Hindawi
Security and Communication Networks
Volume 2023, Article ID 2093407, 18 pages
https://doi.org/10.1155/2023/2093407

Research Article

Improved Multisignature Scheme for Authenticity of Digital Document in Digital Forensics Using Edward-Curve Digital Signature Algorithm

Gauri Shankar,[1] **Liwa H. Ai-Farhani**,[2] **P. Anitha Christy Angelin**,[3] **Parvinder Singh**,[4] **Abdullah Alqahtani**,[5] **Abha Singh**,[6] **Gaganpreet Kaur**,[7] **and Issah Abubakari Samori**[8]

[1]*Chandigarh University Mohali, Mohali, Punjab 140413, India*
[2]*System Analysis, Control and Information Processing, Academy of Engineering, RUDN University, Moscow, Russia*
[3]*PSNA College of Engineering and Technology, Dindigul 624622, Tamilnadu, India*
[4]*Central University of Punjab Bathinda, Punjab 151001, India*
[5]*Department of Computer Science, College of Computer Science, King Khalid University, Abha, Saudi Arabia*
[6]*Department of Basic Science, College of Science and Theoretical Study, Dammam-Female Branch, Saudi Electronic University, Saudi Arabia*
[7]*Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India*
[8]*School of Engineering Sciences, University of Ghana, Accra, Ghana*

Correspondence should be addressed to Issah Abubakari Samori; iasamori@st.ug.edu.gh

Received 18 August 2022; Revised 7 October 2022; Accepted 11 October 2022; Published 7 April 2023

Academic Editor: Keping Yu

At the moment, digital documents are just as important as paper documents. As a result, authenticity is essential, especially in legal situations and digital forensics. As technology advances, these digital signature algorithms become weaker, necessitating the development of digital authentication schemes capable of withstanding current security threats. This study proposed a scheme based on an asymmetric key cryptosystem and the user's biometric credentials to generate keys for digital signatures. A single document can be signed by multiple signatories at the same time under this scheme. The primary goal of this article is to create a safe and cost-effective multiignature scheme. To create keys for document signing and verification, the Edwards-curve Digital Signature Algorithm (EdDSA), especially Ed25519, is employed. The Edwards-curve Digital Signature Algorithm is used with blockchain technology to sign crypto wallets. The Python implementation of a scheme that enables platform independence. We performed performance, security, and comparative analysis to ensure maximum usability. The article's main findings are that the Ed25519 algorithm can be used in blockchain.

1. Introduction

In the present day, everyone prefers to use digital documents instead of paper and gives the same value in terms of contracts, agreements, and more. The authenticity of the document is crucial in case that is accepted as an official of any kind of agreement [1]. As the authenticity and integrity of digital documents are crucial, similar to traditional documents; therefore, digital signature algorithms and models have been suggested to achieve the goal of securing a digital document, with certain cryptographic signature techniques being more secure than others. Asymmetric key cryptography is the foundation for the majority of digital signatures. To establish safe and quick digital signatures, a variety of public-key methods, including Rivest–Shamir–Adleman (RSA) and Elliptical Curve Cryptography (ECC), are utilized in the modern day [2]. There are three forms of cryptography in general. The first type of cryptography is symmetric key cryptography. Using just a single shared key, the receiver and the transmitter encrypt and decode messages in this encryption system. While the symmetric key systems are quicker and easier to use, they need secure key encryption between the sender and recipient. The most popular symmetric key encryption method is technology (DES). Hash functions come in second. There are no keys used in this technique. Since a hash value with a predetermined length is calculated using plain text, it is difficult to interpret the contents of the simple text. Hash algorithms are used by several operating systems to protect passwords. Asymmetric key encryption is the third option. The data are encrypted and decrypted using a pair of keys in this system. For encryption and decryption, public and private keys are utilized. Public and private keys are distinct. Even if everyone is aware of the public key, only the intended receiver can decode this since he has access to the private key.

By maintaining data integrity, you reduce your vulnerability to threats. Neglecting security is negligent since anybody might become a breach victim. Data integrity is a group of vital controls that guarantee the assigned data in a system are secure, unmodified, and safe. Emergencies can happen. Therefore, even after the problem has been fixed, data integrity defines how reliable the data are. Data integrity also involves adhering to online laws and norms, particularly one as important as the General Data Protection Regulation (GDPR). It is essential to use its strategies if you are creating an Internet startup. In the lack of data integrity, online theft is prevalent and has detrimental effects. Most often, data integrity is compromised after a deadly accident, an emergency, or a breach. In order toTota breaches, a reputable firm has established a set of regulations for GDPR compliance. Any online company that processes data in any way is required to follow the standards set to avoid other unintended effects. Data security is important not only just for corporations but also for people. You must evaluate internal handling to prevent sensitive information. The likelihood of data being miscategorized or altered is decreased by validating the data and monitoring the system for error checks. A secure database with total integrity will continue to provide security against bad intent no matter how long you keep or access critical information. Authentication procedures only make your sign-in procedure more rigorous. Authentication procedures in no way further encrypt your information. It is undeniable that authentication procedures reduce a user's vulnerability to identity theft, but it is equally undeniable that they have several drawbacks.

Asymmetric key authentication is also a possibility. The term "signature scheme" refers to both asymmetric encryption and the asymmetric key equivalent of a message authentication code. A signature system includes three operations: key generation, signing, and verification, much like a message authentication code does. The creation of signature methods facilitated authentication. The verification key must be made public to achieve this purpose, and it is often issued in a certificate, which we will refer to as cert (IDASV), where IDA stands for the identification of the key holder of S and V stands for the verification key that corresponds to A. The certificate is issued by a trustworthy organization called the certificate authority. The only duty of the certificate authority is to link parties. With the use of a digital signature, the sender may ensure that the communication the recipient receives was indeed sent by the intended recipient. The digital signature standard is often the foundation of digital signature algorithms (DSA) (DSS). A digital signature enables mathematical validation of the authenticity and integrity of communication, software application, or digital document. Digital signatures, also known as electronic signatures, attest to the communication's sender's identity. Authenticity and integrity should be ensured while making digital transactions since data can be altered or someone could claim to be the sender and anticipate a response. The verifier receives data and a digital signature. The verification algorithm is used to process the digital signature and the public key (verification key), which provides some value. The very same hash function is also used to hash the incoming data, yielding a hash value. To construct a digital signature, e-mail programs and other signing algorithms create a one-way hash of the digital data that need to be signed. The signing method then uses the private key to encrypt the hash value (signature key). This encrypted hash is a part of the digital signature along with additional information like the hashing algorithm. This digital signature is appended to the data before it is sent to the verifier. It is preferred to encrypt the hash instead of the full message or document since a hash function may convert any arbitrary input into a substantially smaller fixed-length result. This saves a lot of time since only a brief hash value needs to be signed now rather than a large message, and hashing takes far less time than signing. In the digital signature algorithm, the sender first calculates a message digest using a secure hash algorithm over the original message (M), now the sender encrypts this hash code using his/her private key (Kpriv). Encrypted hash code is called a digital signature; furthermore, the receiver uses a public-key (Kpub) of the sender to verify the digital signature [3, 4]. Digital signatures provide security services, such as authenticity, integrity, and nonrepudiation, but do not provide privacy of the message. Therefore, the basic characteristics of a signature are authenticity, nonforgery, no reusability, and irrevocability. These characteristics ensure the security of the message from different cryptographic attacks such as masquerade, modification, and fabrication. [2, 4]. Numerous digital signature systems have been put out over the last three decades; however, as technology advances, the security services they offer can be readily undermined. At present, the versions of digital signatures originated from public-key cryptography algorithms. However, some of these signature algorithms are weaker in time such as RSA [5]. Therefore, with the new generation of technology, more secure and fast approaches to digital signatures are required. Hence, a lot of theoretical and experimental progress has been made continuously in the area of digital signatures [6]. In comparison to conventional signature methods, the threshold signature scheme, or TSS, can provide stronger security levels. It is becoming more and more common among the suppliers of cryptographic services as a means of ensuring safe data flow without interruption from outside parties such as hackers and scammers. A threshold signature scheme produces the same results as a single-key digital signature scheme, but somehow it uses MPC to build an interactive multiparty protocol that allows for the formation of private key shares and the fabrication of a single digital signature.

In 1993, Bruce Schneier developed the very first symmetric encryption algorithm, Blowfish. The symmetric key encryption uses a single encryption key to encrypt and decode data. To transform sensitive information into ciphertext, the encryption technique employs sensitive documents and the symmetric encryption key. Blowfish, as well as its sequel Two fish, competed to replace the Data Encryption Standard (DES) but were unable to do so owing to the low size of their blocks. Blowfish embedding capacity is 64, which is considered entirely unsafe. Two fish solved the challenge by creating a 128-bit block. Blowfish is much faster than DES, but it trades speed for security. To replace the Data Encryption Standard (DES) technique, which hackers later learned was easily broken, Triple DES was created. Traditionally, Triple DES was the most used symmetric method and the industry's preferred benchmark. Three separate 56-bit keys are used in triple DES. Despite having a total length of 168 bits, experts think that a key strength of 112 bits is more precise. Despite being gradually phased down, the Advanced Encryption Standard has largely taken the role of Triple DES (AES). By eliminating one-a-round transmission in a typical scheme, ASMS, an improved Schnorr-based multisignature approach, provides public key aggregation. As a result, it is appropriate for e-business and e-government scenarios. We used chain code technology to develop our solution as an application on Fabric, an enterprise blockchain platform [7]. Similar to RSA-like signature systems, the Schnorr signature technique enables data recovery straight from the signature. The quantity of the recovered information is changeable. The main advantages of the quality improvement include shorter keys with equivalent cryptographic strength, shorter signatures, and fewer amounts of delivered data overall. The time required to produce and validate sensitive information that is dependent on the method used and is built on elliptic curves is therefore decreased by using a safe and reliable digital signature approach that incorporates a privacy service [8].

