Indian Journal of Cryptography and Network Security (IJCNS)
ISSN: 2582-9238 (Online), Volume-4 Issue-1, May 2024

Security of the Secp256k1 Elliptic Curve used in the Bitcoin Blockchain

Kannan Balasubramanian

Abstract: ***The article delves into the intricate characteristics and security properties of the secp256k1 elliptic curve used for the generation of addresses in the Bitcoin blockchain. The Bitcoin blockchain is a decentralized digital ledger that records all transactions made with Bitcoin cryptocurrency. In this work, the secp256k1 elliptic curve and its parameters and the method of generating private and public keys using random numbers are described. While the private key allows for the signing of transactions to spend Bitcoin, the corresponding public key and address enable others to verify transactions and send funds to that specific address on the blockchain, ensuring security, authenticity, and privacy in the decentralized network. The attacks on the use of secp256k1 for generating the bitcoin addresses like the Brute force attack, twist attack, fault attacks, and side channel attacks in the implementation of the elliptic curve are discussed. By maintaining the security and integrity of secp256k1, we can ensure that cryptographic operations, such as digital signatures and key exchanges, remain uncompromised. If the curve's security were compromised, malicious users could potentially derive private keys from public keys, leading to unauthorized transactions, double-spending, or other malicious activities. The security of implementation can be enhanced by ensuring cryptographic libraries and software implementations that utilize secp256k1 undergo thorough testing and validation to ensure correct and secure operations. The important attacks on blockchain technology like the 51% attack, Sybil attack, Double-Spending attack, and Smart Contract vulnerabilities are discussed. Through a comprehensive exploration, readers will gain insights into why this particular elliptic curve was chosen for use in Bitcoin's cryptographic protocols, highlighting its role in ensuring the robustness and integrity of the blockchain ecosystem.***

Keywords: Elliptic Curves, Brute Force Attack, Twist Attack, Side-Channel Attacks, Random Number Generators, Sybil Attack, Double-Spending Attack

I. INTRODUCTION

The Bitcoin blockchain is a decentralized digital ledger that records all transactions made with Bitcoin cryptocurrency[1]. Each transaction is verified by a network of computers (nodes) through a process called mining, where transactions are bundled into blocks and added to the chain in chronological order.

Manuscript received on 01 December 2023 | Revised Manuscript received on 09 December 2023 | Manuscript Accepted on 15 May 2024 | Manuscript published on 30 May 2024.

***** *Correspondence Author(s)*
Dr. Kannan Balasubramanian*, Professor, School of Computing, SASTRA University, Thanjavur. E-mail: kannanb@cse.sastra.edu, ORCID ID: 0000-0003-1134-0345

This immutable and transparent system ensures security, and consensus, and prevents double-spending without the need for a central authority**.** The security of the Bitcoin blockchain is fundamentally rooted in Elliptic Curve Cryptography (ECC). Specifically, Bitcoin uses ECC to generate public and private key pairs that facilitate secure transactions. When a user wants to send Bitcoin, their private key signs the transaction, and the recipient uses the sender's public key to verify its authenticity. This cryptographic mechanism ensures that only the owner of the private key can authorize transactions, maintaining the integrity and security of the decentralized ledger system. The hashing function employed in the Bitcoin blockchain is SHA-256 (Secure Hash Algorithm 256-bit). This cryptographic hash function takes an input of any size and produces a fixed-size 256-bit output, ensuring that even a tiny change in the input produces a significantly different output. Each block in the Bitcoin blockchain contains a unique cryptographic hash, which includes the hash of the previous block. This chaining mechanism ensures the immutability of the blockchain; altering any transaction in a block would necessitate recalculating all subsequent blocks' hashes, making the system tamper-evident and enhancing its security. The generic structure of the blockchain and the generic structure of a block are shown in Figure 1 and Figure 2.

