Review Article

ACCENTS Transactions on Information Security, Vol 1(1)
ISSN (Online): 2455-7196

An overview of cryptographic algorithms and security challenges in big data

Anurag Sarkar*
Department of Computer Science, St. Xavier's College, Kolkata, India.

Abstract

With an increasing reliance on online communications and transactions, the need for effective cryptographic algorithms to ensure security and reliability has never been more important, especially with the rise of big data. This paper provides a survey of different cryptographic algorithms that have been used in the past and are being used today to provide such security. The different cryptographic algorithms can be divided into three categories–symmetric key algorithms, asymmetric key algorithms and hash functions. The objective of the paper is to provide an overview of these different types of cryptographic algorithms and to serve as a basic introduction to the field. Another goal is to discuss some of the cryptographic challenges that are faced in the domain of big data.

Keywords

Symmetric key cryptography, Asymmetric key cryptography, Cryptographic hash function, Cryptographic algorithm, Big data security.

1.Introduction

Today, with tens of millions of people communicating online and conducting commercial transactions, information and data security is of the utmost importance. Such security is achieved through the implementation of different types of cryptographic algorithms. The word 'Cryptography' may be defined in several ways. It is often defined as the art (or science) of writing in secret code. Cryptography is not a modern concept and has been used for several millennia to communicate secret information. However, the need for cryptography has never been more than in the digital age in which we now live. With the growing reliance on the Internet over the past few decades, several different cryptographic techniques and algorithms have been developed depending for different situations and applications. These cryptographic algorithms usually have to satisfy certain key principles of security in order to be effective and efficient. The following are these principles which should be at the foundation of most cryptographic systems:

A. Authentication

This is used to correctly identify the sender and intended recipient of a message and provides a guarantee to the recipient that the message has indeed come from the correct sender. It prevents any unauthorized entity or attacker from pretending to be the sender or the recipient. Such a breach of security is called fabrication or masquerading.

B. Confidentiality

This specifies that only the sender and the intended recipient of a message should be able to access the contents of the message, i.e. no unauthorized entity or attacker should be able to gain access to the unencrypted plaintext.

C. Integrity

The integrity of the message should be preserved i.e. an unauthorized entity should not be able to modify or tamper with the contents of the message. Only the sender and recipient should have this privilege[1-4].

D. Non-repudiation

This principle prevents either party involved in the transaction to prevent that the transaction occurred. That is, it prevents the sender from denying that he or she sent the message after having already sent it. Similarly, it prevents the receiver from denying that he or she received the message after having already received it.

Thus, the cryptographic algorithms that implement the above principles serve a number of important functions. Through authentication, they ensure that the correct parties are involved in the transaction or message transfer. By ensuring confidentiality, they prevent unauthorized access and by integrity, they make sure that no outside parties can change the contents of the message. Finally, via non-repudiation, the algorithms also ensure that the parties that are involved in the transaction cannot later deny that they participated in it.

*Author for correspondence

All cryptographic algorithms consist of two basic processes–encryption and decryption. In the encryption process, the algorithm takes the plaintext i.e. the text that is readable and converts it through one or more manipulations, into an encrypted text called the ciphertext. The ciphertext is not readable and hence cannot be understood if intercepted by attackers. The ciphertext is then converted back into the original, readable plaintext by the decryption process.

In this paper, the cryptographic algorithms have been divided into three basic categories – symmetric key algorithms, asymmetric key algorithms and hash functions. The following sections will discuss each of these classes of algorithms separately.

2.Symmetric key algorithms

In symmetric key cryptographic algorithms, only one key is used for both encryption and decryption. That is, the same key is used both to encrypt the plaintext into ciphertext at the sender's end and to decrypt the ciphertext into plaintext at the receiver's end. This symmetry of the encryption and decryption processes gives this class of algorithms its name. Additionally, since the key is known only to the sender and receiver and must be kept secret, this form of cryptography is also known as secret-key cryptography or private-key cryptography.

