Research Article

# Biocrypt: An Advanced Dna-Based Cryptographic Security Framework

Karthikeyan VV[1], Gowtham S[2], Gowtham P[3], Sathiya V[4], Jackulin C[5]

[1]Department of Computer Science Panimalar Engineering College Chennai, India.Email: karthikeyan56v@gmail.com
[2]Department of Computer Science Panimalar Engineering College Chennai, India. Email: sgowtham28122004@gmail.com
[3]Department of Computer Science Panimalar Engineering College Chennai, India. Email: gowtham12b222@gmail.com
[4]Department of Computer Science Panimalar Engineering College Chennai, India. Email: deviviji2000@yahoo.co.in
[5]Department of Computer Science Panimalar Engineering College Chennai, India. Email: cjackulin@panimalar.ac.in

**Abstract**

As genomic data volumes continue to grow rapidly, protecting genetic information throughout its lifecycle—from collection and storage through analysis and sharing—has become a critical challenge. This paper presents BIOCRYPT, a cryptographic framework implementing the DNASSF protocol for secure storage and transport of DNA and RNA sequence data. The system provides multiple security layers: AES-256-GCM for authenticated encryption, X25519 Elliptic Curve Cryptography for key exchange, HKDF-SHA256 for secure key derivation, and age-based key wrapping for device-bound key management. A 2-bit DNA encoding scheme reduces storage requirements by 75% while preserving cryptographic security. The system automatically detects FASTA, FASTQ, and GenBank formats, and converts RNA sequences to DNA for standardized storage. Encrypted workspaces allow multiple data sequences to be stored within a single protected container, supporting collaborative research while maintaining access control. Performance evaluation demonstrates encryption throughput of approximately 4 MB/s with negligible impact on genomic analysis workflows. Implemented in Rust for memory safety, BIOCRYPT provides a robust and efficient solution for protecting genomic data in research, clinical, and biotechnology settings.

**Keywords:** DNA encryption, genomic security, bioinformatics, AES-256-GCM, authenticated encryption, X25519, elliptic curve cryptography, key management, DNASSF protocol.

## Introduction

Advances in high-throughput sequencing technology have produced an unprecedented volume of genomic data, currently estimated to exceed forty exabytes annually and projected to grow at an accelerating rate as sequencing costs continue to decline [2]. This growth imposes correspondingly stringent requirements for protecting the confidentiality, integrity, and availability of genetic information at every stage of its lifecycle.

Genomic data present a uniquely challenging privacy problem. Unlike conventional personal data, a genetic sequence is immutable: once disclosed, it cannot be revoked or altered. An individual's genome serves as a stable biological identifier with implications that extend beyond the individual to biological relatives who share overlapping genetic backgrounds [1]. Genetic sequences can reveal susceptibility to disease, ancestral origin, and other sensitive characteristics that individuals have a legitimate interest in controlling. In addition, genomic data carry substantial commercial value for pharmaceutical research, personalized medicine, and biotechnology, making them an attractive target for industrial espionage and intellectual property theft [3].

Standard file-level encryption provides basic confidentiality through symmetric or asymmetric cryptographic algorithms, but it does not address the structural properties, encoding requirements, and operational constraints that are specific to DNA sequence data. Genomic file formats carry domain-specific metadata and impose strict structural invariants that must be preserved across encryption and decryption. Furthermore, collaborative genomic research, which routinely involves geographically distributed participants from multiple institutions and jurisdictions, requires a principled key management approach that ensures authorized access while preventing unauthorized disclosure.

The BIOCRYPT framework addresses these requirements by introducing the DNASSF (DNA Secure Storage Format) protocol, version 1. BIOCRYPT integrates best practices from applied cryptography, genomic data representation, and the practical security requirements of bioinformatics pipelines. The specific contributions of this work are as follows.