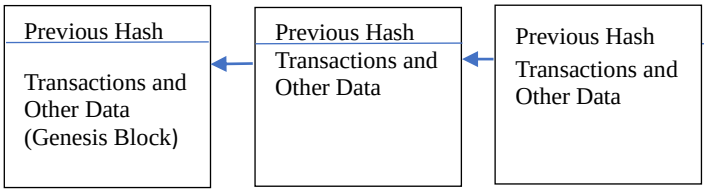

Figure 1: The Generic Structure of a Blockchain

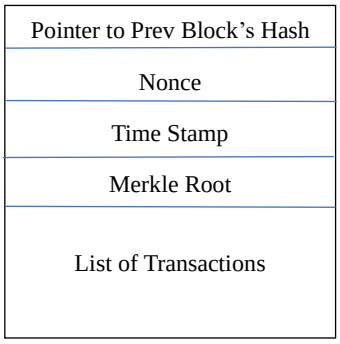

Figure 2: The Generic Structure of a Block

The secp256k1 elliptic curve is a specific elliptic curve used in Bitcoin for cryptographic functions, particularly for generating public and private key pairs.

Bitcoin addresses are derived from public keys, and the security of Bitcoin relies on the computational difficulty of solving certain mathematical problems related to this elliptic curve. It is believed to be secure against certain types of attacks, including those based on the discrete logarithm problem. The secp256k1 curve was chosen, in part, for its efficiency in terms of both computation and storage. The key characteristics of the secp256k1 elliptic curve are described in the next section.

A. Description of the Elliptic Curve

The equation for secp256k1 is $y^2 = x^3 + 7$. The parameters for this are discussed in [2]. The curve operates over a finite field of prime order. The field size is a prime number, specifically 2^ {256} - 2^ {32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 which equals the following number.

115792089237316195423570985008687907853269984665640564039457584007908834671663

The base point or generator point, often denoted as G, is a specific point on the curve. The coordinates of G are pre-defined and are part of the standard: The generator point G=(x,y) is given below:

(55066263022277343669578718895168534326250603453777594175500187360389116729240,
32670510020758816978083085130507043184471273380659243275938904335757337482424)

The order of the generator point (G) is a prime number and denotes the number of points on the curve.

n = 115792089237316195423570985008687907852837564279074904382605163141518161494337

B. Method for Generating Private and Public Keys

Generating private and public keys in the Bitcoin protocol typically involves using elliptic curve cryptography (specifically, the secp256k1 curve) to produce these keys.

Step 1: Choose a Random Private Key:

Start by generating a random 256-bit number. This number serves as your private key. This private key is a crucial piece of information that you must keep secret. Anyone who has access to this key can control the associated Bitcoins.

Step 2: Generate the Public Key:

Use elliptic curve multiplication to derive a public key from the private key. The secp256k1 curve equation $y^2 = x^3 + 7$ over the finite field is used for this purpose. Multiply the base point of the curve (known as the generator point) by the private key to get the corresponding public key.

Step 3: Generate the Bitcoin Address:

The Bitcoin address is derived from the public key but goes through a hashing and encoding process. The public key is first hashed using the SHA-256 algorithm and then RIPEMD-160. This results in a 160-bit hash. This hash is then encoded into a format called Base58Check, which produces the familiar Bitcoin address format you might recognize (starts with a '1' or '3' for mainnet addresses).

The Pseudocode for generating the private and public keys is given in Figure 3. A Python program that implements private and public key algorithms is given in Appendix A. The Python program in Appendix B generates the bitcoin addresses from the private key and public key using the python-bitcoin-lib library.

```
1. Generate Private Key ():
   private key = Random256Bits ()
2. Generate Public Key (private_key):
   public_key = secp256k1_base_point * private_key
   return public_key
3. Generate Bitcoin Address (public_key):
   hash_1 = SHA-256(public_key)
   hash_2 = RIPEMD-160(hash_1)
   checksum = SHA-256(SHA-256(hash_2)) [:4]
   address = Base58Check Encode (hash_2 + checksum)
   return address
```

Figure 3: Method for Generating Private and Public Keys in Bitcoin Blockchain

II. SECURITY ATTACKS ON THE SECP256K1 ELLIPTIC CURVE

The secp256k1 elliptic curve is widely used in various cryptographic applications, most notably as the basis for Bitcoin's public key infrastructure. As such, it has been scrutinized extensively by researchers and cryptanalysts. The various attacks on the use secp256k1 curve include Brute force attacks, Quantum attacks, side-channel attacks, fault attacks, and flaws in the implementation. The brute force attack involves solving the elliptic curve discrete logarithm problem (ECDLP). The security of elliptic curve cryptography (ECC) relies heavily on the computational difficulty of solving the ECDLP problem. Given current computational resources and techniques, brute-forcing a secp256k1 private key is computationally infeasible. The Quantum attack on the ECC-based Cryptographic systems by using algorithms like Shor's algorithm to solve the ECDLP in polynomial time. However, as of now, there's no known practical quantum algorithm that can break secp256k1. Nevertheless, the emergence of quantum computing remains a potential long-term threat to ECC systems.