Symmetric key algorithms suffer from two primary drawbacks:

- Symmetric key algorithms require the two parties to share a secret key. However, the problem is that the sender and receiver must somehow agree on the key in the first place. A naive solution to this is to establish a secret channel of communication accessible only to the sender and the receiver and use that to exchange the key. However, if such a channel exists, then it makes sense to exchange the plaintext message itself. A practical solution to this problem was proposed by Whitfield Diffie and Martin Hellman in 1976, named the Diffie-Hellman Key Exchange algorithm. Refer to [6] for details on this algorithm.
- Each pair of communicating parties requires its own secret key. If only 2 parties are communicating with each other, then 1 key is needed. However, if 3 parties are involved, then the number of keys required increases to 3. For n users, the number of keys is given by (n(n-1))/2. For commercial transactions, involving thousands of merchants and customers, the number of keys is large and impractical.

Regardless of the above drawbacks, symmetric key algorithms are still used widely used in practice as these disadvantages have been overcome by various novel techniques and ideas. We now discuss some of the most important symmetric key algorithms.

2.1.Data encryption standard (DES)

The Data Encryption Standard was a tremendously popular symmetric key algorithm, developed by IBM in the early 1970s, based on work by Horst Feistel. This algorithm was then submitted to the National Bureau of Standards (NBS, now known as NIST) in 1972 after the NBS invited proposals for algorithms to protect sensitive and critical government data. Eventually, after consultation with the National Security Agency (NSA), the NBS adopted a modified version of IBM's algorithm in 1976 and published it is an official Federal Information Processing Standard (FIPS) in 1977.

DES is a block cipher, i.e. it takes as input a fixed-length plaintext string and converts it, through a series of complex operations, into a ciphertext string of the same length. The block size in DES is 64 bits. To encrypt the plaintext (or decrypt the ciphertext), DES uses a 64-bit key of which only 56 bits are actually used, the remaining 8 bits only for parity checking purposes. The algorithm itself consists of 16 rounds and makes use of the Feistel function and the XOR operation. For an in-depth discussion on the working of DES, refer to [5].

In modern times, DES is considered insufficient for a majority of applications, due to its relatively small 56-bit key size. Since the inner workings of the DES algorithm are publicly known, its strength is based on its key size. A 56-bit key size means that there are 256 possible keys. In January 1999, the Electronic Frontier Foundation and distributed.net succeeded in breaking a DES key in 22 hours and 15 minutes. Cryptanalysis techniques such as differential and linear cryptanalysis have been shown to theoretically be able to break the full 16 rounds of DES but are difficult to implement in practice.

2.1.1.Double DES

To overcome the limitations of DES, but preserve its working principles, Double DES was developed. This uses 2 keys, K1 and K2. First, the original plaintext is encrypted using the DES algorithm with key K1. This encrypted text is encrypted again using the DES algorithm, this time with key K2. The output of this second operation is the final cipher text. Decryption is the same, only the order of keys used is reversed.

One drawback of Double DES is that it suffers from the Meet-in-the-Middle attack.

2.1.2.Triple DES (3DES)

To overcome the Meet-in-the-Middle attack, 3DES was introduced, which as the name suggests, consists simply of the application of the DES algorithm three times to produce the final cipher text. There are two types of 3DES – one uses three different keys, while the other uses two different keys. When three keys are used, the encryption process simply involves using each key to encrypt the message one after the other. However, when two keys are used, say K1 and K2, first, the plaintext is encrypted with K1. This encrypted plaintext is then decrypted using K2. This decrypted text is finally encrypted using K2 to produce the final cipher text. This mode is also referred to as the Encrypt-Decrypt-Encrypt (EDE) mode.

2.1.3.Advantages of DES

- Since DES was developed in the 1970s and designed to run on the hardware from that era, it is extremely fast on modern hardware.
- DES being an ANSI standard means that it can be implemented by any person without requiring any licensing costs.
- Although DES has been around for several decades, the only known attack on it is brute-force.