1. A self-describing binary file format that stores encrypted DNA sequences compactly while preserving the integrity of associated metadata, enabling efficient storage and retrieval of encrypted genomic data.
2. A 2-bit nucleotide encoding scheme that reduces pre-encryption storage requirements by 75%, improving both storage efficiency and encryption throughput.
3. An integrated security design combining AES-256-GCM authenticated encryption with X25519 elliptic-curve key exchange, providing strong confidentiality and integrity guarantees.

4. A device-specific key wrapping scheme based on the Age encryption protocol that enables secure cross-organizational key distribution.
5. Automatic detection and processing of multiple genomic sequence formats and text encodings, minimizing operator burden and ensuring correct handling of heterogeneous input files.

The remainder of this paper is organized as follows. Section 2 reviews the theoretical foundations of genomic data formats and the cryptographic primitives employed. Section 3 describes the system architecture and the rationale behind key design decisions. Section 4 presents the mathematical formulation of the encryption pipeline. Section 5 covers the key management framework. Section 6 specifies the DNASSF binary file format. Section 7 reports the implementation details and performance results. Section 8 provides a security analysis and threat model. Section 9 describes the encrypted workspace management system. Section 10 outlines practical deployment scenarios. Section 11 discusses directions for future work, and Section 12 concludes the paper.

**Background And Theoretical Foundations**

This section establishes the theoretical background required to understand the BIOCRYPT cryptographic system, covering the genomic data formats that require protection and the cryptographic primitives that provide the stated security guarantees.

**Genomic Data Formats and Representations**

DNA sequence data are conventionally stored in standardized file formats developed and refined by the bioinformatics community over several decades. These formats were designed to balance human readability with processing efficiency; however, they predate the period in which security was treated as a primary design concern. The DNASSF framework addresses three formats that collectively account for the majority of genomic data currently in storage and circulation.

**FASTA.** The FASTA format was originally developed for sequence alignment and stores nucleotide sequences as plain text. Each sequence record begins with a header line prefixed by the greater-than symbol (>), followed by one or more lines of sequence data. A FASTA dataset may be represented formally as a collection of ordered pairs in which each header is associated with its corresponding nucleotide string:

$$F_{FASTA} = \{(h_i, s_i) \mid h_i \in \Sigma^*, s_i \in \{A, C, G, T, N\}^*\}^n_{i=1} \quad (1)$$

where $h_i$ is the free-form header string for the i-th sequence, $s_i$ is the nucleotide sequence, $\Sigma^*$ denotes the Kleene closure over printable ASCII characters, and n is the total number of records.

**FASTQ.** The FASTQ format extends FASTA by appending a per-base quality score to each record, enabling downstream analyses to account for sequencing error rates. Each FASTQ record comprises four lines: a header, the nucleotide sequence, a separator, and a quality string whose length matches that of the sequence. Quality scores are expressed on the Phred scale and formally defined as:

$$F_{FASTQ} = \{(h_i, s_i, q_i) \mid q_i \in [0, 93]^{|s_i|}\}^n_{i=1} \quad (2)$$

where $q_i$ denotes the vector of Phred quality scores and $q_{i,j}$ is the quality score for nucleotide $s_{i,j}$. The Phred quality score is defined by the logarithmic transformation:

$$Q = -10 \log_{10}(P_e) \quad (3)$$

where Q is the Phred score and $P_e$ is the probability of an incorrect base call.

**GenBank.** The GenBank flat file format, maintained by the National Center for Biotechnology Information (NCBI), encapsulates a biological sequence together with comprehensive annotations including feature tables, literature citations, and organism metadata. The complexity of these flat files requires careful parsing to preserve structural integrity across cryptographic transformations.

**Comparison of Existing Solutions**

Table 1 summarizes BIOCRYPT against representative commercial and academic genomic encryption solutions across five dimensions: supported format, encryption cipher, throughput, open-source availability, and support for local on-premises deployment.