The side-channel attacks do not target the mathematical properties of the curve but exploit weaknesses in the implementation. Side-channel attacks include attacks that use timing information, power consumption, or electromagnetic leaks to gain information about secret keys. The Fault attacks involve intentionally introducing faults (errors) into cryptographic computations and observing the results to extract secret information. For secp256k1, researchers have studied fault attacks, and while they might introduce vulnerabilities in specific implementations, they don't break the fundamental security of the curve itself. Flaws may also exist in the software and hardware implementation of the Cryptographic systems. Poor implementations, random number generation issues, or other software bugs can inadvertently weaken security. However, as with any cryptographic system, continuous monitoring, research, and adherence to best practices are crucial to maintaining the security of the use of sec256k1 elliptic Curve.

A. Subgroup Attack and Twist Attack

There is a subgroup attack on Elliptic Curves using which an attacker can obtain the private key thus breaking the cryptosystem. Assume an elliptic curve has a subgroup H with a prime number of elements. This count is called the group's order. Alice wants insights into Bob's private key, b. She selects a point, P, from subgroup H and tells Bob it's her public key, asking for an encrypted message. Bob calculates Q = b*P as a shared secret, encrypting a message C for Alice. Knowing Q belongs to the H subgroup, Alice attempts to decrypt C using points like P, 2P, 3P, etc., until success at kP. Now, with Q = kP and Q = b*P, Alice deduces k = b mod q, with q being H's order.

By using a specific curve point, Alice gains knowledge about Bob's private key. However, this attack doesn't apply to the elliptic curve secp256k1 due to its prime group element count, meaning no non-trivial subgroups exist according to Lagrange's Theorem. To modify the small subgroup attack, Alice selects a point from a curve variant, E2, with a different constant term than secp256k1's y^2 = x^3 + 7, like y^2 = x^3 + 2. This E2 curve possesses small subgroups. If Bob's elliptic curve multiplication ignores P's actual curve, his software computes on E2. Thus, Alice can execute the aforementioned small subgroup attack. This is called the Twist attack on the Elliptic Curve [3].

B. Weak Nonce Attack on ECDSA Signature

Alice signs a message using the Elliptic Curve Digital Signature Algorithm (ECDSA). Alice starts with her private key, generating her public key through elliptic curve cryptography with the equation $y^2 = x^3+7$ Mod N. Using the generator point G, Alice derives her private key as G added to itself a random number of times. Her public key is derived from M times G. Alice signs a message with ECDSA using R and S values. Bob verifies the signed message using R, S, and Alice's public key. A particular type of attack called the Lenstra-Lenstra-Lovasz (LLL) method can be used on the signature to get the private key [4].

C. Other Attacks on the Elliptic Curve

Several other attacks including fault attacks on the use of the elliptic curve secp256k1 have been discussed [5][17][18][6][7]. Comparison of the secp256k1 with other Curves like the Edwards curve has also been discussed [8]. To avoid backdoor attacks on the use of Elliptic Curves, the use of multiple Elliptic Curves has been suggested [9][20][21].

III. VULNERABILITIES IN THE IMPLEMENTATION OF THE ELLIPTIC CURVE

While secp256k1 is considered to be secure when used correctly, there are potential vulnerabilities or risks associated with its usage:

1. Implementation Flaws: One of the primary concerns with cryptographic algorithms is not necessarily the algorithm itself but how it's implemented. Poorly coded software libraries or hardware can introduce vulnerabilities, such as side-channel attacks where an attacker can gain information about the private key by monitoring physical aspects like power consumption or timing.
2. Reused Addresses: While not a vulnerability in secp256k1 itself, reusing Bitcoin addresses can lead to privacy and security concerns. If an attacker manages to compromise a single private key associated with a reused address, they could potentially access all funds sent to that address. It's essential to generate a new address for each transaction to minimize such risks.
3. Incorrect Key Management: Human errors can be a significant source of vulnerabilities. If individuals fail to securely manage their private keys, such as storing them on insecure devices, sharing them, or losing them, it can lead to unauthorized access and theft of Bitcoin funds.
4. Weak Random Number Generation: Generating private keys requires a robust source of randomness. If the random number generator used to create private keys is flawed or predictable, it could lead to the generation of weak or easily guessable keys, making them susceptible to brute-force attacks.