2.1.4.Disadvantages of DES

- DES is not optimized for software implementations.
- With increasing improvements in hardware, the likelihood of cracking DES is increasing daily.
- The 56 bit key is not considered to be sufficiently long to provide good security in the face of modern hardware and parallel processing.

2.2.International data encryption algorithm (IDEA)

IDEA was proposed by Xuejia Lai and James Massey in 1991 and was intended to be a replacement for the previously described DES. The popular email security protocol PGP (Pretty Good Privacy) is based on IDEA.

Like DES, IDEA is a block cipher consisting of 64-bit blocks, but uses a 128-bit key. It consists of 8 rounds, each of which makes use of modular addition, multiplication and the XOR operation. The 8 rounds are followed by the Output transformation stage.

2.2.1.Advantages of IDEA

- It performs at twice the speed of DES.
- Since the key-length is 128 bits, brute force attacks require an incredible amount of computing power to be successful[7-9].

2.2.2.Disadvantages of IDEA

- Licensing fee is required for using IDEA to provide security in commercial applications.
- DES is more popular and widely accepted since it has been around for far longer than IDEA.

2.3.RC5

RC5 is a symmetric key block encryption algorithm developed by Ron Rivest in 1994. Unlike the previous algorithms, RC5 utilizes a variable number of rounds (0 to 255), a variable block size (32, 64 or 128 bits) and a variable length key size (0 to 2040 bits). The values for each of these parameters can be set according to the specific application and varying security need for which RC5 is required.

2.3.1.Advantages of RC5

- The design is concise and it does not require a lookup table that takes up a large amount of storage.
- It requires far less memory than the AES algorithm we will look at shortly
- The word size, number of iterations and key length are all variable. RC5 is often denoted as RC5-w/r/b where w denotes the word size in bits, r denotes the number of rounds and b denotes the number of 8-bit bytes in the key.
- It is simple and efficient since it makes use of only primitive operations such as modular addition, XOR and shift.

2.4.Blowfish

Blowfish is a block cipher developed by Bruce Schneier and was first published in 1993. Like in IDEA and DES, the block size of Blowfish is 64 bits and the key length varies from 32 to 448 bits. Also similar to DES, Blowfish makes use of 16 rounds of Feistel ciphers and employs large key-dependent S-boxes[10-13].

Blowfish has since been succeeded by the more modern ciphers Twofish and Threefish.

2.4.1Advantages of blowfish

- Fast–The encryption rate on 32-bit microprocessors is 26 clock cycles per byte
- Compact–The algorithm requires less than 5kb memory to execute
- Simple–Like RC5, Blowfish uses primitive operations such as XOR, addition and table lookup, thus allowing for a simple and efficient implementation

- Secure–The variable key length (up to 448 bits) makes the algorithm secure as well as flexible

2.4.2Disadvantages of blowfish

- It is fast except when the keys need to be changed with each new key requiring preprocessing equivalent to the encryption of approximately 4kb of text. This is quite slow compared to other block ciphers.

2.5.Advanced encryption standard (AES)

The AES cryptographic algorithm was established by the US National Institute of Standards and Technology (NIST) in 2001 and is based on the Rijndael cipher developed by Vincent Rijmen and John Daemen.

AES was developed to overcome the limitations of DES that we discussed previously, namely its 56-bit key length and 64-bit block size, both of which were considered too weak to withstand modern exhaustive key searches utilizing a great amount of computational power.

AES is a combination of both permutation and substitution and performs fast in both hardware and software. It uses a fixed block size of 128 bits and key sizes of 128, 192 or 256 bits. Unlike DES, it does not make use of Feistel ciphers. The number of rounds that take place depend on the length of the key. For 128-bit keys, there are 10 rounds, for 192-bit keys there are 12, and for 256-bit keys there are 14 rounds.

Although several theoretical attacks have been proposed and cryptographic breaks have been achieved [10], as of now, there are no known practical attacks on AES.

2.5.1Advantages of AES

- It supports larger key lengths than any variation of DES.
- It is faster than DES in both hardware and software.
- Block size of 128 bits means that it is less susceptible to attacks than 3DES which uses 64-bit blocks[14].

