ijecs open access www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN:2319-7242*
*Volume 2. Issue 9 September 2013 Page No. 2715-2720*

# An Efficient Twisted Edwards-Form Elliptic Curve for Fast Secured Message Passing Interface

***P.Ramyadevi[1], Dr. P Krishnakumari[2]***

[1]Research Scholar, Department of Computer Science,
RVS college of Arts and Science,
Coimbatore, Tamil Nadu, India
*ramyadevicc@gmail.com*

[2]Director, Department of Computer Applications (MCA),
RVS college of Arts and Science,
Coimbatore, Tamil Nadu, India
*kkumari@rvsgroup.com*

**Abstract:** ***Information processed in a distributed cluster is shared among a cluster of distributed tasks or users by the virtue of message passing protocols (e.g., Message Passing Interface MPI) or confidential data transmitted to and from cluster computing nodes. In a public network, when a number of clusters connected to each other is increased becomes a potential threat to security applications running on the clusters. To deal with this, a Message Passing Interface (MPI) is developed to preserve security services in an unsecured network. The proposed work focuses on MPI rather than other protocols because MPI is one of the most popular communication protocols on distributed clusters. Generally AES algorithm is used for encryption/decryption and interpolation polynomial algorithm is used for key management. But, in this research work, Twisted Edwards-Form Elliptic Curve Cryptography is used. This Twisted Edwards-Form Elliptic Curve Cryptography is integrated with Message Passing Interface Chameleon version 2 (MPICH2) with standard MPI interface that becomes ES-MPICH2. This approach provides better security with less overhead and fast when compared with the existing techniques.***

**Keywords:** Twisted Edwards-Form Elliptic Curve Cryptography, cluster, Message Passing Interface, High Performance Computing

## 1. Introduction

Due to the fast development of the internet and web, number of universities and companies are connecting their cluster computing systems to public networks to provide high performance. Those clusters connecting to the internet can be accessed by anyone from anywhere. Information processed in a distributed cluster is shared among a group of distributed tasks or users by the virtue of message passing protocols or confidential data transmitted to and from cluster computing nodes. Preserving data and information security in a message passing environment over an untrusted network is critical for a wide spectrum of security aware MPI applications. The unauthorized access to the security sensitive messages by untrusted process can lead to series security breaches. In this study, focus on MPI rather than other protocols. MPI is one of the most popular communication protocols for cluster computing.

MPICH2 developed by the Argonne National Lab. It is one of the important communication protocol for transferring data. The design goal of MPICH2 a widely used MPI implementation is to combine portability with high performance. Here integrated encryption and decryption algorithms into the MPICH2 library. Data confidentiality of MPI applications will be easily defended without a demand to change the program into the corresponding product version, after all provide a security enhanced MPI library with the standard message passing interface.

## 2. Related work

The Message Passing Interface (MPI) [2] is being wide accustomed develop parallel programs on computing systems like clusters. ES-MPICH2 [1] shows the MPI with enhanced security using AES and 3DES. This can be proved by a superfluity of MPI applications across several disciplines and activities, like natural philosophy, bioinformatics, forecasting, and money modeling [3]. As clusters continue to be a major component of High Performance Computing (HPC) environments [5], MPI is becoming increasingly prevalent. Despite the success of MPI, many MPI library implementations [4], [6], [7], [8] has suffered from software bugs (also referred to as software defects). As an example,

more than 2,000 bug tickets have been formed for various versions of Open MPI [9] since 2006. Same about 700 bug tickets have been reported for MPICH2 [4] since August 2008. The bugs in MPI libraries differently impact the productivity of an outsized range of users (users of MPI libraries refer to MPI application developers).

To detect and locate software bugs in MPI libraries [9]. To locate such a bug that occurs at users' sites, library developers have to be compelled to reproduce the bug at their own sites. This performance is formidable due to platform variations (e.g., architectures and system scales) between users' and developers' sites. Usually certain bugs solely occur on large-scale systems [10]. Much research has been conducted to detect software bugs in HPC systems at runtime. Among previous studies, many [11], [12], [13], [14], [15] focus on MPI applications. For example, Umpire [14] dynamically checks MPI function calls against certain rules such as "all members of one process group must execute collective operations over the same communicator in the same order." Similarly, Intel Message Checker [10] traces MPI calls at runtime and detects incorrect usage of MPI functions based on the traces. In recent years, researchers have explored temporal or spatial similarity exhibited in HPC systems for bug detection [16], [17]. The basic idea is to extract program runtime invariants [18], [19] within one process or across multiple processes, and to detect software system by identifying abnormal method behaviors.