To mitigate these risks, developers, users, and organizations involved with Bitcoin and other blockchain technologies should follow best practices for cryptographic key management, regularly update their software and hardware, use well-reviewed and trusted libraries, and remain informed about potential advancements or threats in the field of cryptography[10][11][12][19][13][14].

IV. RANDOM NUMBER GENERATORS FOR THE GENERATION OF PRIVATE KEYS

A random number generator is used to generate the private key in the Bitcoin Blockchain. Various methods for generating the random numbers are available including Cryptographically Secure Pseudorandom Number Generators (CSPRNGs), Operating System Provided Randomness, Hardware Random Number Generators, and Entropy Accumulation. Most systems rely on CSPRNGs to generate random numbers that are suitable for cryptographic operations. These generators produce sequences of numbers that appear random and unpredictable, making them suitable for generating private keys. In the context of Bitcoin and secp256k1, CSPRNGs are used to produce 256-bit numbers that serve as private keys. These numbers must be truly random to ensure the security of the associated Bitcoin addresses. Many Bitcoin wallet applications leverage the operating system's built-in mechanisms for randomness. For instance, operating systems like Linux provide /dev/random or /dev/random interfaces that offer random data. These sources are often considered to be sufficiently random for cryptographic purposes. Bitcoin software can tap into this randomness to generate private keys.

For even higher security, some systems might use hardware-based random number generators. These devices generate random numbers based on physical processes, such as electronic noise or radioactive decay. Using HRNGs can reduce the reliance on software-based sources of randomness and provide an additional layer of security.

To ensure randomness, systems might accumulate entropy from various sources. This could include user interactions, system events, and other unpredictable factors. By combining entropy from multiple sources, systems can produce more robust random numbers suitable for private key generation.

A. Attacks on the Bitcoin Blockchain Technology

Over the years, the Bitcoin blockchain has faced various attacks and vulnerabilities [15]. Some of the major attacks and issues that have been observed are the 51% attack, the Sybil attack, Transaction Malleability, Denial-of-Service (DoS) attacks, fork attacks, Smart Contract vulnerabilities, and Mining Centralization attacks.

The 51% attack is one of the most discussed potential threats to the Bitcoin network. If a single entity or a group of entities controls more than 50% of the network's mining power, they can potentially alter transaction histories, double-spend coins, or even halt transactions. However, executing a 51% attack on Bitcoin has become increasingly difficult due to its large and distributed mining network. In a Sybil attack, an attacker tries to control a significant portion of the network nodes to influence the network's behavior. This can be used to spread false information or disrupt the network. Bitcoin's proof-of-work consensus mechanism makes it challenging for a single entity to control a majority of nodes, reducing the risk of such attacks.

Transaction Malleability: is a vulnerability that allows an attacker to change the transaction ID before it is confirmed on the blockchain. While it doesn't allow theft of funds directly, it can confuse transaction monitoring systems and lead to potential issues for services relying on unconfirmed transaction IDs. The Denial-of-Service (DoS) attacks can target the Bitcoin network, aiming to flood nodes with unnecessary or malicious data, thereby disrupting the network's functionality. These attacks can slow down transaction processing times and cause inconvenience to users.

A fork in the blockchain can occur due to software bugs or intentional actions by miners. While not necessarily malicious, unintentional forks can create confusion and disrupt consensus. Deliberate forks can be more problematic, as they can lead to a split in the community and the creation of new competing chains (like Bitcoin Cash or Bitcoin SV). While Bitcoin's scripting language is limited compared to other blockchains like Ethereum [16], there have been instances where vulnerabilities in smart contracts or custom scripts have been exploited, leading to loss of funds or disruptions. Although Mining Centralization is not a direct attack, the increasing centralization of mining power in specific regions or by specific entities can pose a risk to the network's decentralization and security. Centralized mining pools have a significant influence over transaction processing and network upgrades. Despite these potential vulnerabilities and attacks, Bitcoin has proven resilient over the years, largely due to its robust network of nodes, miners, developers, and community participants. Continuous research, development, and vigilance are essential to mitigate risks and maintain the network's integrity.