**Table 1: Comparison of Genomic Encryption Solutions**

| Solution | Format | Cipher | Speed | Open | Local |
|---|---|---|---|---|---|
| DNAnexus | FASTQ | AES-256 | Med | × | × |
| Seven Bridges | BAM | AES-256 | Med | × | × |
| GPG | Any | AES/RSA | Low | ✓ | ✓ |
| CryptGenome [7] | FASTA | ChaCha20 | High | ✓ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| SecureGenome [1] | BAM | AES-CTR | Med | × | ✓ |
| **DNASSF (Proposed)** | **DNASSF** | **AES-GCM** | **High** | ✓ | ✓ |

**RNA Sequence Considerations**

RNA sequences share the four-letter nucleotide alphabet of DNA with one exception: thymine (T) is replaced by uracil (U). BIOCRYPT detects the presence of uracil to identify RNA input and applies an in-place substitution that maps the sequence into the DNA alphabet, enabling a single unified storage representation. The forward transformation is:

$$\tau_{RNA \to DNA} : U \mapsto T,\ x \mapsto x \text{ for } x \in \{A, C, G\} \quad (4)$$

The inverse transformation permits reconstruction of the original RNA sequence on demand:

$$\tau_{DNA \to RNA} : T \mapsto U,\ x \mapsto x \text{ for } x \in \{A, C, G\} \quad (5)$$

**Cryptographic Primitives**

The BIOCRYPT framework is built upon well-studied cryptographic primitives that provide strong, formally analyzed security guarantees.

**AES-256-GCM** is an Authenticated Encryption with Associated Data (AEAD) scheme that combines AES in Galois/Counter Mode with a Galois-field authentication tag. It simultaneously provides confidentiality through stream encryption and integrity through polynomial hashing, so any modification to the ciphertext or to the unencrypted associated data is detected before decryption proceeds. The encryption function accepts a 256-bit key K, a 96-bit nonce N, plaintext P, and optional associated data A, and produces ciphertext C together with a 128-bit authentication tag T:

$$Enc_{GCM}(K, N, P, A) \to (C, T) \quad (6)$$

Decryption verifies the authentication tag before releasing the plaintext:

$$Dec_{GCM}(K, N, C, A, T) \to P \text{ if tag valid, } \perp \text{ otherwise} \quad (7)$$

where $\perp$ denotes authentication failure and the ciphertext is unconditionally rejected.

**X25519** is a Diffie–Hellman key agreement function defined over Curve25519, a Montgomery-form elliptic curve with favorable security and performance properties [11]. The function is defined as:

$$X25519(k, u) = x([k]u) \quad (8)$$

where $k \in \mathbb{Z}^{2255-19}$ is the scalar private key, u is the u-coordinate of the input point, [k]u denotes scalar multiplication, and x(·) extracts the x-coordinate of the resulting point. Security rests on the computational Diffie–Hellman assumption for Curve25519.

**HKDF-SHA256 [27]** is an HMAC-based extract-and-expand key derivation function that transforms input keying material of potentially limited quality into uniformly distributed output key material of the required length. The extraction phase concentrates entropy:

$$PRK = HMAC\text{-}SHA256(salt, IKM) \quad (9)$$

The expansion phase produces output blocks iteratively:

$$T_i = HMAC\text{-}SHA256(PRK, T_{i-1} \| info \| i) \quad (10)$$

where $T_0$ is the empty string and the output key material (OKM) is formed by concatenating successive $T_i$ values until the required output length L is reached:

$$HKDF(IKM, salt, info, L) \to OKM \quad (11)$$

**System Architecture**

The DNASSF system is organized as a collection of loosely coupled modules to support extensibility, enforce security boundaries between components, and enable independent testing and auditing of each subsystem. The design separates user-facing application interfaces, cryptographic services, and persistent storage into distinct layers so that each layer can be developed and validated independently.