2.5.2Disadvantages of AES

- With recent advances in hardware and computing power, AES has been shown to be vulnerable to certain types of attacks particularly those related to boomerang and rectangle attack with related key-differentials [15]. Details of the inner workings of all these algorithms can be found in [5].

3. Asymmetric key algorithms

According to [4], "the conceptual differences between the two systems are based on how these systems keep a secret. In symmetric key cryptography, the secret must be shared between two persons. In asymmetric key cryptography, the secret is personal (unshared); each person creates and keeps his or her own secret."

Unlike in symmetric key cryptography, in which only one key is used for encryption and decryption, in asymmetric key cryptographic algorithms, we use two keys, one for encryption and the other for decryption. Each person has two keys – a public key and a private key. Whenever user A wants to send an encrypted message to user B, A must encrypt the message using B's public key, which is not kept secret. However, only B's private key, which is known only to B, can be used to decrypt the message.

The introduction of asymmetric key cryptography does not mean however that symmetric key cryptography is obsolete. Asymmetric cryptographic algorithms are slower than symmetric algorithms because they make use of complex mathematical functions unlike symmetric cryptographic algorithms which use simple operations like addition, XOR, etc. Hence, for encrypting large messages, symmetric algorithms are still preferred. Asymmetric cryptographic algorithms find applications in digital signatures, authentication and key exchanges, and thus, a combination of both symmetric and asymmetric algorithms are needed to build a secure cryptosystem.

The key idea in asymmetric key cryptography is the concept of a trapdoor one-way function. A one-way function is one which is easy to compute but whose inverse is computationally infeasible. A trapdoor one-way function is one in which a trapdoor, i.e. a known secret value, allows the inverse to be computed easily as well.

We now discuss some important asymmetric key cryptosystems.

3.1.RSA cryptosystem

This is the most popular asymmetric key algorithm and is named after its inventors Ron Rivest, Adi Shamir and Len Adleman. It was first publicly published in 1977. It is based on the computational infeasibility of factoring the product of two large prime numbers and uses modular exponentiation for encryption and decryption.

RSA uses a public exponent e and a private exponent d. If P is the plaintext and C is the ciphertext, then at the encryption end, C is calculated as C = (P^e) mod n. Similarly, at the decryption end, P is calculated from C as P = (C^d) mod n. The modulus n is a very large number that is the product of 2 large prime numbers. Thus, in RSA, the one-way function is modular exponentiation and the trapdoor is the exponent d, known only to the person authorized to decrypt the ciphertext. In order to break RSA, the attacker must be able to calculate the eth root modulo n of C. Since there are no known polynomial time algorithms to achieve this, there are no known attacks on RSA.

The value of n, as stated before, is generated as the product of two large prime numbers, p and q. It is imperative that these numbers are large, failing which RSA is prone to brute-force attacks. The recommended size for p and q is 512 bits each, thus making the size of n approximately 1024 bits. The exponents e and d are selected using a key generation process that involves n, p and q. First, p and q are generated and n is calculated as their product. Then we calculate the totient of n, t(n), which is equal to the product of (p-1) and (q-1). The exponent e is then selected from the range [1, t(n)] and must be co-prime to t(n). d is then calculated as the inverse of e modulo t(n). The tuple (e, n) then forms the public key and d is the private key.

As mentioned previously, there are no known attacks that can fully break RSA, however some attacks have been developed that target weak plaintext and weak choices of parameters. These are listed below:

- Chosen Cipher text
- Factorization–if modulus is not sufficiently large
- Plaintext–this includes short message attack, cycling attack and unconcealed message attack
- Common Modulus
- Encryption exponent–if the exponent e is too low; attacks include Coppersmith theorem attack, broadcast attack, related message attack and short pad attack
- Decryption exponent–includes revealed decryption exponent attack and low decryption exponent attack
- Implementation –attacks on the implementation of RSA include timing attacks and power attacks

3.1.1Advantages of RSA

- It is very secure since it is infeasible to factor n in order to find out p and q provided that n is sufficiently large.
- It supports digital signatures
- It has wide industry support

3.1.2Disadvantages of RSA

- It is slower and more computationally intensive than the symmetric key algorithms we discussed previously
- The two-part key may be susceptible to GCD attack if the algorithm is improperly implemented.