V. CONCLUSIONS

This research focussed on the security of the usage of the Elliptic Curve used in the Bitcoin Blockchain. With the prevention of brute force attacks, twist attacks, fault attacks, and side-channel attacks on the implementation of the secp256k1 elliptic curve, the robustness and integrity of the Bitcoin Blockchain can be improved. This research article discussed the vulnerabilities that may arise in the software implementation of the Bitcoin Elliptic Curve and methods to improve the security of the Bitcoin Blockchain by preventing these vulnerabilities from affecting Bitcoin transactions.

Appendix A

A Python Program to generate private and public keys using the secp256k1 curve in the Bitcoin blockchain.

```
pip install ecdsa
import ecdsa
from ecdsa import SigningKey, SECP256k1
# Curve parameters
p = int ("FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF", 16)
a = -3
b = int ("5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B", 16)
# Generator point (base point)
G_x = int ("79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798", 16)
G_y = int("483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8", 16)
G = ecdsa.ellipticcurve.Point(SECP256k1.curve, G_x, G_y, 1)
# Order of the subgroup generated by G
n = int ("FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551", 16)
# Create a key pair
private_key = SigningKey.generate(curve=SECP256k1)
public_key = private_key.get_verifying_key()
```

Output:

Private Key: cfe55b43b13aa4dfeb620e40e49a7d33ff35a162b948093b87a3ae8e9c48cf11

Public Key: 8d0219bb17ad3a1d6f2dda7a3afce035c35aa2aae5ed85f53f0913aaea0a9a7d6f33d9f4cfa0eab9036d47b767eb2fd4a23809e8021f9a976a882bba5b248346

Signature: 30ce478f29ccccb0669cb1b0957fff719fb5bc8582a859aa62f39c23cb2115fccabf19f7bc8088ed37f334467bad668ffb015f8b1b4988d952b66653c575f13d

Verification Result: True

Appendix B

A Python program to generate Bitcoin addresses using public and private keys.

```
pip install python-bitcoinlib
import os
from bitcoin.wallet import CBitcoinSecret, P2PKHBitcoinAddress
# Generate a random private key (you can also use your private key)
private_key = CBitcoinSecret.from_secret_bytes(os.urandom(32))
# Derive the public key from the private key
public_key = private_key.pub
# Create a Bitcoin address from the public key
address = P2PKHBitcoinAddress.from_pubkey(public_key)
# Print the generated address and private key
print("Bitcoin Address:", address)
print("Private Key:", private_key)
```

output:

Bitcoin Address: 1BSZ7Jns6yfMKub6Vi3SBnAYvVz9yV2VkH
Private Key: L4jshZFxYiEPg6TgxrJ6ZkZeXZbZTC7YXA25ggmomkhHPvnAiqZF

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of my knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	I am only the sole author of the article

