ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Enhanced Security Mechanisms in MPI as ES-MPICH2

[1]D.Gangadhar Rao, [2]Balusa Anil Kumar
[1]Final M Tech Student, [2]Asst.Professor
[1,2]Dept of Computer Science and Engineering
[1,2]Krishna Chaitanya Institute of Technology & Sciences, Devarajugattu, Markapur-523320, Prakasam.

***Abstract*:** An increasing number of commodity clusters are connected to each other by public networks, which have become a potential threat to security sensitive parallel applications running on the clusters. To address this security issue, we developed a Message Passing Interface (MPI) implementation to preserve confidentiality of messages communicated among nodes of clusters in an unsecured network. We focus on MPI rather than other protocols, because MPI is one of the most popular communication protocols for parallel computing on clusters.

Our MPI implementation—called ES-MPICH2—was built based on MPICH2 developed by the Argonne National Laboratory. Like MPICH2, ES-MPICH2 aims at supporting a large variety of computation and communication platforms like commodity clusters and high-speed networks. We integrated encryption and decryption algorithms into the MPICH2 library with the standard MPI interface and; thus, data confidentiality of MPI applications can be readily preserved without a need to change the source codes of the MPI applications. The proposed work focuses on MPI rather than other protocols because MPI is one of the most popular communication protocols on distributed clusters.

Here AES, Triple DES and ECC algorithm is used for encryption/decryption algorithm is used for key management which is then integrated into Message Passing Interface Chameleon version 2 (MPICH2) with standard MPI interface that becomes ES-MPICH2. This ES-MPICH2 is a new MPI that provides security and authentication for distributed clusters which is unified into cryptographic and mathematical concept. The major desire of ES-MPICH2 is supporting a large variety of computation and communication platforms.

Keywords: Parallel computing, computer security, message passing interface, encryption/decryption.

I. INTRODUCTION

In unclustered networks, the data encryption for large scale distributed clusters becomes a non trivial and challenging problem, due to the open accessible nature of the internet. Information processed in a distributed cluster is shared among a group of distributed processes or users by virtue of Message Passing protocols (e.g. Message Passing Interface -MPI) running on the internet. To combine the portability with high performance the ES-MPICH2 with the original MPICH2 version is used for incurring the overhead by the confidentiality services. Due to high performance clusters, the security overhead can be reduced in ES-MPICH2. To preserve the data confidentiality, the encryption algorithm can be integrated into the MPICH2 library.

Due to the fast development of the internet, an increasing number of universities and companies are connecting their cluster computing systems to public networks to provide high accessibility. Those clusters linking to the internet can be accessed by anyone from anywhere. For example, computing nodes in a distributed cluster system proposed by Sun Microsystems are geographically deployed in various computing sites. 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.

When two entities are communicating with each other, and they do not want a third party to listen to their communication, then they want to pass on their message in such a way that nobody else could understand their message. This is known as communicating in a secure manner or secure communication. Fig 1 illustrates about the secure communication among distributed clusters [10].

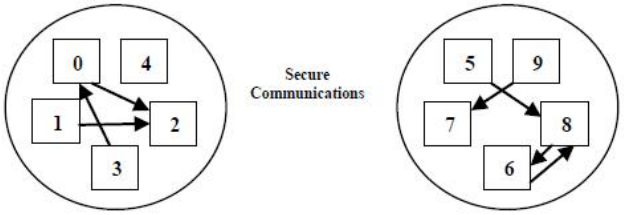

Fig. 1. MPI communication that includes all of the MPI processes with secure algorithms.

Preserving data confidentiality in a message passing environment over an untrusted network is critical for a wide spectrum of security-aware MPI applications, because unauthorized access to the security-sensitive messages by untrusted processes can lead to serious security breaches. Hence, it is imperative to protect confidentiality of messages exchanged among a group of trusted processes. It is a nontrivial and challenging problem to offer confidentiality services for large-scale distributed clusters, because there is an open accessible nature of the open networks. To address this issue, we enhanced the security of the MPI protocol by encrypting and decrypting messages sent and received among computing nodes.