**Figure 1: BIOCRYPT system architecture showing modular component organization across application, cryptographic, and storage layers.**

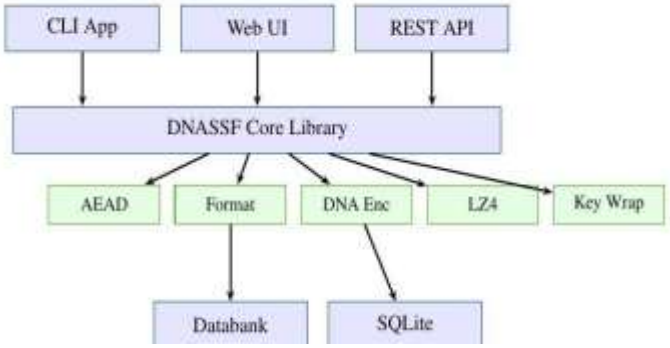


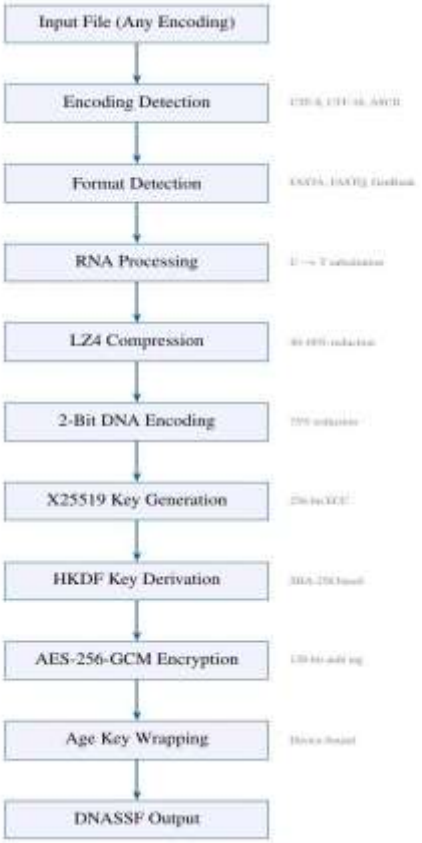

Figure 1 illustrates the layered component structure. At the application layer, three interface types provide access to the cryptographic core: a command-line interface (CLI) for scripted and automated bioinformatics workflows, a web-based graphical interface for interactive use, and a REST API for programmatic integration with laboratory information management systems and analysis pipelines.

The core library comprises five principal modules. The AEAD module applies AES-256-GCM with a freshly generated cryptographic nonce for each encryption operation, preventing nonce-reuse vulnerabilities. The format module serializes and deserializes the DNASSF binary container, embedding versioning metadata to ensure forward compatibility. The DNA encoder implements the 2-bit nucleotide encoding scheme. The compression module applies LZ4 [13] prior to encryption to exploit repetitive structure common in genomic sequences. The key wrapping module employs the Age protocol [8] to encrypt session keys under recipient public keys, enabling secure inter-device key exchange.

**Cryptographic Encryption Pipeline**

The encryption pipeline is a deterministic sequence of data transformations that converts raw genomic data into an encrypted, integrity-protected output while allowing authorized parties to recover the original sequence without loss. Each stage applies a defined operation whose output becomes the input to the subsequent stage.

**Figure 2: Complete encryption pipeline showing all transformation stages and their associated parameters.**



**Encryption Algorithm**

Algorithm 1 presents the ordered processing steps from raw input to the final DNASSF binary container.

**Algorithm 1 DNASSF Secure Encryption Process**

**Require:** Genomic sequence file $D_{in}$, recipient public key $pk_{rec}$ (optional)

**Ensure:** Encrypted DNASSF file $E_{out}$

Enc ← DetectEncoding($D_{in}$)
Fmt ← DetectFormat($D_{in}$)
if Enc ≠ UTF-8 then
$D_{proc}$ ← ConvertToUTF8($D_{in}$)
else
$D_{proc}$ ← $D_{in}$
end if
if Fmt = RNA then
$D_{proc}$ ← ReplaceUwithT($D_{proc}$)
end if
$D_{comp}$ ← ApplyLZ4($D_{proc}$) {Optional compression}
$D_{bin}$ ← Encode2Bit($D_{comp}$) {Optional 2-bit DNA encoding}
(sk, pk) ← GenerateX25519KeyPair()
K ← HKDF-SHA256(X25519(sk, G), salt)
N ← GenerateRandomNonce(96 bits)
(C, T) ← AES-256-GCM-Encrypt(K, N, $D_{bin}$)
if $pk_{rec}$ is provided then
$K_{wrap}$ ← AgeWrap(sk, $pk_{rec}$)
end if
return ConstructDNASSF(Header(pk, N, Flags), C, T, $K_{wrap}$)