1.1. Digital Signature.

A digital signature is an electronic signature having the same value as the written signature and that can be verified by the original signatory. Moreover, it can be used to identify whether or not information has been manipulated since it was generated after the signature. To create a digital signature, specific protocols are used, known as digital signature algorithms [9].

2. Digital Signature Algorithms

The discrete logarithm problem and modular exponentiation are mathematical ideas on which the Digital Signature Algorithm (DSA), a Government Information Processing Standard for digital signatures, is based. The DSA signature system combines the Schnorr and ElGamal signature techniques. The National Institute of Standards and Technology (NIST) first became aware of the DSAS in 1991 and proposed it for adoption as the Digital Signature Standard (DSS). Although the DSA is copyrighted, NIST offers it royalty-free for everybody. According to NIST, older versions of the digital signature algorithm are only employed to validate signatures and not to generate signatures [10]. The DSA performed four major operations: key distribution, key generation, signature creation, and signature verification. For all these operations, an image illustration is provided in Figure 1.

2.1. Key Generation.

Key generation happens in two steps. The first stage entails choosing algorithmic parameters that may be shared by all users of the system, and the second stage entails calculating and creating a single-key pair for an individual user that consists of the public (Pubic-key) and private key (PrivKey). The message is signed using the private key, and the public key is employed to validate the signature.

2.2. Key Distribution.

Private key any trustworthy or secretive method that will be employed to validate the signature can be used to communicate Pubic-key with the recipient. The sender or signatory must not know the private key, abbreviated PrivKey.

2.3. Signature Generation.

The first stage in creating a signature is to create a message's hash, or H(M). The generated hash is added with the private key $Priv_{key}$ of the signatory and now multiply the generated value to some random number K^{-1} so at last, we got the value as the signature S of that message. Therefore,

$$\mathrm{S} = \left(K^{-1}\left(\mathrm{H}(\mathrm{M}) + \mathrm{Priv}_{\mathrm{key}}\,\mathrm{r}\right)\right). \tag{1}$$

The signature is (r, s).

2.4. Signature Verification.

This operation is performed on the receiver side. Verification V requires a signature (r; s), message M, and a public key *PubKey* of the signatory. So

$$V = H(M) + \mathrm{Pub}_{\mathrm{key}}(r, s). \tag{2}$$

2.5. RSA.

The RSA stands for Rivest–Shamir–Adleman, who proposed this algorithm in 1977. The RAS is an asymmetric cryptographic algorithm a type of Public Key Cryptographic. This allows other users to encrypt data with the user's public key (PubKey), which is kept in the system and may be shared with them and transfer it over the network. Only the person whose public key (PubKey) was used to encrypt, or whose private key (PrivKey), may perform the decryption operation. The security of the cryptosystem entails the amount of time and effort required to factor in huge numbers. The hybrid key configuration strategy utilizes less energy on end-user mobile devices whilst dramatically improving security over our previous pure symmetric key-based methodology.

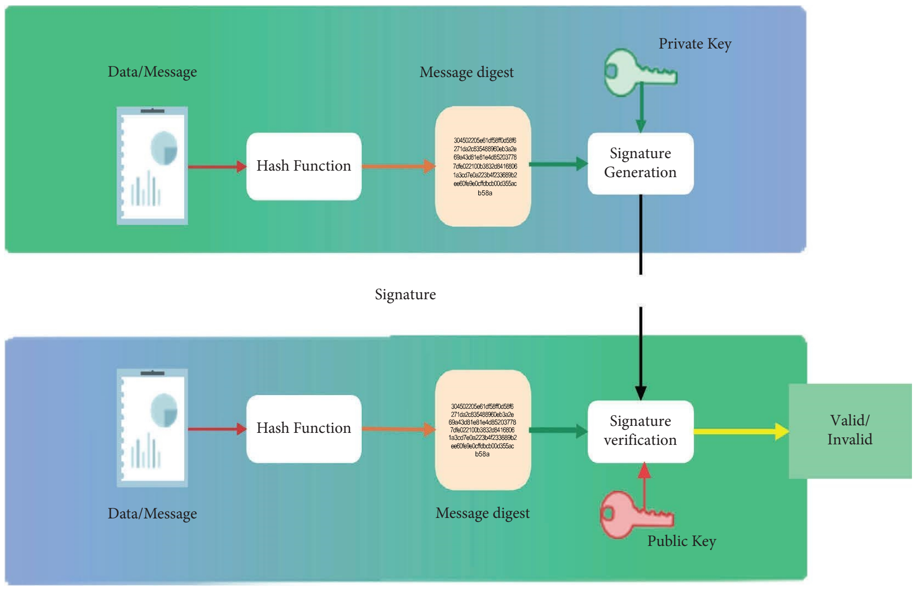

Figure 1: Process of digital signature.

The pure symmetric key-based approach utilizes less node energy than the hybrid key setup technique. However, one expensive elliptic-curve scalar multiplication of a random point is shifted to the security manager side and replaced by one low-cost modular multiplication, one modular addition, and one symmetric key decryption because we verify that the sensor's private key is a public key formed by a linear combination of the static key and the ethereal key rather than a multiplicative combination as in other ECC-based pure public key protocols. As a result, our hybrid key setup approach is quicker and more energy-efficient than prior public key-based techniques. The encoded text and blocked cipher in this plain text are represented by integer values that range from 0 to N 1 for a given value of N. A number is given to every communication. Every block has a binary value that must be less than or equal to N and is used to encrypt the message. The three steps of the RSA algorithm are key creation, encryption, and decryption [11]. Figure 2 in the next section illustrates how the cryptosystem operates.

2.6. RSA Digital Signature. Such RSA Public-key cryptosystem too is employed in the digital signature procedure, which involves singing and confirming the message's authenticity. This can be conducted by a discrete algorithm, modular exponentiation, and computational difficulty of the algorithm. In this, there are three steps or operations that are performed, first key generation, signature, and verification [12]. These three processes are explained is as follows:

Key generation: The RSA is using a 1024 to 65536 bits long Key. Here, we are generating a key for the 128-bit security level, so 3072 bits are required. Through this process, Private Key $RSA_{Privkey}$ (N, E), and public key RSA_{Pubkey} $(N, \textit{generates dates.})$ in which N is the number of bits of the key and E or D represents the exponents.

Signature generation: The process of generating a signature using the private key of the user or signatory. For this, first calculate the hash H of the given message M and then encrypt it with the exponent D to generate signature S.

$$H = \text{hash}(M)S = H_D(\text{modn}). \tag{3}$$

Here, the H and S should be in the range between 0, . . ., and N

And

Signature verification: At the time of signature verification, the message M, the public key RSA_{Pubkey} (N, E) of the signatory, and the signature S are required. In the verification process, first, calculate the hash H of the signed message M and then run the decryption process with exponent E, so

$$H = \text{hash}(M)H^{'} = S_D(\text{modn}), \tag{4}$$

and last, compare H and H' by

$$H^{'} = S_E(\text{modn}) = (H_D)_E(\text{modn}) = H. \tag{5}$$

If the condition is satisfied, then the signature is authenticated else not.

2.7. ECC. The authors of [13] proposed a public-key cryptosystem using the concept of elliptic curves known as Elliptic Curve Cryptography (ECC) in 1985. The algorithm is operating on discrete logarithm issues and the algebraic structure of the elliptic curve over finite fields. ECC is capable of performing all key operations, signatures, and key exchanges in a public-key cryptosystem. Comparatively speaking, it has a smaller key size than RSA. The private key used in the ECC is a straightforward random integer

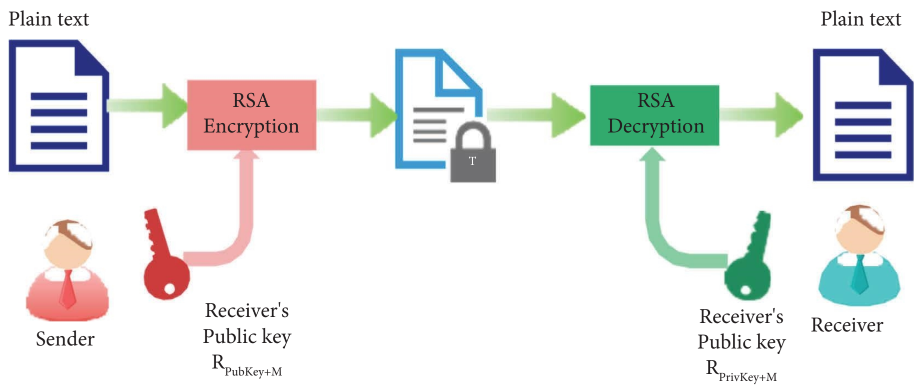
Plain text
RSA Encryption
RSA Decryption
Plain text
Sender
Receiver's Public key R_PubKey+M
Receiver's Public key R_PrivKey+M
Receiver

Figure 2: Process RSA cryptosystem.

number. The PrivKey generating process is safe and collision-resistant. The private key for the ECC can be any integer, hence, the key has a particular length size. The public key, on the other hand, is the integer obtained from the curve's elliptic curve point pair. EC points can be compressed into a single point and it can be odd or even. From the EC curve can be generated a different elliptic curve that will be having different levels of security performance, key length, and different types of ECC algorithms. Another characteristic of the ECC is that, while offering the very same level of security with a short key length, it uses fewer resources than an asymmetric cryptographic algorithm. Another feature of this cryptographic system is that the public keys have a trapdoor mechanism, making it impossible to extract the private key from the public key [2]. An elliptical curve is expressed in mathematical form as follows:

$$y^2 = x^3 + ax + b. \quad (6)$$

In the ECC public-key cryptosystem, the following types of digital signature algorithms are generated from the EC curve, which are mostly used these days.