In this study, we focus on MPI rather than other protocols, because MPI is one of the most popular communication protocols for cluster computing environments. Numerous scientific and commercial applications running on clusters were developed using the MPI protocol. Among a variety of MPI implementations, we picked MPICH2 developed by the Argonne National Laboratory. The design goal of MPICH2—a widely used MPI implementation— is to combine portability with high performance [14]. We integrated encryption algorithms into the MPICH2 library. Thus, data confidentiality of MPI applications can be readily preserved without a need to change the source codes of the MPI applications. Data communications of a conventional MPI program can be secured without converting the program into the corresponding secure version, since we provide a security enhanced MPI-library with the standard MPI interface.

It is a nontrivial and challenging problem to offer confidentiality services for large-scale distributed clusters, because there is an open accessible nature of the open networks. To address this issue, we enhanced the security of the MPI protocol by encrypting and decrypting messages sent and received among computing nodes. Numerous scientific and commercial applications running on clusters were developed using the MPI protocol. Among a variety of MPI implementations, we picked MPICH2 developed by the Argonne National Laboratory. The design goal of MPICH2—a widely used MPI implementation— is to combine portability with high performance. We integrated encryption algorithms into the MPICH2 library. Thus, data confidentiality of MPI applications can be readily preserved without a need to change the source codes of the MPI applications

A. Possible Approaches

There are three possible approaches to improving security of MPI applications. In first approach, application programmers can add source code to address the issue of message confidentiality. For example, the programmers may rely on external libraries (e.g., SEAL [3] and Nexus [2]) to implement secure applications. Such an application-level security approach not only makes the MPI applications error prone, but also reduces the portability and flexibility of the MPI applications. In the second approach, the MPI interface can be extended in the way that new security-aware APIs are designed and implemented .This MPI-interface-level solution enables programmers to write secure MPI applications with minimal changes to the interface. Although the second approach is better than the first one, this MPI-interface-level solution typically requires an extra code to deal with data confidentiality. The third approach—a channel-level solution—is proposed in this study to address the drawbacks of the above two approaches. Our channel-level solution aims at providing message confidentiality in a communication channel that implements the Channel Interface 3 (CH3) in MPICH2

B. Contributions

The three major contributions of this study includes

- We implemented a standard MPI mechanism called ES-MPICH2 to offer data confidentiality for secure network communications in message passing environments. Our proposed security technique incorporated in the MPICH2 library can be very useful for protecting data transmitted in open networks like the Internet.
- The ES-MPICH2 mechanism allows application programmers to easily write secure MPI applications without additional code for data-confidentiality protection. We seek a channel-level solution in which encryption and decryption functions are included into the MPICH2 library. Our ES-MPICH2 maintains a standard MPI interface to exchange messages while preserving data confidentiality.
- The implemented ES-MPICH2 framework provides Secured configuration file that enables application programmers to selectively choose any cryptographic algorithm and symmetric-key in ES-MPICH2. This feature makes it possible for programmers to easily and fully control the security services incorporated in the MPICHI2 library. To demonstrate this feature, we implemented the AES, 3DES and ECC algorithms in ESMPICH2. We also show in this paper how to add other cryptographic algorithms into the ES-MPICH2 framework.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

II. MPICH2 OVERVIEW

MPICH—one of the most popular MPI implementations—- were developed at the Argonne National Laboratory [4]. The early MPICH version supports the MPI-1 standard. MPICH2—a successor of MPICH—not only provides support for the MPI-1 standard, but also facilitates the new MPI-2 standard, which specifies functionalities like one-sided communication, dynamic process management, and MPI I/O [13]. Compared with the implementation of MPICH, MPICH2 was completely redesigned and developed to achieve high performance, maximum flexibility, and good portability.