**Mathematical Formulation of Encryption**

The complete encryption transformation may be expressed as a composition of functions applied sequentially to the input data. Let D denote the raw genomic input. The final encrypted output E is:

$$E = E_{complete}(D)$$
$$= Age_{pkr}(AES\text{-}GCM_K(Compress(Encode_{2\text{-}bit}(\tau(D))))) \quad (12)$$

where τ is the RNA-to-DNA conversion, $Encode_{2\text{-}bit}$ applies 2-bit nucleotide encoding, Compress applies LZ4, $AES\text{-}GCM_K$ performs authenticated encryption under key K, and $Age_{pkr}$ wraps the session key under the recipient's public key $pk_r$.

The AES-GCM encryption key is derived from the X25519 ephemeral shared secret via HKDF:

$$K = HKDF\text{-}SHA256(X25519(sk, G), salt, \text{“dnassf-aes-key”}, 32) \quad (13)$$

where sk is a freshly generated ephemeral private key, G is the Curve25519 base point, salt provides additional entropy, and the context string “dnassf-aes-key” provides domain separation.

**Two-Bit DNA Encoding Scheme**

The 2-bit DNA encoding scheme exploits the fact that the standard DNA alphabet contains exactly four nucleotide symbols, each of which can therefore be represented in exactly two bits—the theoretical minimum. The encoding function ϕ assigns each nucleotide a unique 2-bit binary value:

$$\phi(A) = 00_2 = 0 \quad (14)$$
$$\phi(C) = 01_2 = 1 \quad (15)$$
$$\phi(G) = 10_2 = 2 \quad (16)$$
$$\phi(T) = 11_2 = 3 \quad (17)$$

For a nucleotide sequence $S = s_1 s_2 \cdots s_n$ of length n, four nucleotides are packed into each output byte:

$$\Phi(S) = \Sigma^{n}_{i=1}\, \phi(s_i) \cdot 4^{n-i} \quad (18)$$

Concretely, the j-th output byte is computed as:

$$B_j = 64\phi(s_{4j-3}) + 16\phi(s_{4j-2}) + 4\phi(s_{4j-1}) + \phi(s_{4j}) \quad (19)$$

The resulting storage efficiency gain is:

$\eta_{encoding} = 1 - Size_{2\text{-}bit} / Size_{ASCII} = 1 - 2n/8n = 1 - 1/4 = 0.75 = 75\%$
(20)
This 75% reduction in data volume lowers both storage requirements and the computational cost of subsequent encryption.

**Compression Analysis**
LZ4 compression exploits the repetitive structure prevalent in genomic sequences arising from tandem repeats, transposable elements, and segmental duplications. The compression ratio ρ is defined as:

$$\rho = |D_{original}| / |D_{compressed}| \quad (21)$$

Empirical evaluation on representative genomic datasets yields compression ratios in the range:

$$\rho_{DNA} \in [1.5, 3.0] \quad (22)$$

Applied in combination, 2-bit encoding and LZ4 compression yield compounded reduction. For a 1 MB input file the approximate final size is:

$$Size_{final} \approx 1.0 \times 0.25 \times 0.4 = 0.1 \text{ MB} \quad (23)$$

corresponding to a 90% reduction that substantially decreases both storage requirements and encryption latency.

**Key Management Framework**
Robust key management is a prerequisite for any cryptographic system deployed in production genomic workflows. Whereas theoretical protocol analyses typically assume ideal key distribution conditions, practical deployments must address key generation, secure storage, controlled distribution to authorized parties, and eventual revocation or rotation. BIOCRYPT adopts a hierarchical key management architecture that supports per-user local encryption and provides a secure mechanism for cross-organizational key exchange.