3.2.Rabin cryptosystem

The Rabin cryptosystem is another asymmetric key algorithm which is similar to RSA and was developed by Michael Rabin in 1979. Unlike RSA, which is based on exponentiation congruence, the Rabin cryptosystem is based on quadratic congruence. In this system, the values of e and d are fixed; e = 2 and d = ½. Thus, the encryption process reduces to calculating C = P2 mod n, and the decryption process reduces to calculating P= (C^½) mod n. The public key is n and the tuple (p, q) is the private key. Like in RSA, n is the product of two large prime numbers p and q.

3.2.1 Advantages of rabin cryptosystem

- The problem of breaking the Rabin cryptosystem has been shown to be as difficult as integer factorization.
- The square of modulo n must be calculated for encryption which makes it more efficient than RSA in which a cube must be computed.

3.2.2 Disadvantages of cabin cryptosystem

- A major drawback of the Rabin system is that decryption produces four equally probable plaintexts and thus the correct plaintext must be guessed from among these four. This is not a major issue when the plaintext is alphabetic but numeric plaintexts can be very problematic in this regard and necessitate the use of advanced disambiguation methods.

3.3.El gamal cryptosystem

Like RSA and Rabin, the ElGamal cryptosystem is an asymmetric key cryptosystem, developed by Taher ElGamal in 1985. It is based on the Diffie-Hellman key exchange algorithm and finds use in recent versions of PGP, the GNU Privacy Guard software and several other cryptosystems. The core concept in the ElGamal system is the discrete logarithm problem. That is, if p is a very large prime number, e1 is a primitive root in the group G = <Zp*, x> and r is an integer, then e2 = (e1^r) mod p can be computed easily but given the values e1, e2 and p, computing r = loge1e2 mod p is computationally infeasible. The ElGamal cryptosystem involves 3 stages–key generation, encryption and decryption.

1) Key generation
The sender selects a large prime number p and a member d of the group G = <Zp*, x> such that 1 <= d <= p-2. The primitive root of this group is e1. Then, e2 is calculated as e2 = e1^d mod p. The public key is formed as the tuple (e1, e2, p) and the private key is d.

2) Encryption
A random integer r is selected in the group G given above. C1 is calculated as e1^r mod p and C2 is calculated as (P x e2^r) mod p where P is the plaintext. C1 and C2 are the two ciphertexts.

3) Decryption
To retrieve the plaintext, P is calculated as [C2 (C1^d)-1] mod p.

3.3.1.Advantages of El gamal cryptosystem
- It is suitable for use in hybrid cryptosystems where the original plaintext message is encrypted using a symmetric algorithm and then ElGamal is employed to encrypt the key

3.3.2.Disadvantages of El gamal cryptosystem
- It is susceptible to known plaintext attack if the same value of r is reused during encryption
- In general, ElGamal encryption results in a message expansion of 2:1 from plaintext to ciphertext

Refer to [4] for a more detailed discussion on the working principles of these asymmetric cryptosystems.

4. Cryptographic hash functions
Cryptographic hash functions are cryptographic algorithms that differ from the above two classes of algorithms in that they do not use a key for encryption/decryption. Instead, they compute a fixed-length value called the hash which is based on the properties of the plaintext in a way that makes it impossible for the length or contents of the original plaintext to be recovered from the hash value. A common use of hash functions is to encrypt passwords. Common hash functions are given below:

4.1.Message digest (MD)
MD algorithms include a number of byte-oriented algorithms that generate 128-bit hash values from messages of any arbitrary length. The most popular version is MD5 which was developed by Ron Rivest.