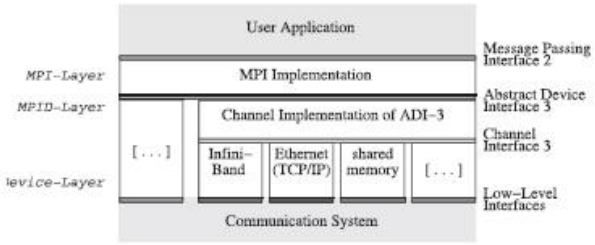

Fig. 2. Hierarchical Structure of MPICH2 [13].

Fig. 2 shows the hierarchical structure of the MPICH2 implementation, where there are four distinct layers of interfaces to make the MPICH2 design portable and flexible. The four layers, from top to bottom, are the message passing interface 2 (MPI-2), the abstract device interface (ADI3), the CH3, and the low-level interface. ADI3—the third generation of the abstract device interface— in the hierarchical structure (see Fig. 2) allows MPICH2 to be easily ported from one platform to another. Since it is nontrivial to implement ADI3 as a full-featured abstract device interface with many functions, the CH3 layer simply implements a dozen functions in ADI3 [1].

As shown in Fig. 3, the TCP socket Channel, the shared memory access (SHMEM) channel, and the remote direct memory access (RDMA) channel are all implemented in the layer of CH3 to facilitate the ease of porting MPICH2 on various platforms. Note that each one of the aforementioned channels implements the CH3 interface for a corresponding communication architecture like TCP sockets, SHMEM, and RDMA. Unlike an ADI3 device, a channel is easy to implement since one only has to implement a dozen functions relevant for with the channel interface.

To address the issues of message snooping in the message passing environments on clusters, we seek to implement a standard MPI mechanism with confidentiality services to counter snooping threats in MPI programs running on a cluster connected an unsecured network. More specifically, we aim to implement cryptographic algorithms in the TCP socket channel in the CH3 layer of MPICH2 (see Fig. 3).

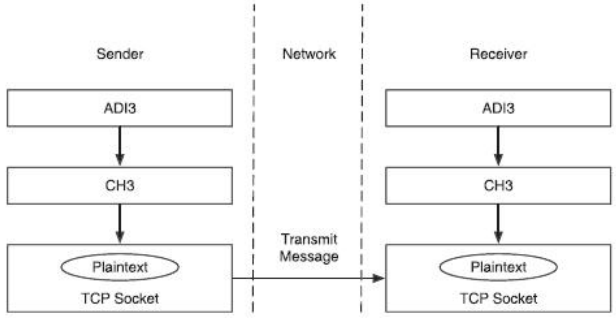

Fig. 3. Message passing implementation structure in MPICH2.

III. Related Work

Due to an increasing number of commodity clusters connected to each other by public networks, the encrypting and decrypting messages sent and received among computing nodes are not efficient and the data confidentiality is not readily preserved. So to implement secure applications, the programmers may rely on external libraries (e.g. SEAL [3] and NEXUX [2]).There is a minimal changes to the interface to write the secure MPI applications and the calculation time and memory needs for larger key sizes are more in the popular asymmetric cryptosystems like RSA.

IV. PROPOSED SYSTEM

To offer data confidentiality for secure network communications in message passing environments, a standard MPI mechanism called ES-MPICH2 was introduced. This proposed security technique incorporated in the MPICH2 library can be very useful for protecting data transmitted in open networks like the Internet. The ES- MPICH2 mechanism allows application programmers to easily write secure MPI applications without any additional code for data-confidentiality protection. We seek a channel-level solution in which encryption and decryption functions are included into the MPICH2 library. Thus the ES-MPICH2 maintains a standard MPI interface to exchange messages while preserving data confidentiality. ES-MPICH2 framework provides a secured configuration file that enables application programmers to selectively choose any cryptographic algorithm. It provides easy and full control of security services. AES, 3DES and ECC algorithms are used in ESMPICH2.

A. Scope of ES- MPICH2

Confidentiality, integrity, availability, and authentication are four important security issues to be

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

addressed in clusters connected by an unsecured public network. Rather than addressing all the security aspects, we pay particular attention to confidentiality services for messages passed among computing nodes in an unsecured cluster.

