protocol. Furthermore, our authentication mechanism can provide on-demand authentication service. In other words, authentication can be enabled only when it is necessary and only to the partition or some QPs.

Finally, we analyzed the performance overhead. While authentication functions decrease performance insignificantly, some authentication algorithms notably increase security strength with suboptimal forgery probability. This means that even if our method uses a small size tag, 32-bit, it can improve security strength sufficiently. Combined with the on-demand authentication method, this advantage helps InfiniBand provide more flexible service to users who have various security requirements. Then, we presented several simulation results to verify our suggestions. It was shown that our methods strengthen the security level of IBA considerably while it causes minor performance overhead.

9. References

[1] Adcock, J.M. Balenson, D.M. Carman, D.W. Heyman, M. Sherman, A.T., "Trading off strength and performance in network authentication: experience with the ACSA project," In *Proceedings of DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00.*

[2] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway, "UMAC: Fast and Secure Message Authentication," *Advances in Cryptology - CRYPTO '99*. Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 216-233.

[3] A. Bosselaers, "Even faster hashing on the Pentium," presented at the rump session of *Eurocrypto '97.*

[4] Kay Connelly and Andrew A. Chien, "Breaking the Barriers: High Performance Security for HighPerformance Computing," in *New Security Paradigms Workshop '02*, 2000.

[5] Sven Dietrich, Neil Long, and David Dittrich, "Analyzing distributed denial of service attack tools: The shaft case," in *14th Systems Administration Conference, LISA 2000*, 2000.

[6] Rossen Dimitrov and Matthew Gleeson, "Challenges and New Technologies for Addressing Security in High Performance Distributed Environments," in *Proceedings of the 21st National Information Systems Security Conference* 457-468.

[7] Dave Dittrich, "Distributed Denial of Service (DDoS) attacks/tools resource," http://staff.washington.edu/ittrich/ misc/ddos/, 2000.

[8] Geer, D. "Just How secure Are security Products?," *Computer*, Volume 37, Issue 6, Jun. 2004.

[9] O. Elkeelnay, M. Mtalgah, K. P. Sheikh, M. Thaker, G. Chaudhry, D. Medhi, J. Qaddour, "Performance Analysis of IPSec Protocol: Encryption and Authentication," In the *Proceeding of International Conference on Communication, 2002*, 2002.

[10] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)", November 1998, RFC 2409.

[11] Ian Foster, Nicholas Karonis, Carl Kesselman and Steven Tuecke, "Managing Security in High- Performance Distributed Computations," *Cluster Computing*, Vol 1, issue 1, pages 95-107, 1998.

[12] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," November 1998, RFC 2401.

[13] Kohnfelder, "Toward a Practical Public Key Cryptosystem," Bachelor's thesis, MIT Department of Electrical Engineering, May 1978

[14] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," February 1997, RFC 2104.

[15] X. Lai, R.A. Rueppel, and J. Woollven, "A Fast Cryptographic Checksum Algorithm Based on Stream Ciphers," *Advanced in Cryptology-AUSCRYPT '92 Proceedings*, Springer-Verlag, 1993, pp. 339-348.

[16] Manhee Lee, Eun Jung Kim, and Cheol Won Lee, "A Source Identification Scheme against DDoS Attacks in Cluster Interconnects," To be appeared in *Proceedings of International Workshop on Network Design and Architecture (IWNDA) 2004*, 2004.

[17] M. Lindemann and S. W. Smith, "Improving DES Coprocessor Throughput for Short Operations," in *Proceedings of the 10th USENIX Security Symposium*, pages 67-81, August 2001.

[18] A. Machie, J. Roculan, R. Russell, and M. V. Velzen, "Nimda Worm Analysis," Tech. Rep., Incident Analysis, SecurityFocus, Sept. 2001.

[19] Stefan Miltchev, Sotiris Ioannidis, "A Study of the Relative costs of Network Security Protocols," In *Proceedings of the USENIX Annual Technical Conference*, Freenix Track, pages 41-48, June 2002.

[20] W. W. Reterson and D.T.Brown, "Cyclic codes for error detection," *Proceedings IRE*, vol. 49, pp. 228-235, Jan. 1961

[21] Phillip Rogaway, UMAC: Fast and Secure Message Authentication, http://www.cs.ucdavis.edu/~rogaway/

[22] R. Russell and A. Machie, "Code Red II Worm," Tech. Rep., Incident Analysis, SecurityFocus, Aug. 2001.

[23] Bruce Scheneier, *Applied Cryptography: Protocol, Algorithms, and Source Code in C*, John Wiley & Sons, 1996, 2nd ed.

[24] D. Song, R. Malan, and R. Stone, "A Snapshot of Global Internet Worm Activity," Tech. Rep., Arbor Networks, Nov.2001.

[25] R. Taylor, "An Integrity Check Value Algorithm for Stream Ciphers," *Advances in Cryptology-CRYPTO '93 Proceedings*, Springer-Verlag, 1994, pp. 40-48.

[26] G. Tsudik, "Message Authentication with One-Way Hash Functions," *Proceedings of IEEE INFOCOM* 1992.

[27] Avishai Wool, "A Quantitative Study of Firewall Configuration Errors," *Computer*, Volume 37, Issue 6, Jun. 2004.

[28] William Yurcik, G.A.Koenig, X. Meng, J. Greenseid, "Cluster Security as a Unique Problem with Emergent Properties: Issues and Techniques," In *Proceedings of Linux Revolution 2004*, 2004.

[29] Lisa Wu, Chris Weaver, Todd Austin, "CryptoManiac: a fast flexible architecture for secure communication," *Proceedings of the 28th annual international symposium on Computer architecture (ISCA-2001)*, Jun. 2001.

[30] "Attackers infiltrating supercomputer networks," http://news.com.com/2100-7349-191024.html

[31] "CERT/CC, CERT Advisory CA-2001-26 Nimda Worm," http://www.cert.org/advisories/CA-2001-26.html, Sept. 2001.

[32] Committee on National Security Systems, "National Information Assurance Glossary," Available from http://www.nstissc.gov/

[33] CRC32 Generator/Checker for 10Gbps, http://www.morethanip.com/products/tgigeth/index.html

[34] HIPP Security Processor, http://www.hifn.com.

[35] IBM Cryptographic Coprocessor, http://www.ibm.com/security/cryptocards.

[36] InfiniBand Trade Association, "InfiniBand Architecture Specification, Volume 1, Release 1.1," November 2002. Available from http://www.infinibandta.org.

[37] Modes of Operation for Symmetric Key Block Cipher, NIST, http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/

[38] S. Wilton, N. Jouppi, "Cacti: An enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, May 1996. pp. 677-688.

[39] A. Hodjat, U. Verbauwhede, "Minimum area cost for a 30 to 70 Gbits/s AES processor," *IEEE Computer Society Annual Symposium on VLSI*, pp. 83-88, 2004.