ECDSA

EdDSA

Both algorithms are discussed in the following sections.

2.8. ECDSA. The Elliptic Curve Digital Signature Algorithm (ECDSA) is an elliptic curve cryptography-based signature algorithm (ECC). ECDSA is based on elliptic curve cyclic groups over limited fields and the problems of the ECDLP problem (elliptic curve discrete logarithm problem). The ECDSA sign/verify method functions as follows and is based on EC point multiplication. ECDSA keys and signatures are lower than RSA keys and signatures for the same security level. In terms of security, a 256-bit ECDSA signature is comparable to a 3072-bit RSA signature. The elliptic curve digital signature method is one sort of electronic signature (ECDSA). It is essentially just used as identifying documents by bitcoin traders. The heart of the ECDSA key-creation process is complexity ECDSA methods. It is theoretically difficult to breach an ECDSA code, although hackers will undoubtedly attempt to do so. Websites strive to load pages in under a second. The little keys used by ECDA help speed up a website. You must utilize ECDSA if you are working in the bitcoin ecosystem. ECDSA does the same task as any other digital signing signature, but more quickly. This is so that ECDSA can offer the same security level as any other digital signature method while using fewer keys. ECDSA certificates, a type of electronic document used to validate the certificate's owner, are produced using ECDSA. Credentials contain the signature of the certificate's issuer, which is a trusted institution, information about the key used to construct the certificate, information about the certificate's owner, and certificate data. This trustworthy issuer is typically a certificate authority having a signed certificate that can be tracked back to the original giving certificate authority via the chain of trust.

This algorithm is working on finite fields in the classical Weierstrass form over elliptic curves. Therefore, these curves are represented by the elliptic curve domain parameter that is specified by various cryptographic standards. The elliptic curves that are used in cryptography can be defined as follows:

Point G is a point for scalar multiplication on the curve that is multiplied by the integer by an elliptic curve point.

G is generating another point n which is the subset of the elliptic curve point that expresses the length of the private key such as 256 bits.

For this example, the 256-bit elliptic curve secp256k1 has

$$n = 115792089237316195423570985008687907852837564279074904382605163141518161494337. \quad (7)$$

Generator point G x = 5506626302227734366957871889516853432625060345377759417550018736038911672924 0, y = 3267051002075881697808308513050704318447127338065924327593890433575733748242 4

The process of key generation, signature generation, and verification of the ECDSA are the same as the EdDSA, which is discussed in the next section.

2.9. EdDSA. Based on the ECC, EdDSA is a variation of the Schnorr signature method [14, 15]. The private key (Prk) in EdDSA is a randomly created hashed number at the encryption point, and the public key (Puk) is derived from the private key. Ed25519, an EdDSA point that uses the secure hash method to produce the key pair and establish a digital signature, was proposed by Bernstein et al. in 2011 [16] (SHA-512). The key pair is created via a cryptographic hash function that EdDSA utilizes, and these hash functions must possess the following four key characteristics.

Hash functions are one-way functions; therefore, it is easy to compute the hash value for a given message, but the reverse is not possible.

Preimage resistance-for a given message x and hash code $h = H(x)$, it is computationally impossible to find a message y such that $x \neq y$ with $h = H(y)$; second preimage resistance-for a given message x, it is computationally impossible to find another message y with the same hash value.

Strong collision resistance: it is computationally infeasible to find a pair of messages (x, y) with the same hash value.

2.9.1. The Curve Ed25519. The Ed25519 is using SHA-512 for hashing of data at the elliptic curve point 25519, as suggested in the nomenclature. This algorithm generates a key pair with each key size 256 bits long, and the hash value is 512-bit long. This algorithm is fast and more secure against many cryptographic attacks compared to other public key cryptographic algorithms [15]. In Ed25519, the private key $Ed_{privkey}$ is generated from a random integer, further private key and curve generator C on the elliptic curve are used to generate the public key Ed_{pubkey}.

$$Ed_{\text{privkey}} = \text{Integer}Ed_{\text{pubkey}} = \text{EdDSA}_{\text{privkey}} * C. \tag{8}$$

The private key ($Ed_{privkey}$) is used in the generation of the EdDSA signature (*Design*) for any message M that is explained step-by-step as follows where H is a hash function:

Generating a secreted integer, I from H (H ($Ed_{privkey}$)|| M).

Calculating the public key point r from I by multiplaying with C as $r = i * C$

Calculate the hash

$$h = H\left(r + Ed_{\text{pubkey}} + M\right)\text{modq}. \tag{9}$$

(The q is prime in the range $[2^{b-4}, 2^{b-3}]$ where b is an integer ≥10 on the curve).

Calculate integer:

$$S = \left(i + h * Ed_{\text{privkey}}\right). \tag{10}$$

The calculated signature is $\{r, S\}$.

A simplified structure of the above process can be formulated as

$$Ed_{\text{sign}}\left(M, Ed_{\text{privkey}}\right) \Rightarrow \{r, S\}. \tag{11}$$

After signing the message and generation of a signature for it, the receiver must verify the Ed_{verif} EdDSA signature. (Ed_{sign}) by using the Ed_{pubkey} of the signer. The process for verification of EdDSA signature is explained step-by-step as follows:

Calculate the hash:

$$h = H\left(r + Ed_{\text{pubkey}} + M\right)\text{modq}. \tag{12}$$

Calculate the first point of the curve:

$$Pnt_1 = S * C. \tag{13}$$

Then, calculate the second point of the curve:

$$Pnt_2 = r + h * Ed_{\text{pubkey}}. \tag{14}$$

Compare

$$Pnt_1 = Pnt_2. \tag{15}$$

From the above steps, a simplified equation of the verification process of the signature can be presented as follows:

$$Ed_{\text{verif}}\left(M, Ed_{\text{pub}}, r, S\right) \Rightarrow \frac{\text{valid}}{\text{invalid}}. \tag{16}$$

The Ed25519 is having some properties that make it more useable than other versions of EdDSA. That is the motivation behind using this specific version of the proposed scheme. Those properties are discussed as follows:

EdDSA provides more security than ECDSA as that key generation is possible through not only random numbers but also through other input factors.

Some of the security issues with discrete log signatures are resolved by deterministic signatures.

In batch verification with numerous signatures, EdDSA has superiority compared to ECDSA.

EdDSA is proven secure for the next two decades in the chain and other security effects.

In the next 20 years, it is unlikely that 256-bit ECDLP instance-solving quantum computers will be created.

Cryptographic hash functions also exhibit an avalanche effect that makes the whole information invalid if a single bit is changed in the hash string. Therefore, EdDSA becomes a secure scheme for digital signatures due to the use of the hash function.

3. Comparison between RSA, ECDSA, and EdDSA

In this section, we compare RSA, ECDSA, and EdDSA which are the popular digital signature algorithms these days. These algorithms are working on public-key cryptography. A comparison of Table 1 is presented as follows:

3.1. Cryptographic Hash Functions. Hash functions are used in a programming language to convert text (or any other information) into integer numbers. Various inputs generally correlate to different outputs; however, sometimes a collision might occur. Textual or binary data are converted into a fixed-length hash value through cryptographic hash functions that are proven to be collision-resistant and irrevocable. SHA-256 is an example of a cryptographic hash function:

$$SHA3-256(\text{"}hello\text{"}) = \text{``}3338be694f50c5f338814986cdf0686453a888b84f424d792af4b9202398f392\text{''}. \tag{17}$$

Mainly, cryptographic hash functions are commonly employed to encode data without disclosing it due to their inability to be reversed [17]. Encryption and hashing have served as the foundation for new security modules, among other network security developments. One of the most used hash algorithms is the safe hash algorithm with digest size of 256 bits or SHA 256. Although there are numerous variations, SHA 256 has been the most often used in practical applications. The Secure Hash Algorithm, or SHA, is a family of algorithms that includes the SHA 256 algorithm. The NSA and NIST collaborated to publish it in 2001 as a replacement for the SHA 1 family, which was gradually becoming less resistant to brute force assaults. The final hash digest value, represented by the number 256 in the name, is significant.

3.2. Identity-Based Digital Signature. User identifications such as biometric characteristics, identity cards, social security numbers, or emails are used in the identity-based digital signature system to produce the public-private key pair. Key pair generation is expensive in public key infrastructure, and this infrastructure is susceptible to key search, brute force, and man-in-the-middle (MIMT) attacks. A trustworthy key exchange center generates keys in an identity-based digital signature system utilizing the user's identification. As a result, a trustworthy third party lowers the total cost and increases the security of key creation in the context of persons. Therefore, this signature scheme's main benefit is that it lessens the vulnerability of the public key infrastructure. Furthermore, an identity-based digital signature can also be used in multisignature models, known as the identity-based multisignature (IBMS) model [18]. Figure 3 clearly describes the process of identity-based digital signature.

3.3. Multisignature Scheme. A multisignature scheme provides a way that allows several signers to sign the same message M at the same time using their respective private keys such as $Pk_1, Pk_2, \ldots, and\ Pk_n$. In this scheme, the private keys of all signatories are combined to generate a single private-key such that $Pk = (Pk_1, Pk_2, \ldots, Pk_n)$, this reduces the time and cost of signing the document compared to the process in which every party individually signs. However, the security level and the size of the signature are the same as the standard signature. The same process is followed to combine the public keys of the signer for the verification of the received document.