Although preserving confidentiality is our primary concern, an integrity checking service can be readily incorporated into our security framework by applying a public-key cryptography scheme. In an MPI framework equipped with the public-key scheme, sending nodes can encode messages using their private keys. In the message receiving procedure, any nodes can use public keys corresponding to the private keys to decode messages. If one alters the messages, the ciphertext cannot be deciphered correctly using public keys corresponding to the private keys. Thus, the receiving nodes can perform message integrity check without the secure exchange of secret keys.

B. Design structure of ES- MPICH2

One of the objectives in MPICH2 design is portability. To facilitate porting MPICH2 from one platform to another, MPICH2 uses ADI3 (the third generation of the Abstract Device Interface) to provide a portability layer. ADI3 is a full-featured abstract device interface and has many functions, so it is not a trivial task to implement all of them. To reduce the porting effort, MPICH2 introduces the CH3 interface. CH3 is a layer that implements the ADI3 functions, and provides an interface consisting of only a dozen functions. A ―channel‖ implements the CH3 interface. Channels exist for different communication architectures such as TCP sockets, SHMEM, etc. Because there are only a dozen functions associated with each channel interface, it is easier to implement a channel than the ADI3 device. The hierarchical structure of MPICH2, as shown in Figure1, gives much flexibility to implementers. The three interfaces (ADI3, CH3, and RDMA Channel Interface) provide different trade-offs between communication performance and ease of porting. As a successor of MPICH, MPICH2 [9] aims to support not only the MPI-1 standard, but also functionalities such as dynamic process management, one-sided communication and MPI I/O, which are specified in the MPI-2 standard.

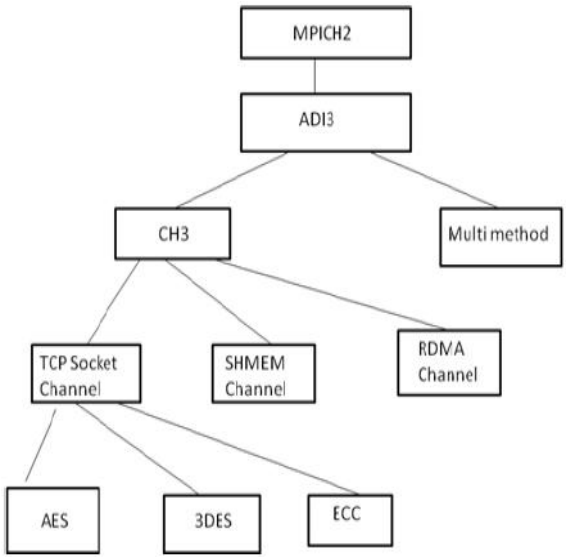

Fig 4. ES- MPICH2 implementation structure

However, MPICH2 is not merely MPICH with MPI-2 extensions. It is based on a completely new design, aiming to provide more performance, flexibility and portability than the original MPICH2. The future development for MPICH, including those necessary to accommodate extensions to the MPI Standard now being contemplated by the MPI Forum.

The process of creating a standard to enable portability of message-passing applications codes began at a workshop on Message Passing Standardization and the Message Passing Interface (MPI). Confidentiality, integrity, availability, and authentication are four important security issues to be addressed in clusters connected by an unsecured public network. Rather than addressing all the security aspects, we pay particular attention to confidentiality services for messages passed among computing nodes in an unsecured cluster. Although preserving confidentiality is our primary concern, an integrity checking service can be readily incorporated into our security framework by applying a public-key cryptography scheme.

In the implementation of ES-MPICH2 comprises of three main modules such as:

--Secure Login
--Key Generation
--Communication

A. Secure Login

The server verifies the authentication of the user and allows him to login his/her page. Thus, MPI application is developed to preserve authentication which is to be addressed in clusters connected by an unsecured network.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

B. Key Generation

Based on the number of users (communicate with each other) the interface of the server generate symmetric key and provide to the authorized clients in the clusters. It also provides rekeying system. Fig. 5 presents the key generation infrastructure in ES-MPICH2.