## 3. Threats

A geographically distributed cluster system is one within which computing elements at local cluster computing platforms communicate and coordinate their actions by passing messages through public networks just like the web. To ingress the safety of clusters connected to the general public networks, one could build a personal network to attach an array of local clusters to create an outsized scale cluster. Building a personal network, however, is not a cheap way to product distributed clusters system. The Internet a large distributed system can be used to support large-scale cluster computing. Being a public network, the web becomes a possible threat to distributed cluster computing environments. Here first describe the confidentiality side of security in clusters followed by three specific attack instances.

### 3.1 Sniffing message traffic

Message traffic of associate MPI program is often sniffed. As an example, once MPCH2 is deployed in a cluster connected by a Gigabit local area network, hackers will sniff original text messages transmitted through the TCP protocol socket. Message sniffing can reveal metadata, security sensitive data and information.

### 3.2 Snooping on message buffer

In associate MPI program, buffers are engaged to send and receive information. Despite specific MPI implementations, message buffers are generated before the send and receive primitives are adjure. Hackers who snoop into the message buffers in memory can access information without being given specific access privileges.

### 3.3 Message traffic profiling

Message traffic profiling attacks look for to use time stamps, message size, message type and other metadata to investigate message exchange patterns and kinds of protocols being employed in message transmissions. As an example, an attacker and hackers can monitor the network connection of a cluster running associate MPI program. If a message has been frequently transmitted, the attacker will speculate the importance of the message and intercept the content of the message.

Confidentiality services will effectively counter the same threats in MPI applications running on clusters connected by a public network. During this analysis, the code messages encoded with the TEC (twisted Edwards form elliptic curve cryptography) algorithm.

## 4. Possible Approaches

There are three possible approaches to improving security of MPI applications. In initial approach, the application programmers can add source code to address the issue of message confidentiality. As an example, the programmers may await on external libraries to enforced secure applications. Alike an application-level security approach not only makes the MPI applications error prone, also reduces the flexibility and portability of the MPI applications. Then the second approach, MPI interface can be extended in the way that new security-aware APIs are designed and implemented (see, for example, MPI Sec I/O [23]). This MPI-interface-level solution enables programmers to write secure MPI applications with minimal changes to the interface. Admitting the second approach is higher than the initial one, MPI-interface level result typically requires an extra code to deal with data confidence. In the third approach a channel level result is proposed in this study to address the drawbacks of the above two approaches. Channel-level solution aims at providing information confidentiality in a communication channel that implements the Channel Interface 3 (CH3) in MPICH2 (see Fig. 1).

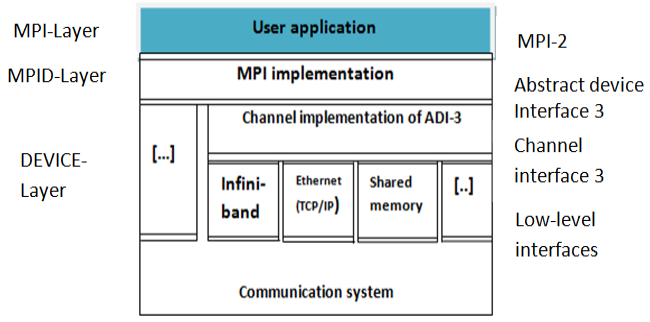


**Figure 1:** Hierarchical Structure of MPICH2

## 5. Existing System

In the existing system AES (Advanced encryption standard) and 3DES (Triple Data encryption standard) algorithm was used for the security purpose. Algorithm was integrated in the MPICH2 library for the encryption and decryption process. The Advanced Encryption Standard (AES) is formal encryption method adopted by the National Institute of Standards and Technology of the US Government, and is accepted worldwide. This paper introduces AES and key management, and discusses some important topics related to a good data security strategy.