Furthermore, the overall transmission time in the multiignature schemes is less compared to the individual signature schemes. Figure 4 clearly describes the multisignature scheme.

In the current scenario, ECDSA is a widely used approach for digital signatures. ECDSA is more secure and has a better performance in terms of time and space complexity [4] in comparison to other asymmetric digital signature algorithms. Another ECC-based digital signature algorithm is EdDSA. EdDSA is faster and more secure than ECDSA, this motivates us to develop an identity-based multisignature scheme using EdDSA for digital documents. EdDSA is generally used in blockchain or in the cryptocurrency wallet to generate hashed signatures [19–21] due to its speed and security features. Furthermore, we know that digital documents can be forged in the absence of digital signatures. In the current digital era, technology is getting advanced, and with this, the algorithms for digital signatures become more vulnerable. This also motivates us to develop a more secure model for digital signatures. Multisignature threshold schemes combine the qualities of threshold group-oriented signature schemes with multisignature schemes to provide a signature technique that allows extra group members to sign any message collectively. Genuine multisignatures, as opposed to threshold group signatures, enable the public to identify particular signers, removing their anonymity. The distributed-key management infrastructure (DKMI) includes the distributed-key generation (DKG) and distributed-key redistribution/updating (DKRU) protocols. The round optimum DKRU protocol provides group members with a way to recognize dishonest or flawed shareholders in the first round, hence eliminating repeated protocol executions, which resolves a significant issue with current secret redistribution/updating techniques.

3.4. Our Contribution to the Article. The article is representing a scheme for a digital signature on documents from multiple signatories that sign a single document at the same time. To prove that our proposed system is fast and secure, we provide a performance and security comparison in the study with earlier suggested plans. The comparison suggests that the manuscript provides a fast and secure identity-based

Table 1: Comparison of RSA, ECDSA, and EdDSA schemes.

	Properties	RSA	ECDSA	EdDSA
1	Security bits			
	80	1024	160	160
	112	2048	224	224
	128	3072	256	256
	192	7880	384	384
	256	15360	512	512
2	Performance	Slow due to long key size	Fast	Fastest
3	Popularity	Widely used	Not much used	New and widely used

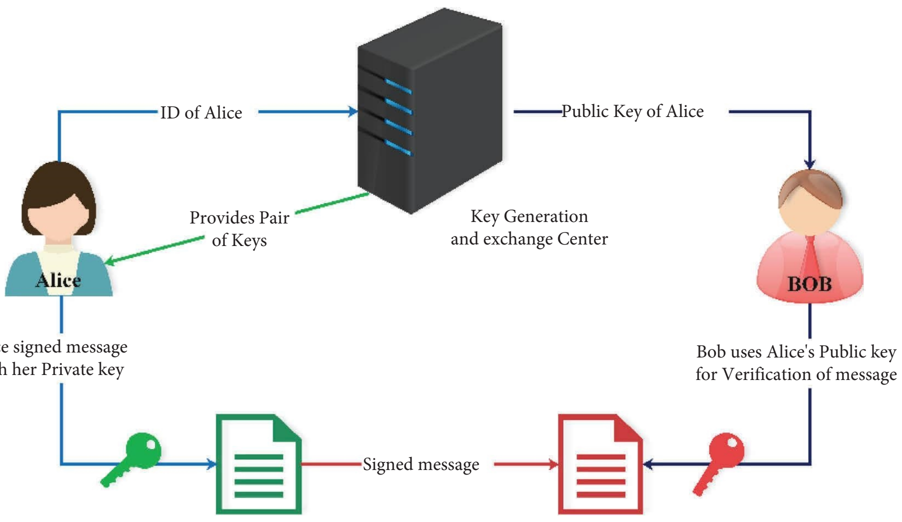
ID of Alice
Public Key of Alice
Provides Pair of Keys
Key Generation and exchange Center
Alice
BOB
Alice signed message with her Private key
Bob uses Alice's Public key for Verification of message
Signed message

Figure 3: The basic architecture of identity-based digital signature (IDBDS).

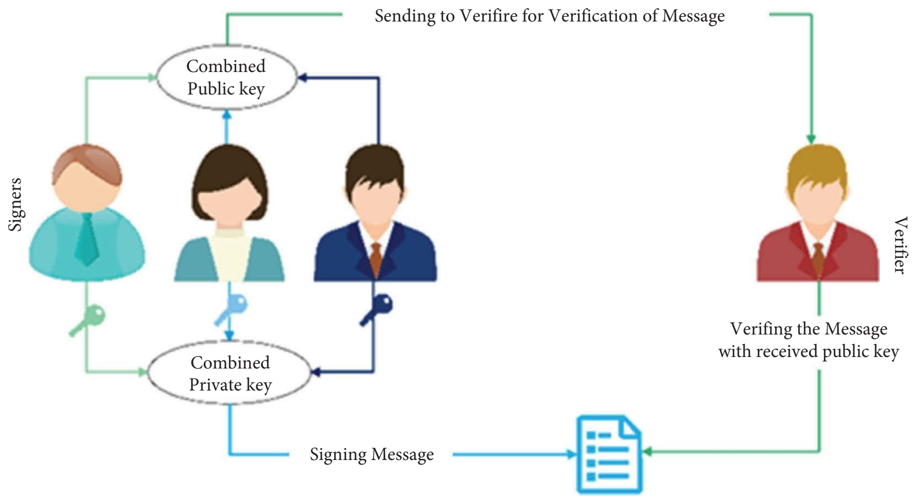
Sending to Verifire for Verification of Message
Combined Public key
Signers
Verifier
Combined Private key
Verifing the Message with received public key
Signing Message

Figure 4: The process of the Multisignature scheme.

multisignature scheme to sign digital documents to save them from forgery.

The major contributions of the scheme are listed as follows:

We introduce a fast and more secure identity-based multisignature scheme for a document using the EdDSA signature algorithm

The EdDSA signature algorithm is the first time implemented without a blockchain technology for digital signature and verification of documents

The scheme is very cost-effective, secure, and fast than the other schemes, because of the EdDSA signature algorithm is used without the blockchain technology and implementation of the scheme in Python makes it platform-independent

The credentials such as an Aadhar number and a fingerprint are used with a random number generator in the scheme that generates a unique and secure key pair for each signatory.

3.5. Organization of the Article. The article has the following sections:

Section 1 in this section, the article provides an introduction to the field of research and related tools and technologies and mentions the author's contribution and suggests how this research work adds more knowledge to the relevant field of research.

Section 2 discusses the previously proposed schemes related to digital signature algorithms and accordingly proposes the research work.

Section 3 discusses the overview and algorithms of the proposed scheme along with the process diagram of the implementation of the scheme. Moreover, providing the justification that why we are selecting the specific algorithm ED25519.

In Section 4, in this section, the implementation part of the proposed scheme is discussed properly. Also, provide information tools and packages used in the implementation.

In Section 5, the security analysis of the proposed system is discussed. Different kinds of attacks performed on the digital signature algorithms are discussed in the proposed scheme.

Section 6: The result is described in this section, which also explains the advantages and limitations of using our proposed model in comparison with the previously proposed scheme.

Section 7: This section is provided a summary of the whole work and suggests the future scope of the work.

3.5.1. Previous Work. Recently, in 2019, Saho and Ezin [2] have given a comparative analysis according to the significant use of RSA-based and ECC-based digital signature algorithms. This comparative analysis shows that the key length of ECC is much shorter than the key length of the RSA algorithm for the same security level. This article also claims that the time complexity is antiproportional to the key length in the case of ECC for the generation of digital signatures. El-Rahman et al. in 2018 [4] suggested a cloud-based digital signature technique to safeguard IoT applications. The secure hash algorithm SHA-512 is used in this method, which is based on the ECDSA. In this approach, the message digest is produced using a hash function (SHA-512), and the digital signature is produced using ECDSA. The authors asserted that this technique's structure minimizes the time complexity of creating a digital signature. Reference [16] describes how to create an ED25519 signature algorithm using the Fiat–Shamir paradigm. The authors assert that the performance and security analysis assures that it cannot be faked. The author discusses several clamping of private scalars and the nonprime order group. Researchers also demonstrated that Ed25519-IETF is SUF-CMA compliant. All Ed25519 schemes are resistant to key replacement attacks.

Using Ed25519 as a case study, researchers [22] examine several EdDSA authentication strategies; one is speed-optimized, whereas the other prioritizes a little amount of RAM. The speed-optimized version uses a joint-sparse form to describe the two scalars and performs the double scalar multiplication in parallel. Due to the Frobenius endomorphism, Koblitz curves, a particular class of elliptic curves, perform better while calculating scalar multiplication in elliptic curve encryption. The performance of a single scalar multiplication has increased thanks to the double-base number system technique for Frobenius expansion. Addition, squaring, multiplication, and inversion are the four fundamental arithmetic operations carried out by the binary arithmetic processor to execute point multiplication. The standard base representation is used for all arithmetic operations. A single clock cycle can be used to conduct addition and squaring. In the standard basis representation, adding is an exclusive OR (XOR) action whereas square rooting is a cyclic shift operation. The mathematical calculation of KP-IQ is split into two parts by the memory-optimized variant, according to the authors: a fixed-based scalar multiplication using a traditional comb technique with eight precomputed points and a differential scalar multiplication using the typical Montgomery ladder on the birationally-equivalent Montgomery curve. The segregated approach is 24% slower than the simultaneous strategy, but it uses 40% less RAM, according to the authors' research using a 16-bit ultra-low-power MSP430 microcontroller. This makes the divided approach desirable for "lightweight" cryptographic libraries, especially when both X25519 key exchange and Ed25519 signature creation and verification are required.