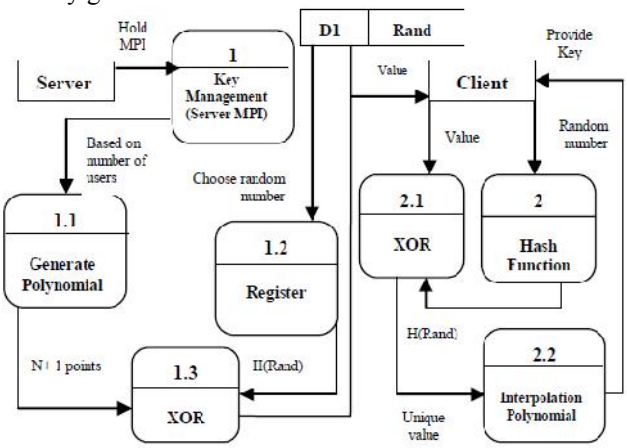

Fig. 5. Message passing implementation structure in ES-MPICH2 key generation details.

C. Communication

Before a message is delivered through the server, data contained in the message are encrypted by a cryptographic algorithms.

1. Key Agreement

In Key Agreement, the Diffie–Hellman key exchange algorithm is used that establishes a shared secret that can be used for secret communications by exchanging data over a public network. Figure 6 presents the key agreement infrastructure using Arithmetic XOR operations (e.g. 1 0→ 1). The terminal A and terminal B can share their secret keys by using hash functions. The positive response (e.g. 1) can be saved in the source side. The positive acknowledgement can be stored in the destination side. Thus the secret communications is done by accepting the secret key in the destination side and if it is matched send the acknowledgement to the destination side.

2. Advanced Encryption Standard

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the *state*, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

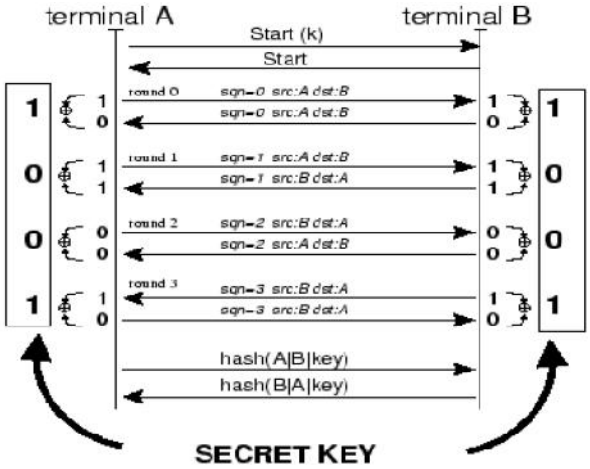

Fig 6. Key Agreement using XOR operations for sharing secret keys.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the cipher text. The numbers of cycles of repetition are as follows:

- 10 cycles of repetition for 128 bit keys.
- 12 cycles of repetition for 192 bit keys.
- 14 cycles of repetition for 256 bit keys.

Each round consists of several processing steps, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key. Figure 7 depicts the internal operations of Advanced Encryption Standard in which each round consists of four special functions. They are

- Byte Substitution
- Permutation
- Arithmetic operations over a finite field
- XOR with a key

These transformations are applied to a 128-bit input block in a certain sequence to perform an AES encryption or decryption. In both cases, the transformations are grouped to so-called rounds. There are three different types of rounds, namely, the initial round, the normal round, and the final round. AES is a symmetric block cipher. It acts as a resistance against all known attacks. In a wide range of platforms, the AES has speed and code compactness. The key unit is used to store keys and to calculate the key expansion function. Due to the fact that the AES is standardized for 128, 192, and 256-bit keys, the interface between the key unit and the data unit is designed for the key expansion for several different key

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

sizes can be implemented on the same chip. It is the preferred algorithm for implementations of cryptographic protocols.