The AES encryption algorithm is a block cipher that uses an encryption key and a several rounds of encryption. A block

cipher method is an encryption algorithm that works on a single block of information data at a time. Within the case of standard AES encipher the block is 128 bits, or 16 bytes, in key length.

Triple DES is the common name for the Triple Data Encryption Algorithm (TDEA or Triple DEA) block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.

### 5.1 Drawbacks of AES and 3DES

- Need for secure channel for secret key exchange: Sharing the secret key in the starting is a problem in symmetric key encipherment process. It has to be exchanged during a approach that ensures it remains secret.
- Too many keys: A new shared key has to be generated for communication with every different party. This makes a problem with managing and ensuring the security of all these keys.
- Origin and authenticity of message cannot be guaranteed: Since both sender and receiver use the same secret key, messages cannot be checked to have come from a particular user. This will be a problem if there is a dispute.

In the process of designing ES-MPICH2, to overcome the above said drawbacks, Twisted Edwards Curves (TEC) algorithm is integrated with the MPICH2 library.

## 6. The Design of ES-MPICH2

### 6.1 Scope of ES-MPICH2

Confidentiality, availability, integrity, and authentication are four necessary security problems to be addressed in clusters connected by an unsecured public network. Rather than addressing all the production problems, here pay specific attention to confidentiality services for messages passed among computing nodes in an unsecured cluster network. Although productive confidentiality is our primary concern, an integrity checking service may be promptly incorporated into our security framework by applying a public-key cryptography procedure. MPI framework equipped with the public-key theme, transmitting nodes can encrypt messages using their private keys. In the data receiving techniques, any nodes will use public keys corresponding to the private keys to decrypt messages. If anyone alters the data, the cipher text cannot be decrypt correctly using public keys corresponding to the private keys. Thus, the receiving nodes will perform message integrity check without the secure exchange of secret keys.

### 6.2 Design Issues

The goal of the development of the ES-MPICH2 mechanism is to enable application programmers to easily implement secure enhanced MPI applications without additional code for data-confidentiality protection. With ES-MPICH2 in place, secure MPI application programmers are able to flexibly choose a cryptographic algorithm, data block size and key size, for each MPI application that needs data confidentiality protection. ES-MPICH2 offers message confidentiality in an MPI programming environment by incorporating MPICH2 with encryption and decryption algorithms.

6.2.1 Message confidentiality

ES-MPICH2 plans to secure data confidentiality from unauthorized persons by untrusted processes.

6.2.2 Complete transparency

Product data confidentiality in MPICH2 is perfectly transparent to application programmers. That confidentiality transparency is suitable and the reason is two-fold. Initially, the encryption and decipher processes can be make in the MPICH2 library at the channel transmission layer. Second, control the same interface as the APIs of the MPICH2 operation. Therefore, it is not necessary to change MPI programs to adapt ES-MPICH2.

6.2.3 Compatibility and portability

ES-MPICH2 needs to be comfortably ported from one platform to another with no addition to the operation source code. ES-MPICH2 is a development of MPICH2. ES-MPICH2 must have the same level of portability and compatibility as MPICH2. However, it is difficult to achieve high portability in ES-MPICH2, because it has to implement a cryptographic subsystem in each channel in the CH3 layer in MPICH2.

6.2.4 Extensibility

ES-MPICH2 must permit application programmers to choose any cryptographic techniques and keys incorporated in MPICH2. This form goal makes it possible for programmers to flexibly select any cryptographic algorithm implemented in ES-MPICHI2.

## 7. Proposed TEC in MPICH2

Harold Edwards introduced a normal form for elliptic curves in July 2007 along with a simple symmetric addition law. Bernstein and Lange [21] established the connexion of Edwards' work for elliptic curve cryptography and came up with a lot of efficient formulas for point addition and doubling using standard projective coordinates. Here additionally extended Edwards' curve definition to a lot of general form that covers a much larger category of elliptic curves. In formal terms, a supposed Edwards curve over a prime field $F_p$ will be described by the equation[1]