Researchers [23] propose an ECC-based ID with multiple signatures, which are much more robust than forgery that can be detected in the proposed approach because the secret key is generated from the identity hash and a random number $D_i = H(ID_i|r)$. This may be further enhanced by using a blind signature and encryption on the message. It may also be configured for multiple receivers. Authors [24] present a novel variation of signature construction using sequential OR proofs. They aim to obtain strong protection against adaptive corruptions, optimize efficiency, and immediately achieve strong existential unforgeability building signatures in the nonprogrammable random oracle model (NPROM). This results in a little different construction, and they ema ploy of t distinct and extra lossless format recognition scheme features. Signatures that provide strong multiuser security against adaptive corruptions are a popular building component for highly secure authentication schemes exchange systems. The authors used the ECDSA algorithm to achieve their goal of strong multiuser security against adaptations.

The identity-based digital signature scheme proposed by Rahmawati et al. [3] uses a fingerprint scan to generate the key pairs. Furthermore, the RSA-based multisignature scheme was proposed by Bellare and Neven [18], in which the user's identity is used to generate the key pairs. In this study, the authors provided security notation with a random oracle model to prove that the proposed scheme was unforgeable and a trapdoor system for the key pairs. The trapdoor system is a one-way computation which means it can be computed easily in one direction, but the inverse of this function is computationally infeasible to compute. In 2019, an identity-based multisignature model was proposed in [1], this model uses the RAS-based digital signature algorithm. This model uses e-mail ID and Aadhaar number as an input to generate the key pairs. The authors analyze the security of the given model against different attacks and claim it is secure against different cryptographic attacks. Another algorithm introduced by Ahmed et al. [25] is based on SHA-256 and RSA digital signature algorithms. This algorithm uses face biometrics to generate the key pairs. In this scheme, signers register with the system using their facial identities. Now, the system uses these face identities to generate the key pair and create the digital signature. In 2017, for more flexibility, a multisection multisignature model was proposed [26], that overcomes the restrictions of signing the whole message by every signer. This scheme provides the facility for a signer to sign the specific section at the time of multisignature. To provide security services-confidentiality, authenticity, and integrity, a cloud-based double signature scheme DS-SHA256 is proposed by the authors [27], in which the RSA digital signature is used with the SHA-256 two times to provide a strong security to the authenticity of the document while it is uploaded on the cloud server. Complexity and power consumption is high, but the authors represent the encryption and decryption process as taking less time than normal RSA and AES.

Jin et al. [28] proposed an identity-based combined signature and encryption (IBCSE) model. This model is based on Boneh and Franklin's encryption and Cha and Cheon's signature algorithms. The security analysis for IBCSE shows that the proposed scheme is secure against different cryptographic attacks such as chosen identity attacks. Another model for identity-based digital signature is proposed in [29], which is based on ECC. As the presented model is based on the ECC with the nonpairing scheme, hence, the cost of computation for signature and verification is very less.

A multisignature model for a specified group of verifiers is proposed with an improvement by [30], such that examines Zhang and Xiao's scheme against rogue key attacks. Furthermore, a tightly secure multiparty signature scheme was proposed in [31]. Recently, in 2020, Ra- Rajkumar, and Juneja presented a security protocol [32] for multisignature authentication and key management. This model uses Newton's Foreword Interpolation for chaining keys from multiple signatories. Furthermore, Nagashima et al. [33] have presented a detailed explanation of the vulnerabilities of digital signatures. In the presented article, the authors have discussed different types of vulnerabilities of digital signatures at different stages and discussed the solutions for these vulnerabilities. At the moment, based on the brief overview of already proposed schemes, to overcome the security threats, in this work, in the article [34], the authors create and test the model of digital signature for the artist to sign their creative content and integration with the creative content license. In the development of this model, the authors used ECC algorithms to generate the signature of a single or multiple creators and take the creator's work with the creator's ID as input depending on to create the public and private keys. The key pair creators are eligible to sign their creative content after the signature, they get permission to access or modify their content on the developed platform. Tempered with the content can be identified from the verification of the signature.

Authors [35] proposed hardware implementation of a field-programmable gate array for the EdDSA. This architecture is provided high efficiency and performed well in comparison to AES-128. Based on the test and analysis results, the authors claim that the efficiency of the EdDSA is increased and improved by greater than 84% in comparison with the previous work. The speed of generating signatures can be gone more than 8x speedup. The proposed scheme generates 62,000 digital signatures per second. The side-channel attack countermeasures are included in the architecture. In high-performance architecture and efficient processed 2,200 signs and 5,100 and 15 verifications per second.

We discuss the implementation of a more secure and fast identity-based multisignature model using the EdDSA algorithm.

4. Proposed Model

The proposed scheme uses multiple identity sources to generate the key pairs. Therefore, for this purpose, we are using the fingerprint and Aadhaar number of individual signatories as identity sources for key generation. The Aadhaar number is a unique identity of the Indian citizen similar to the social security number in the US. Aadhaar uses an e-KYC service for the authentication of the cardholder. A central database is used to store the personal and biometric information of the cardholder, and this database is maintained by the Unique Identification Authority of India (UIDAI). The fingerprint is a unique biometric that cannot be the same for two persons, even in the case of twins. Other biometrics such as iris, vein pattern, retina, and gait recognition are also unique, but require high-cost equipment and take more time to acquisition in comparison to fingerprints. Furthermore, in the Aadhaar database fingerprints are also stored along with the name, address, and photo of the person, and this provides proof of identity in the proposed model [36].

In the proposed scheme, signatories create public and private-key pairs based on their identities. These key-pairs are required at the time of signing and verification. Digital signatures are generated using a private key and verified using the public keys of signatories. Figure 5 visualizes the architecture of the proposed scheme.

The working process of the proposed model is clear in Figure 5. The proposed model is based on EdDSA and by using this model, multiple signatories can digitally sign a document in a single attempt. In the first step, signatories provide fingerprint (Fprint) and Aadhaar number (Anum) for generating public key ($EdDSA_{pub}$) and private key ($EdDSA_{priv}$) pairs. The Aadhaar number is linked to the biometric database created by the government of India, so there is no need for the verification of the signatory. Hereafter, we operate the XOR operation on an Aadhaar number, fingerprint, and a random Number (Rand). After that, a secure hash algorithm (SHA-256) is implemented on the XORed value to generate a 256-bit long hash string. This hashed string is used as a private key in the EdDSA algorithm and used as input in $EdDSA_{KeyGen}$ to generate the corresponding public key; therefore,

$$\mathrm{EdDSA_{priv}} = \mathrm{hashfunc}\left(F\mathrm{print}\oplus\mathrm{Anum}\oplus\mathrm{Rand}\right)\mathrm{EdDSA_{pub}} = \mathrm{EdDSA_{KeyGen}}\left(\mathrm{EdDSA_{priv}}\right). \tag{18}$$

After key generation, signatories must have to save the private keys securely, because the security of the whole system depends on the safety of the private key, and with this, the key generation process is also finished.

In a multisignature scheme, a single private ($EdDSA_{priv}$) key is required to generate the digital signature, and this master private key depends on the private keys of all signatories. In this process, signatories provide their respective private keys, then these keys are ANDed to generate a single private key that is used for a digital signature. This process can be expressed as

$$\text{andandEdDSApriv} = (\text{EdDSApriv1}\wedge\text{EdDSApriv2}\wedge\ldots\ldots). \tag{19}$$

Now, this master private-key $EdDSA_{priv}$ is used in Ed-the DSA algorithm to generate the digital signature for the given document, such as

$$\mathrm{EdDSA_{Sig}} = \mathrm{Sign}\left(\mathrm{Doc}, \mathrm{EdDSA_{Prive}}\right). \tag{20}$$

Forth coming, a digital signature is appended with the document and sent to the receiver, now receiver verifies the signature to confirm the authenticity and integrity of the received document. At the time of verification, a signed document *(Doc),* master public *($EdDSA_{pub}$)* key, and digital signature *($EdDSA_{Sig}$)* are required. Similar to the generation of a master private key, all public keys combine using AND operation to generate the single master public key. Now, this public key *(EdDSApub)*, signed document *(Doc),* and digital signature *(EdDSASig)* are used as input in the EdDSA signature algorithm for the verification process. This process can be expressed as

$$\begin{gathered}\text{andEdDSAPub} = (\text{EdDSAPub1}\wedge\text{EdDSAPub2}\wedge\ldots\ldots\ldots)\\ \text{EdDSAverif} = \text{Verify}(\text{Doc}, \text{EdDSAPup}, \text{EdDSASig}).\end{gathered} \tag{21}$$

Based on the verification, the system shows a message either "signature is verified" or "signature is not verified". Algorithms for three different modules, namely, key generation is in Algorithm 1, signing the document is in Algorithm 2, and verification of the signature is in Algorithm 3, are described.

From the above equations and algorithms, we can understand the internal process of the proposed scheme. when we are talking about the integrity of the message in our scheme. The Ed25519 signature algorithm is having the SHA-256 hashing algorithm by default that generates a hash of every message that signs by the user's private key of ED25519 for the message. That hash is calculated for the receiver by the ED25519 signature algorithm and verifies the integrity of the signature. The hashing scheme of ed25519 is the same as others, but the keys are not. These keys are used to sign and verify or authenticate the message very less compared to another digital signature algorithm. That reduces the time of all three processes of digital signature, i.e., key generation, signature generation, and verification. When the time is taken by our proposed scheme, then it is also cost-effective, as the process will complete in very less time compared then other digital signature algorithms.