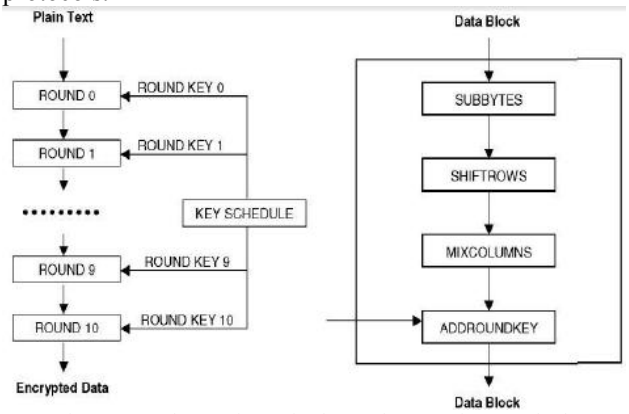

Fig 7. Internal operations of Advanced Encryption Standard.

3. Triple Data Encryption Standard

The original DES cipher's key size of 56 bits was generally sufficient when that algorithm was designed, but the availability of increasing computational power made brute-force attacks feasible. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm. Triple DES uses a "key bundle" which comprises three DES keys, K_1, K_2 and K_3, each of 56 bits (excluding parity bits). Figure 8 depicts the internal operation of Triple Data Encryption Standard in which it has three keys (56 x 3 = 168 bits) for encryption and decryption.

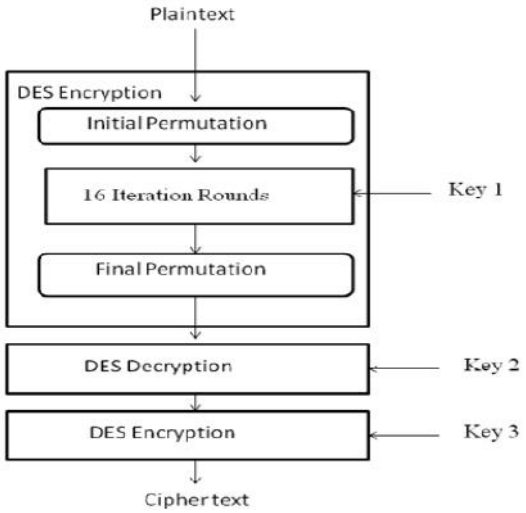

Fig 8 Internal operation of Triple Data Encryption Standard.

The encryption algorithm is: Cipher text = $E_{K3}(D_{K2}(E_{K1}(\text{plaintext})))$ i.e., DES encrypts with K_1, DES *decrypt* with K_2, then DES encrypt with K_3.Decryption is the reverse:

Plaintext = $D_{K1}(E_{K2}(D_{K3}(\text{cipher text})))$ i.e., decrypt with K_3, *encrypt* with K_2, and then decrypt with K_1.

4. Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Popular asymmetric cryptosystems like RSA are known to be very costly concerning calculation time and memory needs for larger key sizes. To solve this problem, elliptic curve cryptosystems based can be used. An elliptic curve cryptosystem is an asymmetric cryptosystem relying on the hardness of the discrete logarithm problem in elliptic curve groups With ECIES encryption, it is possible to encrypt data with a 160-bit key as secure as with RSA using a 1024-bit key. So ECIES encryption is the better choice compared to RSA when calculation time and available memory are restricted, e.g. when using cryptosystems on smartcards