$$E: x^2+y^2=1+dx^2y^2 \qquad (1)$$

With $d \in F_p \setminus \{0, 1\}$ Edwards curves have some attractive properties for practical use, most notably efficiency of the point arithmetic and completeness of the addition law when d is not a square in $F_p$. Completeness means the addition formula is valid for all $P, Q \in (F_{p)}$, including the special cases P = Q, P = -Q, P = 0 and Q = 0. Bernstein and Lange [21] also showed that every Edwards curve contains a point of order 4 and, thus, has a co-factor of $h \geq 4$. In 2008, Bernstein et al [20] introduced twisted Edwards's curves as a generalization of Edwards's curves. Formally, a twisted Edwards curve over a prime field Fp is defined via the equation

$$E: ax^2+y^2=1+dx^2y^2 \qquad (2)$$

Where 'a' and'd' are distinct, non-zero elements of $F_p$. Bernstein et al observed empirically that the twisted Edwards form covers much more curves than the "original" Edwards form2 based on Equation 1. Furthermore, as demonstrated in [20], every twisted Edwards curve over a non-binary field $F_q$ is birationally equivalent over $F_q$ to a Montgomery curve (i.e. every twisted Edwards curve can be transformed to a

Montgomery curve, and vice versa). Bernstein et al [20] also presented explicit formulas for addition and doubling on a twisted Edwards curve; these formulas are complete if 'a' is a square and d a non-square in the underlying field.

### 7.1 Advantages of TEC

- No Need for secure channel for secret key exchange.
- Too many keys. Low power and timing consumption.
- Origin and authenticity of message can be guaranteed
- Faster execution (i.e. take small time for execution in terms of iteration times).

## 8. Experimental Evaluation

To evaluate the performance and features of ES-MPICH2, enforced ES-MPICH2 and deployed it on two clusters with completely different configurations. The primary cluster has six nodes of 2.2 GHz Intel processors with two GB memory. The second cluster contains 10 nodes. The master node have a 3.0 GHz Intel Pentium Core two Duo processor with one GB memory, whereas the 9 slave nodes have 333 MHz Intel Pentium II processors with sixty four MB memory.

We run the Intel MPI Benchmarks and the Sandia Micro Benchmarks and to evaluate and compare the performance of MPICH2 and ES-MPICH2. When test ES-MPICH2 in each experiment, set the cryptographic service to TEC respectively. The length of data encryption keys generated and distributed in ES-MPICH2 is 192-bit.

### 8.1 SMB: Sandia Micro Benchmark

The Sandia National Laboratory developed the Sandia Micro Benchmark Suite (a.k.a., SMB) to evaluate and test high-performance network interfaces and protocols. The Table 1 described the four performance metrics used in the SMB benchmark suite.

**Table 1:**
Performance metrics in the Sandia micro benchmark suite

| Metric | Explanation |
|---|---|
| Iter -t | Total amount of time for the loop to complete. |
| work -t | For every repetition of the post work wait loop |
| overhead -t | The amount of work performed. |
| base-t | The length of time that a used processor is engaged in the reception or transmission of each message. |
| | The message transfer time arithmetic threshold. |

These metrics include total execution time (i.e., iter_t), CPU execution time for iterations (i.e., work_t), message passing overhead (i.e., overhead_t), and message transfer time calculation threshold (i.e., threshold or base_t). The detailed information on these metrics can be found at http://www.cs.sandia.gov/smb. Note that the message passing overhead can be derived by subtracting the CPU execution time from the total execution time. Each benchmark has 1,000 iterations. Fig. 2 shows the total execution time of the SMB benchmark running on the original MPI implementation (i.e., MPICH2), AES-based ES-MIPCH2, 3DESbased ES-MPICH2 and the proposed Twisted Elliptic Curve (TEC) based ES-MPICH2. It is observed from this figure that when the message size is small (e.g., 2 KB), the performance of ESMPICH2 is very close to that of MPICH2. These results indicate that ES-MPICH2 can preserve confidentiality of small messages with negligible overhead.