4.1. Implementation of Proposed Model. We have implemented an offline tool for a digital signature-based proposed algorithm that runs on command-line interface (CLI). List of tools and packages used to implement the proposed scheme is given in Table 2.

All three different modules, namely, generation of key pairs, signing of the document, and verification of signature, have been implemented as follows:

Key generation: In the key pair generation process, the Python module is used to calculate the hash code of the imported credentials of users. In this process, the length of Aadhaar number is 12 digits long and the scanned fingerprint is used as an image. A Python module *random* is imported to generate random numbers that make our private key distinct and more secure. Now, we generate a 256-bit hash code from user credentials and a random number, for example, a 256-bit hashed code is represented as bed6d45b0aa9802cd31706afc9c090111 4ec80c72536f016330f11bc8a440339. This hash code or string is used as a private key to generate the master private key used in EdDSA digital signature module; furthermore, this private key is also used to generate a 256-bit long public key represented as: 5d8b8b5331dad29532b15-ce82105f727addcf23a0e 61e88124c49e5979565226. After

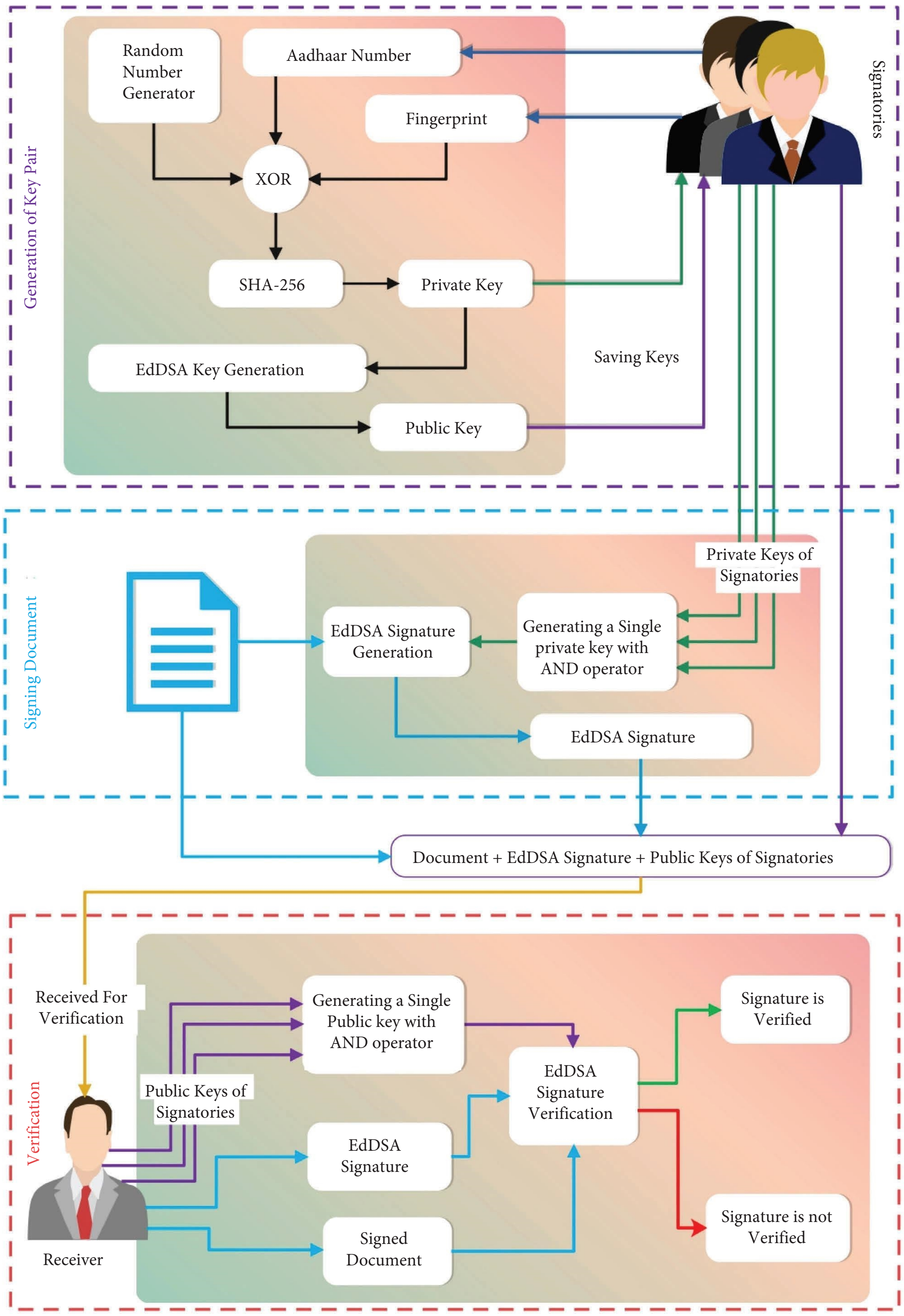
Generation of Key Pair
Random Number Generator
Aadhaar Number
Fingerprint
XOR
SHA-256
Private Key
EdDSA Key Generation
Public Key
Saving Keys
Signatories
Signing Document
Private Keys of Signatories
EdDSA Signature Generation
Generating a Single private key with AND operator
EdDSA Signature
Document + EdDSA Signature + Public Keys of Signatories
Verification
Received For Verification
Public Keys of Signatories
Generating a Single Public key with AND operator
EdDSA Signature
Signed Document
EdDSA Signature Verification
Signature is Verified
Signature is not Verified
Receiver

FIGURE 5: The architecture of the proposed identity-based multisignature scheme using EdDSA.

the completion of the key pair generation process, the private keys are kept secret by users. The key pair generation process is completed in approximately 0.011285 seconds.

Signing of document: To generate the cryptographic signature for a document, we use EdDSA digital signature module. In this process, a message id-test of 512-bit length

```
Input: Biometric Credential of users, Aadhaar Number, Random Integer.
Output: Public and Private Key pairs.
Notations;
$EdDSApriva \leftarrow$ EdDSA private-key
$EdDSApub \leftarrow$ EdDSA public-key
$hashfunc(.) \leftarrow$ SHA-256 hash function
Initialization;
$Fprint \leftarrow$ User fingerprint input
$Anum \leftarrow$ Aadhar Number
$Rand \leftarrow$ Random Number
Key Generation;
$EdDSA, priv = hashfunc(Fprint \oplus Anum \oplus Rand)$
$EdDSApub = EdDSAKeyGen\ (EdDSApriv)$
Print EdDSApriv and EdDSApub.
```

Algorithm 1: Key pair generation.

```
Input: Document, Private Keys of all signatories
Result: Digital Signature
Notations;
$EdDSA_{priv} \leftarrow$ EdDSA private-key
$EdDSA_{Prive} \leftarrow$ EdDSA combined private-key
$Doc \leftarrow$ File to be signed
$EdDSA_{Signature} \leftarrow$ EdDSA Signature
Initialization;
$EdDSA_{priv1}$ ▷Private key of user 1
$EdDSA_{priv2}$ ▷Private key of user 2.
.
.
$EdDSA_{privn}$ ▷Private key of user n.
$EdDSAPrive = EdDSA_{priv1} \wedge EdDSA_{priv2} \ldots \wedge EdDSA_{privn}$
Digital Signing;
$EdDSA_{Sign} = Sig\ (Doc, EdDSAPrive)$
Print $EdDSA_{Sig}$
```

Algorithm 2: Document signing.

```
Input: Signed Document, Digital Signature,
Public Keys of Signatories
Result: Verification Results
Notations;
$EdDSA_{pub} \leftarrow$ EdDSA public-key
$EdDSA_{Pub} \leftarrow$ EdDSA combined public-key
$Doc \leftarrow$ Signed document
$EdDSA_{Signature} \leftarrow$ EdDSA Signature
Initialization;
$EdDSA_{pub1}$ ▷Public key of user 1.
$EdDSA_{pub2}$ ▷Public key of user 2.
.
.
$EdDSA_{pubn}$ ▷Public key of user $n$.
$\mathrm{EdDSA_{Pub}} = \mathrm{EdDSA_{pub1}} \wedge \mathrm{EdDSA_{pun2}} \ldots\ldots \wedge \mathrm{EdDSA_{pubn}}$
Digital Signature Verification;
```

Algorithm 3: Continued.

```
$EdDSA_{verif}$= Verify (Doc, $EdDSA_{Pub}$, $EdDSA_{Sig}$)
if $EdDSA_{verif}$== 1 then
Signature verified Successfully;
else
Verification is Failed;
end
```

ALGORITHM 3: Signature verification.

is generated from the original message, and now, this message digest is encrypted using a private key that is derived from the private keys of individuals. This whole process takes approximately 0.012143 seconds. Finally, the original message, the digital signature, and a set of public keys of individuals are sent to the receiver.

Signature verification process: In the verification process, the public keys of signers are *ANDed* to generate a 256-bit long master public key. Now, the receiver generates 512-bit message digest from the original documents and decrypts the digital signature using the master public key. Finally, the receiver checks the message digest to the decrypted data blocks to ensure the message's integrity and authenticity. If the digital signature is validated, the system indicates that the message signature is validated! else shows the signature is failed to verify! The approximate time for the verification process is 0.016799 seconds. The proposed model is implemented using Python programming, and it is a platform-independent tool.

5. Security Analysis

In the proposed scheme, we are using the fingerprint and Aadhaar number of multisignatories for the key pair generation. As we know, biometric traits are difficult to be tampered and this scheme also uses a unique identity number like Aadhaar number that is verified by a government agency. Only authorized users can use their private keys for signing documents. Hence, the proposed scheme secures and provides the integrity and authenticity of the documents.