**Key Generation and Security Properties**
X25519 key pairs are generated by sampling a private key uniformly at random from the key space:

$$sk \leftarrow^{\$} \{0, 1\}^{256} \quad (24)$$

The corresponding public key is obtained by scalar multiplication of the Curve25519 base point by the private scalar:

$$pk = X25519(sk, 9) \quad (25)$$

where 9 is the u-coordinate of the Curve25519 base point in Montgomery form. The security of the key pair rests on the computational intractability of the elliptic curve discrete logarithm problem: for any probabilistic polynomial-time adversary A,

$$Pr[A(pk) = sk] \leq negl(\lambda) \quad (26)$$

where λ is the security parameter and negl(λ) denotes a negligible function.

**Per-File Key Isolation**
A fundamental security property of DNASSF is per-file key isolation: each encrypted file is protected by an independently generated session key, ensuring that the compromise of one key does not affect the confidentiality of any other file. Formally:

$$\forall f_i, f_j \in F : i \neq j \Rightarrow sk_i \neq sk_j \quad (27)$$

where F is the set of all encrypted files. This isolation implies zero mutual information between file contents and cross-file keys:

$$I(D_i; sk_j) = 0, \ \forall i \neq j \quad (28)$$

guaranteeing that knowledge of one file's session key yields no information about any other file's plaintext.

**HKDF Key Derivation Process**
HKDF-SHA256 derives the AES-GCM session key from the X25519 shared secret. This derivation step ensures that the encryption key exhibits full 256-bit entropy even when the raw shared secret has known structure, binds the derived key to its intended purpose through the info parameter, and supports derivation of multiple independent keys from the same shared secret when required.

During extraction, a pseudorandom key is derived:

$$PRK = HMAC\text{-}SHA256(salt, shared\_secret) \quad (29)$$

During expansion, the session key is produced:

$$K_{enc} = HMAC\text{-}SHA256(PRK, info \| 0x01) \quad (30)$$

The info parameter is set to "dnassf-encryption-key-v1", providing domain separation from other applications that may derive keys from the same shared secret.

**Age Key Wrapping Protocol**

The Age encryption protocol [8] provides a straightforward mechanism for securely sharing session keys across devices and organizations. To distribute a DNASSF session key, the sender encrypts it under the recipient's Age public key, producing a wrapped key file that may be transmitted over an untrusted channel:

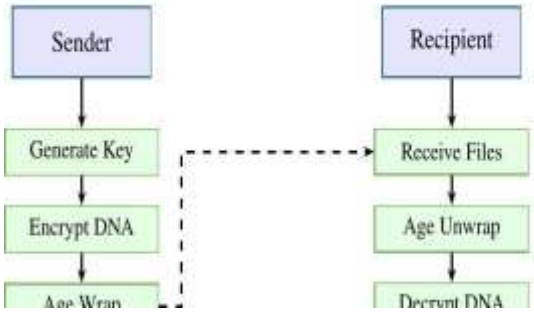
$$\text{wrapped\_key} = \text{Age}_{\text{pk}_{\text{recipient}}}(\text{sk}_{\text{DNASSF}}) \quad (31)$$

The recipient recovers the session key using the corresponding private key:

$$\text{sk}_{\text{DNASSF}} = \text{Age}_{\text{sk}_{\text{recipient}}}^{-1}(\text{wrapped\_key}) \quad (32)$$

Figure 3 illustrates the protocol. The sender generates the session key, encrypts the genomic data, and wraps the key under the recipient's public key. Both the encrypted data file and the wrapped key file may be transmitted over any channel without exposing the underlying key material. The recipient unwraps the key and decrypts the genomic data.