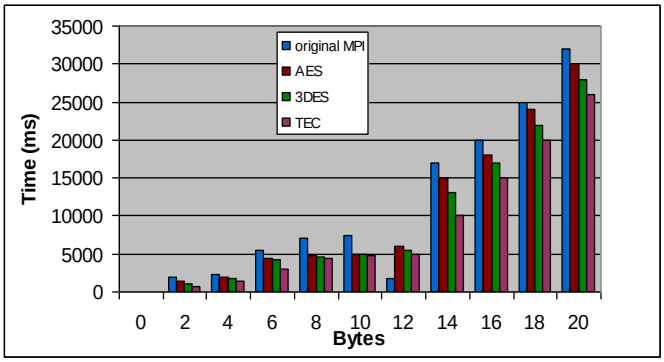


**Figure 2:** Comparison for original MPI, AES, 3DES, proposed TEC cryptography on different message size

Figs.2, 3 show the total execution time, overhead, CPU time and threshold of MPICH2 and ES-MPICH2 when the message size is 2, 16, 128, 512, and 1,024 KB, respectively. The results plotted in Fig. 3 show that the proposed TEC based ES-MPICH2 shows better performance when compared with the other techniques.

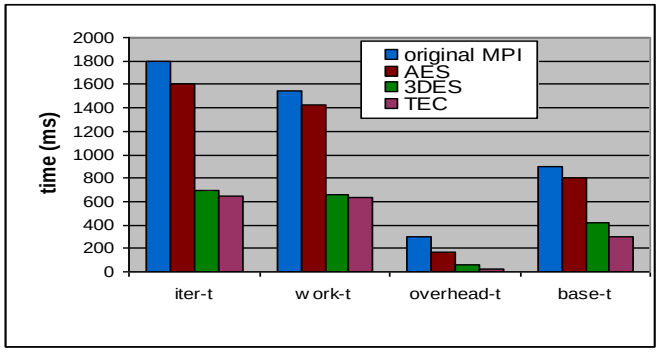


**Figure 3:** Iteration Time, Work Time, Overhead Time, and Base Time (Message Size is 2KB)

**Table 2:** Comparison table for proposed TEC with existing

| Metrics | Original MPI | AES | 3DES | Proposed TEC cryptography |
|---|---|---|---|---|
| Iter_t | 1800 | 1600 | 700 | 650 |
| work_t | 1550 | 1420 | 660 | 630 |
| overhead_t | 300 | 170 | 55 | 25 |
| base_t | 900 | 800 | 420 | 300 |

When the transferring message with size 2kb, the proposed TEC based ES-MPICH2 shows that better performance because metrics like total execution time of iteration time, overhead time, work time, and base time are low in TEC compared to other technique. In the proposed TEC technique the execution time will be less when compared to other technique. It shows in table 2.

### 8.2 IMB: Intel MPI Benchmarks

The Intel MPI benchmark suite or IMB was developed for testing and evaluating implementations of both MPI-1 and MPI-2 standards. IMB contains approximately 10,000 lines of code used to measure the performance of important MPI functions. We have evaluated the performance of ES-

MPICH2 and the original MPICH2 by running the benchmarks on the 6-node cluster.

The benchmarks in IMB-MPI1 can be categorized in three groups: single transfer bench mark, parallel transfer bench mark, and collective benchmarks. Single transfer benchmarks are viewing on a single message transferred between two contact processes. Dissimilar single transfer benchmarks, parallel transfer benchmarks scope at testing patterns and activities in a group of communicating processes with concurrent actions. The collective benchmarks are implemented to test higher level collective functions, which include processors in a defined communicator group. Please refer to http://software.intel.com/en-us/articles/intelmpi-benchmarks for more information concerning IMB.

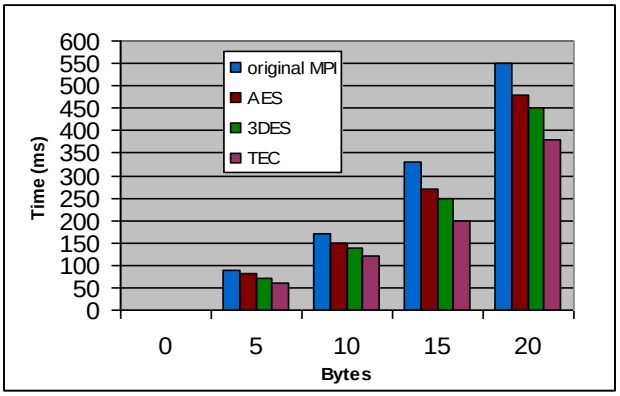


**Figure 4:** PingPong

Fig 4 Show the performance of PingPong a single transfer benchmarks in IMB. After all single transfer benchmarks are recycled to test a pair of two active processes, here run PingPong on two nodes of the cluster. The total execution times of PingPong go up when the message size increases because larger messages give rise to higher encryption and decryption overheads. Compared with MPICH2, AES-based and 3DESbased ES-MPICH2 the execution times of proposed TEC method are more sensitive to message size.