REFERENCES

1. Bitcoin, https://www.bitcoin.org
2. SEC2 Recommended Elliptic Curve Domain Parameters, https://secg.org/sec2-v2.pdf
3. Dangers of using secp256k1 for encryption-Twist Attacks, https://github.com/christianlundkvist/blog/blob/master/2020_05_26_secp256k1_twist_attacks/secp256k1_twist_attacks.md
4. M.M.Ulla, D.S.Sakkari, Research on Elliptic Curve Crypto System with Bitcoin Curves – SECP256k1, NIST256p, NIST521p and LLL, Journal of Cyber Security and Mobility, Vol. 12 1, 103–128. M.doi: 10.13052/jcsm2245-1439.1215 https://doi.org/10.13052/jcsm2245-1439.1215
5. M. Semmouni, A. Nitaj, M. Belkasmi. Bitcoin Security with a Twisted Edwards Curve. Journal of Discrete Mathematical Sciences and Cryptography, non, In press. HAL-02320909, https://core.ac.uk/download/237332050.pdf
6. H.Mayer, ECDSA Security in Bitcoin and Ethereum: a Research Survey, https://www.coinfabrik.com/wp-content/uploads/2016/06/ECDSA-Security-in-Bitcoin-and-Ethereum-a-Research-Survey.pdf
7. A.Takahashi, M.Tibouchi, Degenerate Fault Attacks on Elliptic Curve Parameters in OpenSSL, https://www.research.ed.ac.uk/en/publications/degenerate-fault-attacks-on-elliptic-curve-parameters-in-openssl
8. T.P.Dusane, Generation, Verification, and Attacks on Elliptic Curves and their Generation, Verification, and Attacks on Elliptic Curves and their Applications in Signal Protocol Applications in Signal Protocol, Masters Thesis, Rochester Institute of Technology, https://scholarworks.rit.edu/theses/10715/
9. W.Bi, X.Jia, M.Zheng, A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain https://arxiv.org/ftp/arxiv/papers/1808/1808.02988.pdf
10. A.J.DiScala, A.Gangemi, G.Romeo, G.Vernetti, Special Subsets of Addresses for Blockchains Using the secp256k1 Curve, https://www.mdpi.com/2227-7390/10/15/2746 10.
11. P.Urien, Innovative Countermeasures to Defeat Cyber Attacks Against Blockchain Wallets: A Crypto Terminal Use Case, https://arxiv.org/pdf/2303.17206
12. S.Zhai, Y.Yang, J.Li, C.Qiu, J.Zhao, Research on the application of Cryptography on the Blockchain, Journal of Physics, https://iopscience.iop.org/article/10.1088/1742-6596/1168/3/032077/pdf
13. M.M.Ulla, M.S.Khan, Preethi, D.S.Kakkari, Security and Performance Analysis of Elliptic Curve Crypto System using Bitcoin Curves, IAENG International Journal of Computer Science, 50(2), June 2023.
14. D.Aggarwal, G.K.Brennen, T.Lee, M.Santha, M.Tomomichel, Quantum attacks on Bitcoin, and how to protect against them, https://arxiv.org/abs/1710.10377
15. Y.Chen, H.Chen, Y.Zhang, M. Han, M.Siddula, Z.Cai, A Survey on Blockchain Systems: Attacks, defenses and Privacy Preservation, High-Confidence Computing,2(2022) https://doi.org/10.1016/j.hcc.2021.100048
16. Ethereum, https://www.ethereum.org
17. Shaldehi, A. H., Shaldehi, M. H., & Hedayatpanah, B. (2022). A Model for Combining Allegorical Mental Imagery with Intuitive Thinking in Understanding the Limit of a Function. In Indian Journal of Advanced Mathematics (Vol. 2, Issue 2, pp. 1–7). https://doi.org/10.54105/ijam.d1128.102222
18. Dhar, S., Biswas, A., & Singh, N. (2019). SciMath: A Mathematical Information Retrieval System using Signature Based B Tree Indexing. In International Journal of Innovative Technology and Exploring Engineering (Vol. 8, Issue 11, pp. 234–244). https://doi.org/10.35940/ijitee.k1298.0981119
19. Ghorai, A. (2023). Mung Seeds Under Constant Low Potential Difference During Post-Germination When Sprout Length Grows. In Indian Journal of Advanced Physics (Vol. 2, Issue 2, pp. 6–8). https://doi.org/10.54105/ijap.a1036.102222
20. Lata, K., & Khan, S. S. (2019). Experimental Analysis of Machine Learning Algorithms Based o n Agricultural Dataset f or Improving Crop Yield Prediction. In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 1, pp. 3246–3251). https://doi.org/10.35940/ijeat.f9308.109119
21. Wanjau, S. K., Wambugu, G. M., & Oirere, A. M. (2022). Network Intrusion Detection Systems: A Systematic Literature Review o f Hybrid Deep Learning Approaches. In International Journal of Emerging Science and Engineering (Vol. 10, Issue 7, pp. 1–16). https://doi.org/10.35940/ijese.f2530.0610722

AUTHOR PROFILE

Dr. Kannan Balasubramanian, is currently working as Professor in the School of Computing, Sastra University, Thanjavur. He received his M.Sc. (Tech) degree in Computer Science from BITS Pilani in 1989 and M. Tech degree in Computer Science and Engineering from IIT Bombay in 1991 and Ph.D degree in Computer Science from UCLA in 1999. He has worked on the areas of multiple access protocols for optical WDM networks and scheduling algorithms for input queued switches focussing on simulation of networks and network switches. He has published two books with IGI-Global and has published in many International Journals and Conferences. His areas of Interest are Computer Networks, Cryptography and Network Security, Cyber Security and Machine Learning for Cryptography.