4.2.Secure hash algorithm (SHA)
This was developed as the NIST's Secure Hash Standard (SHS). The SHA-1 variant generates a 160-bit hash value. SHA-2 consists of 5 different SHA algorithms, the original SHA-1, along with SHA-224, SHA-256, SHA-384 and SHA-512, which produce hash values of length 224 bits, 256 bits, 384 bits and 512 bits respectively.

4.3.Race integrity primitives evaluation message digest (RIPEMD)
This is a family of hash functions develop by Hans Dobbertin, Antoon Bosselaers and Bart Preneel in Leuven, Belgium, published in 1996. The design of RIPEMD is similar to that of MD4 and has a similar performance to SHA-1. The improved RIPEMD-160 was designed to perform optimally on 32-bit processors and was intended to replace 128-bit hash functions. Other variants are RIPEMD-128, RIPEMD-256 and RIPEMD-320.

4.4.Hash of variable length (HAVAL)
Invented by Yuliang Zheng, Josef Pieprzyk and Jennifer Seberry in 1992, HAVAL is a hash function capable of producing hashes of variable lengths (128 bits, 160 bits, 192 bits, 224 bits, 256 bits) and also allows the user to determine the number of rounds that are to be used to produce the hash value.

4.5.Tiger
This was designed by Ross Anderson and Eli Biham in 1995 to run efficiently on 64-bit platforms. It produces a hash of size 192 bits. Shorter versions have been developed that generate hashes of size 128 and 160 bits to provide compatibility with other commonly used hash functions.

4.6.Whirlpool
This cryptographic hash function was designed by Vincent Rijmen and Paulo S. L. M. Barreto in 2000 and is based on a heavily modified version of AES. It takes as input messages of size less than 2^256 bits and produces a message digest of 512 bits. Since its structure is vastly different than that of SHA-1 and MD5, it is not prone to the same types of attacks.

5. Cryptographic challenges in big data
Big Data is a field that has gained tremendous popularity and importance in the recent past. With the increasing amount of data and information due to the rapid growth of social networks and the internet, it is crucial that secure ways of working with big data are researched and developed. In this section, we take a look at some of the key challenges in implementing security in the domain of big data.

5.1.Infrastructure security

5.1.1Distributed secure computation

In order to process massive amount of data, as is the case with big data applications, parallel computations and storage is required. The MapReduce framework offers these capabilities. An important challenge is to secure the individual processors from attack and to protect the data from any compromised processors.

5.1.2Security for non-relational data

Most non-relational data stores do not provide explicit support for implementing security to protect against various attacks. Thus, developers using NoSQL databases usually have to embed security measures in the middleware. This becomes even more problematic when dealing with clusters of databases as is the case in big data.

5.2Data privacy

5.2.1Data mining and analytics

Data mining algorithms working on large collections of data about people may inadvertently disclose private and confidential information. Thus, appropriate measures need to be taken to ensure the privacy of individuals and avoid legal infringement.

5.2.2Granular access control

Controlling access to data is a major challenge in a big data setting with a large amount of data and a large amount of users, each with its set of unique access privileges. Ensuring users can access the data they are eligible to and can't access data they are not authorized to must be accomplished in a cost-effective manner.

5.2.3 Data centric Security

Although data can be protected by securing the system in which it is contained, this may be problematic in a big data scenario where networks consisting of several parallel systems and nodes may offer many ways for attackers to breach the system security. Thus, it is more desirable to use various cryptographic measures to secure the data contained within the systems rather than simply securing the system and leaving the data inside vulnerable[16].

5.3Integrity and reactive security

5.3.1.Real time monitoring

Monitoring security in real-time is challenging in a big data environment as security alerts are generated regularly by some device or the other amongst the many hundreds of devices that may be in action. Many of these alerts turn out to be false positives and thus the challenge is to determine which alerts need to be addressed immediately, which can be addressed later and which can be ignored.

5.3.2 End-point validation and filtering

As big data involves collecting input data from a large amount and variety of sources, a key challenge is to ensure proper validation of all of this diverse data. Even more important is filtering out any input data that has malicious intent such as viruses, worms, injection attacks, etc.