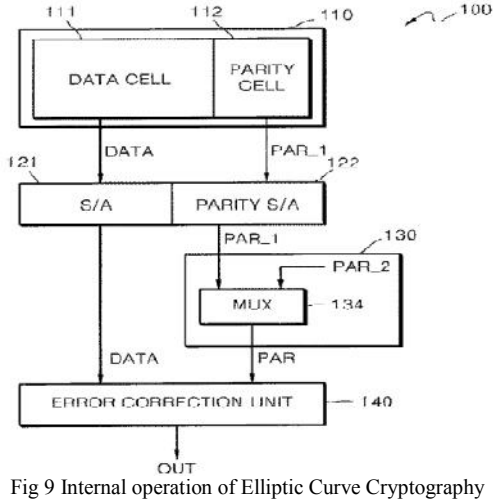

Fig 9 Internal operation of Elliptic Curve Cryptography

The Elliptic Curve Integrated Encryption Scheme (ECIES), also known as Elliptic Curve Augmented Encryption Scheme or simply the Elliptic Curve Encryption Scheme. Figure 9 depicts the internal operations of Elliptic Curve Cryptography in which two parity cells are added and compared and if it is accepted then no error is found if it is not accepted then the error is detected by the Error Correction Unit.

V. CONCLUSION

The current version of ES-MPICH2 is focused on securing the transmission control protocol (TCP) connections on the internet, because we addressed the data confidentiality issues on geographically distributed cluster computing systems. In addition to the MPI library, other parallel programming libraries will be investigated. Candidate libraries include the shared memory access library and the remote direct memory access library. We

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

plan to provide confidentiality services in the SHMEM and RDMA [1] libraries. A third promising direction 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. So far, our study has been restricted to a fairly small platform which consists of 8 nodes. In the future, we plan to use larger clusters to study various aspects of our designs regarding scalability. Another direction we are currently pursuing is to provide support for MPI-2 functionalities such as one-sided communication using RDMA and atomic operations in InfiniBand. We are also working on how to support efficient collective communication on top of InfiniBand [1].

References

[1] J. Liu, W. Jiang, P. Wyckoff, D.K. Panda, D. Ashton, D. Buntinas, W. Gropp, and B. Toonen, "Design and Implementation of Mpich2 over Infiniband with rdma Support," Proc. 18th Int'l Parallel and Distributed Processing Symp., p. 16, Apr. 2004.

[2] A. Geist, W. Gropp, S. Huss-Lederman, A. Lumsdaine, E.L. Lusk, W. Saphir, T. Skjellum, and M. Snir, "Mpi-2: Extending the Message-Passing Interface," Proc. Second Int'l Euro-Par Conf. Parallel Processing (Euro-Par '96), pp. 128-135, 1996.

[3] D.S. Wong, H.H. Fuentes, and A.H. Chan, "The Performance Measurement of Cryptographic Primitives on Palm Devices," Proc. 17th Ann. Computer Security Applications Conf. (ACSAC), pp. 92-101, 2001.

[4] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the Mpi Message Passing Interface Standard," Parallel Computing, vol. 22, no. 6, pp. 789-828, 1996.

[5] Intel Corporation. Intel mpi Benchmarks User Guide and Methodology Description, 2008.

[6] D.E. Denning, "Secure Personal Computing in an Insecure Network," Comm. ACM, vol. 22, no. 8, pp. 476-482, 1979.

[7] J. Daemen and V. Rijmen, The Design of Rijndael. Springer, 2002.

[8] J.J. Dongarra, S.W. Otto, M. Snir, and D. Walker, "An Introduction to the Mpi Standard," technical report, Knoxville, TN, 1995.

[9] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, Software Implementation of Elliptic Curve Cryptography over Binary Fields, 2000.

[10] M. Pourzandi, D. Gordon, W. Yurcik, and G.A. Koenig, "Clustersand Security: Distributed Security for Distributed Systems," Proc. IEEE Int'l Symp. Cluster Computing and the Grid (CCGrid '05), vol. 1, pp. 96- 104, May 2005.

Gangadhara Rao.D received his B.Tech in Malineni Lakshmaiah Engineering College, in 2010. The M.Tech. Degree in CSE from Krishna Chaitanya Institute of Technology & Sciences, in 2013. At present, He is engaged in**"Enhanced Security Mechanisms in MPI as ES-MPICH2".**

Balusa Anil Kumar working as Assistant Professor in CSE Department, Krishnachaitanya Institute Of Technology and Sciences, Markapur. Completed His M.Tech in (Computer networks & information security) at Holymary Institute Of Technology And Sciences, Hyderabad.