5.1. Key Guessing Attacks. A key guessing attack such as brute force attack in that a cryptanalyst tries all possibilities to guess the private key [37, 38]. In the proposed scheme, we are using SHA-256 hash function to generate 256-bit long private keys. Each private key depends on the user's credentials, namely, fingerprint and Aadhaar number along with a 256-bit random number. Therefore, the security of the proposed system is not feasible to break by the brute force attack in finite time

5.2. Collusion Attack. The collusion attack occurs when a specific user or a group of users intentionally has a secret agreement with an eve or a compromised the security policy. In this scheme, the private keys are kept secret by individuals; hence, no participant can steal the private key of the other participants.

5.3. Forgery or Key Replacement. In the implemented scheme, private keys are kept secret by individuals; hence, key replacement is not possible with private keys. However, most of the time, the attacker tries to replace or forge the values of a public key, attack is performed by a man-in-the-middle attack. In the proposed scheme, we are using EdDSA for digital signatures, and the version of EdDSA which we are using in the proposed scheme is unforgeable [39] by man-in-the-middle attack.

5.3.1. Security Analysis of EdDSA with Random Oracle Model. The random oracle model is used for security analysis, which works on the randomly chosen function. Hash functions are defined as key pairs for any digital signature algorithms or models that provide a random fixed length of hash when it processes a random amount of data. The newly generated hash function cannot be predictable for given random numbers. As the proposed system uses Ed25519, which is a curve point of ECC, we discuss the security analysis with the random oracle model as follows.

5.4. Tampering with Ed25519 Signature. The point $(X_1, y_1) \longrightarrow x_1 modp$ is independent of the value of y1 in the Ed25519 function. Two points present in the elliptic curve at the same x_1 coordinate and both points are opposite mean K can be replaced with $-Kmodm$, which would also follow the same $x1$ and hence the same r. This tempering process replaces s with $-Smodp$. Therefore, the (r, s) can be replaced by $(r, smodp)$ which can act as a real signature for the same message. However, using the hash function is preventing this tempering.

5.5. Reset Attack on Ed25519. Assume that the pseudorandom generator k is identifiable and the internal state can be reconfigurable. The scheme can be penetrable with two different message signatures. As if the signatory signs D1 by k and then resets it to generate k for $D2$, so the signatures we get $(r, s1)$ for $D1$ and $(r, s2)$ for $D2$. Therefore, we get that

$$x = \frac{s2SHA2(D1) - s1SHA2(D2)}{r(s2 - s1)} \text{ modp.} \tag{22}$$

5.6. Comparison of Security Analysis. Here, we present a comparison of security analysis with previous works in the following Table 3, in the context of the above discussed points.

Table 2: List of tools and packages used to implement the proposed scheme.

Component	Description
Processor	I3 5th gen intel processor
RAM	12 GB
Operating system	Ubuntu 20.10
Programming language	Python 3.7
IDE	VS code
Component	Description

Table 3: Comparison of Security analysis with previously implemented schemes.

Security property	Bisheh N. et al. [34]	Brendel et al. [16]	Großschädl et al. [22]	Proposed
Key guessing attacks	N	N	N	Y
Collusion attack	N	N	N	Y
Forgery or key replacement	N	Y	N	Y
Tampering with Ed25519 signature	N	Y	N	Y
Reset attack on Ed25519	N	Y	N	Y

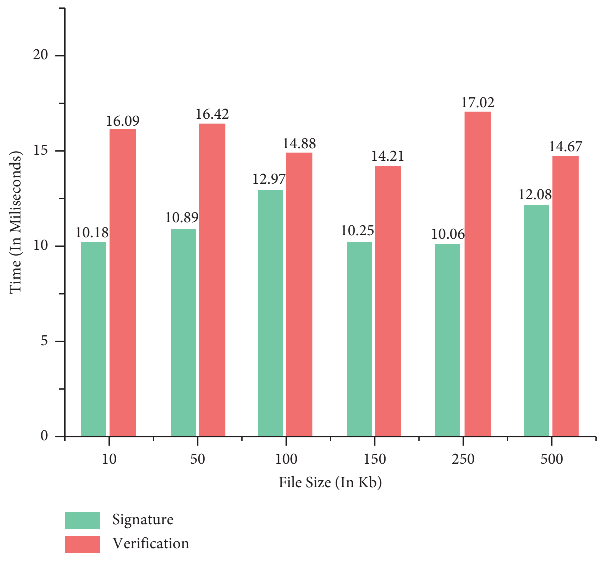

Figure 6: Bar-chart of time comparison for signature generation and verification for two signers.

6. Results and Discussion

The implementation has been tested for two, five, and ten signatories to compare the time of signature generation and its verification with different file sizes. Figures 6–8 are representing the results.

Table 4 compares the suggested plans with the schemes that have already been put into place. The execution reveals that the length of the private key and the public key is 256 bits, and the processing time for key pair creation is around 11 milliseconds. Similarly, the processing time for the signing and verification process is approximately 12 milliseconds and 16 milliseconds successively. From Figure 9, one can easily see that the key length of EdDSA is small in comparison with the RSA for the same security level; therefore, one can say EdDSA is less space-complex than RSA. The time complexity for signing and verification is less for EdDSA comparison to RSA [7]. The EdDSA also possesses the property of batch verification; therefore, it is useful for multisignatures schemes [39]. Therefore, the implemented scheme is fast and more secure compared to previously implemented schemes.

6.1. Limitations of the Work. As in the previous section, we can understand that the proposed work is better than the previously presented scheme. However, every scheme has their constraints

When the number of signatories is rising, the time complexity of the algorithm is increasing. This happens because of the many numbers of public and private keys that will be used. This is also a slow process of signing and verification time.

This scheme is not only a test in an offline environment as it is designed as an offline application but can be tested online also by developing as an online application with Python web development tools.

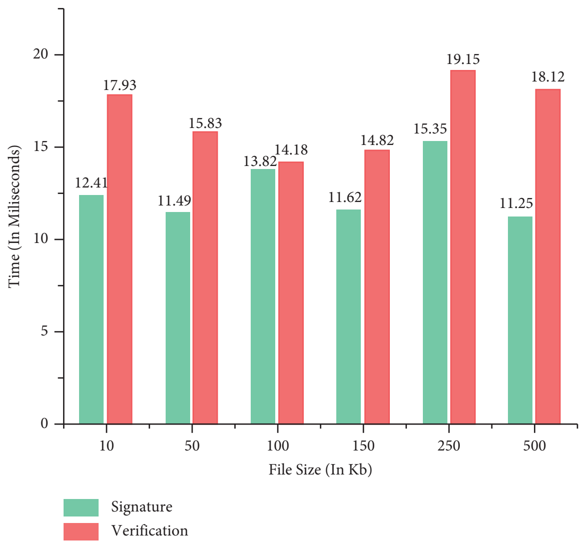

Figure 7: Bar-chart of time comparison for signature generation and verification for five signers.

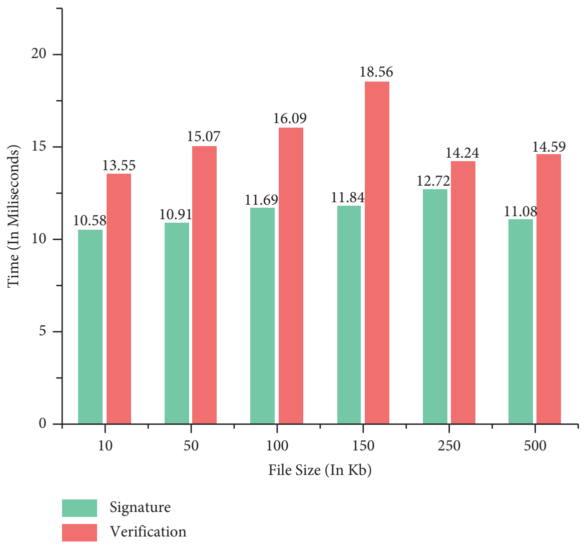

Figure 8: Bar-chart of time comparison for signature generation and verification for ten signers.

Table 4: Comparative Overview of the proposed scheme and previously implemented schemes.

Scheme	Credentials for key generation	Based on	Security analysis
Ahmed et al. [25]	Face recognition	RSA	No
Rahmawati et al. [3]	Fingerprint	RSA	No
Bellare and Neven [18]	Unavailable	RSA	Yes
Tanwar and Kumar [1]	E-mail and Aadhaar number	RSA	Yes
Prathapkumar et al. [40]	Unavailable	Double RSA	No
Nagashima et al. [33]	Creator's content and ID	ECC	No
Tanwar et al. [23]	ID card	ECDSA	Yes
Diemert et al. [24]	Unavailable	ECDSA	Yes
Bisheh N. et al. [34]	Unavailable	EdDSA	Yes
Proposed scheme	Fingerprint, Aadhaar number, and random number	EdDSA	Yes

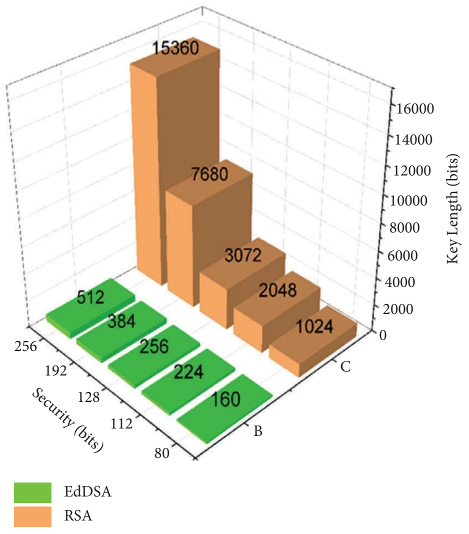

Figure 9: Bar-chart of time comparison for signature generation and verification for ten signers.