**Table 3:** Comparison table for PingPong

| Metrics | Original MPI | AES | 3DES | Proposed TEC cryptography |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 5 | 90 | 80 | 70 | 60 |
| 10 | 170 | 150 | 140 | 120 |
| 15 | 330 | 270 | 250 | 200 |
| 20 | 550 | 480 | 450 | 380 |

When comparing to original MPI, AES, 3DES technique the proposed TEC cryptography technique is best because it consume less time (ms) in bytes for decrypt and encrypt .In original TEC cryptography the total execution times of PingPong go up when the message size increases because larger messages give rise to higher encryption and decryption overheads. The performance result shows in table 3.

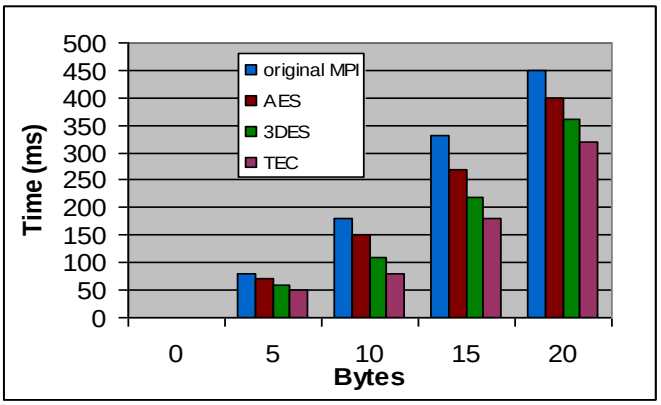


**Figure 5:** Sendrecv

When comparing to original MPI, AES, 3DES technique the proposed TEC cryptography technique is best. Because the total execution time of the SendRecv benchmark does not noticeably change when vary the number of computing nodes in the cluster. Original TEC consume less time. The performance of proposed TEC-based MPICH2 is close to that of the original version of MIPCH2 when message size is relatively small.

**Table 4:**
Comparison table for existing and proposed system

| Metrics | Original MPI | AES | 3DES | Proposed TEC cryptography |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 5 | 80 | 70 | 60 | 50 |
| 10 | 180 | 150 | 110 | 80 |
| 15 | 330 | 270 | 220 | 180 |
| 20 | 450 | 400 | 360 | 320 |

Now here analyze the performance of Sendrecv a parallel transfer benchmarks in IMB running on ES-MPICH2 and MPICH2 on the 6-node cluster in figure 5. Sendrecv, in which the main purpose is to test the MPI Sendrecv function, subsist of processes creating a periodic communication chain. This performance result values are in table 4.

## 9. Conclusion and Future work

### 9.1 Conclusion

MPI mechanism called ES-MPICH2 is implemented which offers both integrity services and data confidentiality for secure network communications in message passing environments. The proposed security technique included in the MPICH2 library is useful for protecting data transmitted in open networks like the Internet. In this paper, in order to increase the security, Twisted Elliptic curve cryptography is used. It provides better security and confidentiality. Moreover, the execution time taken by the proposed approach is very less when compared with AES and DES. The experimental results clearly show that the proposed approach provides better performance when compared with AES and DES.

### 9.2 Future work

An interesting direction for future work is to consider several strong and efficient cryptographic algorithms like the Elliptic Curve Cryptography in ES-MPICH2. After all

elliptic curve cryptography is an efficient and fast cryptographic result; both the performance and the productivity of ES-MPICH2 are likely to be improved by incorporating ECC. Another promising control for further work is to integrate encryption and decryption algorithms in other communication channels like SHMEM and InfiniBand in MPICH2 because an increasing number of commodity clusters are built using standalone and advanced networks such as Infiniband and Myrinet.

## REFERENCES


[1] Xiaojun ruan, "ES-MPICH2: A message passing interface with enhanced security" secure computing, vol.9, no.3, June 2012.

[2] "Message Passing Interface Forum," http://www.mpi-forum.org, 2012.

[3] "Papers about MPI," http://www.mcs.anl.Gov/research/ projects/mpi/papers, 2012.