5.4Data management

5.4.1Secure storage & transaction logs

Due to the massive size of data sets used in big data applications, auto-tiered systems are used for data management. However, unlike manual tiered systems in which data movement could be monitored by administrators, auto-tiered systems do not effectively keep track of data moving between tiers. This of course is a threat to securing the data and protecting it from being stolen. Thus, effective measures are needed to prevent any unauthorized access.

5.4.2Data provenance

Keeping records about the origins and ownership of the data that is accumulated for big data applications is a difficult task. Methods are needed to effectively manage such provenance metadata and to analyze it in order to find useful and meaningful dependencies between different data sources in a manner that is not too computationally intensive[17].

5.4.3Granular audits

While real time security monitoring (5.3.1) is certainly a goal of big data security, it is still necessary to perform audits to ensure no significant attacks or alerts went unattended. Understanding when, why and how an attack took place is essential in providing security in the future.

6. Conclusion

This paper provided a brief overview of symmetric key and asymmetric key cryptographic algorithms as well as cryptographic hash functions and discussed some popular algorithms in each category. We also looked at some of the cryptographic challenges offered by the field of big data. With increasing computational power and resources, there has been a need to constantly improve upon each of the cryptographic algorithms in order to provide security against attacks and intruders. Thus, in the future, it would be useful to conduct a study comparing the performance and efficiency of these algorithms, particularly in a modern setting, and research ways in which these algorithms may need to be enhanced so that they remain immune to attacks and may be effectively applied in big data applications.

Acknowledgment
None.

Conflicts of interest
The author has no conflicts of interest to declare.

References
[1] Kessler GC. An overview of cryptography. https://www.garykessler.net/library/crypto.html. Accessed 26 October 2015.
[2] Wheeler DA. Secure Programming HOWTO. https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf Accessed 26 October 2015.
[3] Noubir, G. Fundamentals of cryptography:algorithms, and security services. http://www.ccs.neu.edu/home/noubir/Courses/CSU610/S06/cryptography.pdf. Accessed 26 October 2015.
[4] Forouzan BA, Mukhopadhyay D. Cryptography and network security (Sie). McGraw-Hill Education; 2011.
[5] Kahate A. Cryptography and network security. Tata McGraw-Hill Education; 2013.
[6] Palmgren K. Diffie-hellman key exchange: a non-mathematician's explanation. ISSA J. 2006.
[7] Standard DE. Data encryption standard. Federal information processing standards publication. 1999.
[8] https://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm. Accessed 26 October 2015.
[9] https://en.wikipedia.org/w/index.php?title=Blowfish_(cipher)&oldid=680738067. Accessed 26 October 2015.
[10] Miller FP, Vandome AF, McBrewster J. Advanced encryption standard.2009.
[11] https://en.wikipedia.org/w/index.php?title=RSA_(cryptosystem)&oldid=688848088 Accesed 26 October 2015.
[12] https://en.wikipedia.org/w/index.php?title=Rabin_cryptosystem&oldid=685082841. Accessed 10 October 2015.
[13] https://en.wikipedia.org/w/index.php?title=ElGamal_encryption&oldid=675442904. Accessed 10 October 2015.
[14] Gawali DH, Wadhai VM. Rc5 algorithm: potential cipher solution for security in wireless body sensor networks (WBSN). International Journal of Advanced Smart Sensor Network Systems. 2012; 2(3):1-7.
[15] Jain R, Jejurkar R, Chopade S, Vaidya S, Sanap M. AES algorithm using 512 bit key implementation for secure communication. International Journal of Innovative Research in Computer and Communication Engineering. 2014; 2(3):3516-22.
[16] Yadav PS, Sharma P, Yadav KP. Implementation of RSA algorithm using Elliptic curve algorithm for security and performance enhancement. International Journal of Scientific & Technology Research. 2012; 1(4):102-5.
[17] https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf. Accesed 26 October 2015.

Anurag Sarkar is currently a post-graduate student at the Department of Computer Sciene at St. Xavier's College, Kolkata. His research interests include machine learning and data mining.