The scheme is not a test in a real-time environment. It can be tested in the real-time environment with some specific adjustments such as using hardware to collect the fingerprint images in real-time.

The scheme is designed to test documents only, specifically PDF files, but can be a test for other types of files also and can identify the complexity of different files.

7. Conclusion

We have evaluated multisignature systems that have previously been proposed as well as identity-based multisign systems (IBMS). We have suggested our improven identity-based multisignature technique utilizing EdDSA for electronically signing documents based on the findings of this investigation. In this scheme, we have used the Ed25519 algorithm that is based on the ECC. This algorithm is more secure and faster in compare to RSthe A algorithm. In the implemented scheme, an Aadhar number, fingerprint, and a random number are used to generate the key pairs. In this study, we have provided a detailed description of key-pair generation, signing documents, and verification of digital signatures. We further also provide security analysis based on the properties of our proposed scheme, which suggests that the proposed scheme is more secure and fast compared to previously proposed schemes. In the future, it will be interesting to extend this scheme to authenticate social messaging and verify the integrity and authenticity of big data. Furthermore, the proposed scheme can also be used to authenticate multimedia files [40].

Data Availability

The data that used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Tanwar and A. Kumar, “An efficient and secure identity based multiple signatures scheme based on RSA,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 6, pp. 953–971, 2019.

[2] N. J. G. Saho and E. C. Ezin, “Securing document by digital signature through RSA and elliptic curve cryptosystems,” in *Proceedings of the 2019 International Conference on Smart Applications, Communications, and Networking (SmartNets)*, pp. 1–6, IEEE, Sharm El Sheik, Egypt, December 2019.

[3] E. Rahmawati, M. Listyasari, A. S. Aziz et al., “Digital signature on file using biometric fingerprint with a fingerprint sensor on a smartphone,” in *Proceedings of the 2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, pp. 234–238, IEEE, Surabaya, Indonesia, September 2017.

[4] S. A. El-Rahman, D. Aldawsari, M. Aldosari, O. Al- rashed, and G. Alsubaie, “A secure cloud-based digital signature application for iot,” *International Journal of E-Services and Mobile Applications*, vol. 10, no. 3, pp. 42–60, 2018.

[5] T. Jager, J. Schwenk, and J. Somorovsky, “On the security of tls 1.3 and quick against weaknesses in pkcs# 1 v1. 5 encryption,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1185–1196, Denver, Colorado, October 2015.

[6] A. A. Imem, “Comparison and evaluation of digital signature schemes employed in ndn network,” 2015, https://arxiv.org/abs/1508.00184.

[7] C. Li, Y. Wu, and F. Yu, “An Improved Schnorr-Based Multi-Signature Scheme with Blockchain Applicatio,” in *Proceedings of the IEEE 3rd International Conference on Civil Aviation Safety and Information Technology*, pp. 858–863, ICCASIT), Changsha, China, September 2021.

[8] S. Kazmirchuk, A. Ilyenko, S. Ilyenko, O. Prokopenko, and Y. Mazur, “The improvement of digital signature algorithm based on elliptic curve cryptography,” *Advances in Computer Science for Engineering and Education III ICCSEEA 2020*, vol. 1247, 2021.

[9] R. Kaur and A. Kaur, “Digital signature,” in *Proceedings of the 2012 International Conference on Computing Sciences*, pp. 295–301, IEEE, Washington, DC, USA, September 2012.

[10] R. C. Merkle, "A certified digital signature," *Conference on the Theory and Application of Cryptology*, Springer, Heidelberg, Germany.

[11] F. O. Mojisola, S. Misra, C. Falayi Febisola, O. Abayomi-Alli, and G. Sengul, "An improved random bit-stuffing technique with a modified RSA algorithm for resisting attacks in information security (rbmrsa)," *Egyptian Informatics Journal*, vol. 23, no. 2, pp. 291–301, 2022.

[12] S. C. Gupta and M. Sanghi, "Matrix modification of RSA digital signature scheme," *Journal of Applied Security Research*, vol. 16, no. 1, pp. 63–70, 2021.

[13] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[14] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[15] S. Josefsson and I. Liusvaara, "Edwards-curve digital signature algorithm (eddsa)," *Internet Research Task Force*, vol. 8032, pp. 257–260, 2017.

[16] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of ed25519: theory and practice," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1659–1676, IEEE, Francisco, CA, USA, May 2021.

[17] B. Preneel, "Cryptographic hash functions," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 431–448, 2010.

[18] M. Bellare and G. Neven, "Identity-based multi-signatures from RSA," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 145–162, Springer, Francisco, CA, USA, February 2007.

[19] N. Storublevtcev, "Cryptography in blockchain," in *Computational Science and its Applications – ICCSA 2019*, S. Misra, O. Gervasi, B. Murgante et al., Eds., pp. 495–508, Springer International Publishing, Heidelberg, Germany, 2019.

[20] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ecdsa with fast trustless setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1179–1194.

[21] N. K. Dewangan and P. Chandrakar, "Enhanced privacy and security of voters' identity in an inter-planetary file system-based e-voting process," in *Blockchain for Information Security and Privacy*, pp. 113–132, Auerbach Publications, Boca Raton, FL, USA, 2021.

[22] J. Großschädl, C. Franck, and Z. Liu, "Lightweight eddsa signature verification for the ultra-low-power internet of things," in *International Conference on Information Security Practice and Experience*, pp. 263–282, Springer, Heidelberg, Germany, 2021.

[23] S. Tanwar, S. Badotra, M. Gupta, and A. Rana, "Efficient and secure multiple digital signature to prevent forgery based on ECC," *International Journal of Applied Science & Engineering*, vol. 18, no. 5, pp. 1–7, 2021.

[24] D. Diemert, K. Gellert, T. Jager, and L. Lyu, "More efficient digital signatures with tight multi-user security," in *Proceedings of the IACR International Conference on Public-Key Cryptography*, Springer, Edinburgh, UK, May 2021.

[25] A. Ahmed, T. Hasan, F. A. Abdullatif, and M. S. M. Rahim, "A digital signature system based on real-time face recognition," in *Proceedings of the 2019 IEEE 9th International Conference on System Engineering and Technology*, pp. 298–302, Shah Alam, Malaysia, October 2019.

[26] D. M. Tuan, "Msms: a multi-section multi-signature model with distinguished signing responsibilities," in *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, pp. 131–135, Yogyakarta, Indonesia, May 2017.

[27] Y. Zhou, Z. Li, F. Hu, and F. Li, "Identity-based combined public key schemes for signature, encryption, and encryption," in *Information Technology and Applied Mathematics*Vol. 3–22, Springer, Heidelberg, Germany, 2019.

[28] H. Jin, H. Debian, and C. Jianhua, "An identity-based digital signature from ecdsa," *Second International Workshop on Education Technology and Computer Science*, IEEE, vol. 1, pp. 627–630 2010.

[29] M. K. Chande, C. C. Lee, and T. Y. Chen, "An improved multi-signature scheme for specified group of verifiers," *International Journal of Electronic Security and Digital Forensics*, vol. 9, no. 2, pp. 180–190, 2017.

[30] L. Wei, J. Ai, and S. Liu, "A tightly secure multi-party-signature protocol in the plain model," in *Proceedings of the 2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*, pp. 672–677, IEEE, Shenyang, China, October 2015.

[31] K. Ramkumar and M. Juneja, "Multi-signature authentication and key management system to ensure reliable paths for payload delivery," in *Proceedings of the 2020 Indo– Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN*, pp. 194–201, IEEE, Rajpura, India, February 2020.

[32] G. Lax, F. Buccafurri, and G. Caminiti, "Digital document signing: vulnerabilities and solutions," *Information Security Journal: A Global Perspective*, vol. 24, no. 1-3, pp. 1–14, 2015.

[33] N. Nagashima, M. Inamura, and K. Iwamura, "Implementation of secondary available digital content protection schemes using identity-based signatures," in *Proceedings of the 7th International Conference on Information Systems Security and Privacy, ICISSP 2021*, pp. 485–491.

[34] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Cryptographic accelerators for digital signature based on ed25519," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 29, no. 7, pp. 1297–1305, 2021.

[35] S. Dargan and M. Kumar, "A comprehensive sur- vey on the biometric recognition systems based on physiological and behavioral modalities," *Expert Systems with Applications*, vol. 143, Article ID 113114, 2020.

[36] C. Ambrose, J. W. Bos, B. Fay, M. Joye, M. Lochter, and B. Murray, "Differential attacks on determine- istic signatures," in *Cryptographers' Track at the RSA Conference*, pp. 339–353, Springer, Heidelberg, Germany, 2018.

[37] N. Samwel, L. Batina, G. Bertoni, J. Daemen, and R. Susella, "Breaking ed25519 in wolfssl," in *Crypt- Geographers Track at the RSA Conference*, pp. 1–20, Springer, Heidelberg, Germany, 2018.

[38] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[39] B. Harsha, A. Damodaran, S. Ranganath, V. Raut, and S. Holla, "An approach to enable secure and reliable communication on iot devices," *International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, IEEE, , vol. 4, pp. 1–6, 2019.

[40] K. Prathapkumar and A. T. Raja, "Double signa- ture based cryptography using ds-sha256 in cloud computing," *NVEO-NATURAL VOLATILES & ES- ESSENTIAL OILS Journal| NVEO*, vol. 8, no. 5, pp. 9535–9541, 2021.