[4] "Architecture Share in top 500 Supercomputers for 06/2009,"http://www.top500.org/stats/list/33/archtpe2012.

[5] "MPICH2: A High-Performance and Widely Portable Implementation of the Message Passing Interface (MPI) standard, "http://www.mcs.anl.gov/research/projects/mpich2, 2012.

[6] "MVAPICH2: MPI-2 over OpenFabrics-IB, OpenFabrics-iWARP, PSM, uDAPL and TCP/IP" http://mvapich.cse.ohiostate.edu/overview/mvapich2,2012.

[7] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," EuroPVM/MPI, 2004.

[8] J.M.Squyres and A. Lumsdaine, "A Component Architecture for LAM/MPI," Proc. EuroPVM/MPI, 2003.

[9] "Open MPI Bug Tickets," https://svn.open-mpi.org/trac/ompi/ ticket/689, 2012.

[10] D.C. Arnold, D.H. Ahn, B.R. de Supinski, G. Lee, B.P. Miller, and M. Schulz, "Stack Trace Analysis for Large Scale Debugging," Proc. IEEE Int'l Parallel and Distributed Processing Symp (IPDPS), 2007.

[11] J. DeSouza, B. Kuhn, B.R. de Supinski, V. Samofalov, S. Zheltov, and S. Bratanov, "Automated, Scalable Debugging of MPI Programs with Intel Message Checker," Proc. Second Int'l Workshop Software Eng" for High performance computing System Applications (SE-HPCS), 2005.

[12] T. Hilbrich, B.R. de Supinski: Schulz, and M.S. Muller, "A Graph based approach for MPI deadlock detection", Proc. 23rd Int'l Conf, supercomputing (ICS), 2009.

[13] B. Krammera, K. Bidmona, M.S. Muller, and M.M. Rescha, "MARMOT: An MPI Analysis and Checking Tool," Proc. Parallel Computing (PARCO), 2003.

[14] G. Luecke, H. Chen, J. Coyle, J. Hoekstra, M. Kraeva, and Y. Zou, "MPI-CHECK: A Tool for Checking Fortran 90 MPI Programs," Concurrency and Computation: Practice and Experience, vol. 15, no. 2, pp. 93-100, 2003.

[15] J.S. Vetter and B.R. de Supinski: "Dynamic Software testing of MPI applications with umpire," Proc. ACM/IEEE Conf, Supercomputing (CDROM), 2000.

[16] Q. GAO, F. Qin, and D.K. Panda, "DMTracker: Finding Bugs in Large-Scale Parallel Programs by Detecting Anomaly in Data Movements," Proc. ACM/IEEE Conf. Supercomputing, 2007.

[17] A.V. Mirgorodskiy, N. Maruyama, and B.P. Miller: "Problem Diagnosis in Large-Scale Computing Environments," Proc. ACM/ IEEE conf. Supercomputing, 2006.

[18] M.D. Ernst, J. Cockrell, W.G. Griswold, and D. Notkin, "Dynamically Discovering Likely Program Invariants to Support Program Evolution," Proc. 21st Int'l Conf. Software Eng. (ICSE), 1999.

[19] S. Hangal and M.S. Lam, "Tracking Down Software Bugs Using Automatic Anomaly Detection," Proc. 24th Int'l Conf. Software Eng. (ICSE), 2002.

[20] A. Petitet and R.C. Whaley, and J. Dongarra, and A. Cleary: "High performance Linpack", http://www.netlib.org/benchmark/hpl, 2012.

[21] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, Twisted Edwards's curves. In Progress in Cryptology — AFRICACRYPT 2008, vol. 5023 of Lecture Notes in Computer Science, pp. 389–405. Springer Verlag, 2008.

[22] D. J. Bernstein and T. Lange, Faster addition and doubling on elliptic curves. In Advances in Cryptology — ASIACRYPT 2007, vol. 4833 of Lecture Notes in Computer Science, pp. 29–50. Springer Verlag, 2007.

[23] R. Prabhakar, C. Patrick, and M. Kandemir, "MPISec I/O: Providing Data Confidentiality in MPI-I/O," Proc. IEEE/ACM Ninth Int'l Symp, Cluster Computing and the Grid, pp. 388-395, 2009.