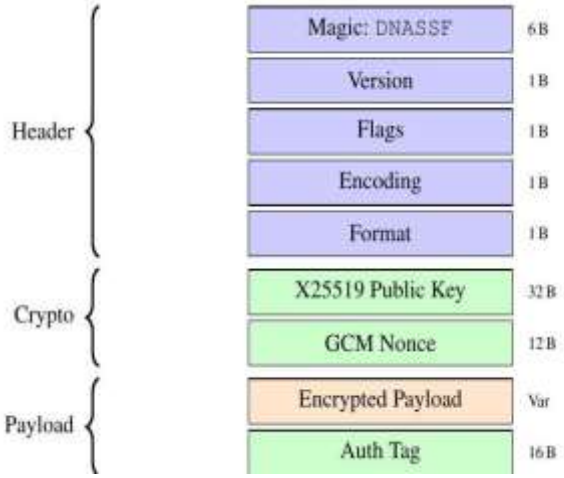
**Figure 3: Age key wrapping protocol for secure cross-organizational session key distribution.**




**Dnassf File Format Specification**

The DNASSF file format defines a self-describing binary structure for storing encrypted genomic data in a compact, consistent, and forward-compatible manner. The format is organized into three regions: a fixed-length header carrying format metadata, a cryptographic parameters block, and a variable-length encrypted payload followed by the authentication tag.

**Figure 4: DNASSF binary file format structure with field sizes.**




The single-byte flags field encodes the processing options applied during encryption, enabling the decryption routine to invert exactly the transformations that were applied:

$$\text{Flags} = (b_7, b_6, b_5, b_4, b_3, b_{\text{RNA}}, b_{2\text{-bit}}, b_{\text{compress}})_2 \quad (33)$$

where $b_{\text{compress}}$ indicates LZ4 compression was applied, $b_{2\text{-bit}}$ indicates 2-bit DNA encoding was applied, $b_{\text{RNA}}$ indicates the original sequence was RNA and uracil was converted to thymine, and bits $b_3$ through $b_7$ are reserved for future use. Table 2 lists the header fields with their byte offsets and sizes.

**Table 2: DNASSF Header Field Definitions**

| Field | Offset | Size | Description |
|---|---|---|---|
| Magic | 0 | 6 | Format identifier string (DNASSF) |

| | | | |
|---|---|---|---|
| Version | 6 | 1 | Protocol version number |
| Flags | 7 | 1 | Processing option bitmask |
| Encoding | 8 | 1 | Source text encoding identifier |
| Format | 9 | 1 | Genomic sequence format identifier |
| Public Key | 10 | 32 | Ephemeral X25519 public key |
| Nonce | 42 | 12 | GCM initialization vector |

**Implementation And Performance Analysis**

BIOCRYPT is implemented in Rust [12], a systems programming language whose ownership model and borrow checker enforce memory safety at compile time without imposing a garbage-collection overhead. Rust eliminates entire classes of memory safety defects—including buffer overflows, use-after-free errors, and data races—that are a significant source of vulnerabilities in security-critical software.

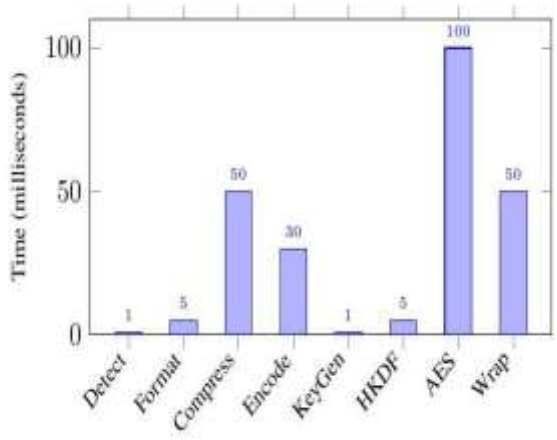
**Cryptographic Library Dependencies**

The implementation depends on audited cryptographic libraries from the RustCrypto project and the dalek-cryptography project, both of which have received detailed independent security reviews and are widely deployed in production systems. Key dependencies include aes-gcm for authenticated encryption, x25519-dalek [26] for elliptic-curve operations, sha2 and hkdf for key derivation, and age for the key-wrapping protocol.

**Performance Characteristics**

Figure 5 presents measured execution times for each pipeline stage when processing a 1 MB DNA sequence file. AES-GCM encryption is the dominant cost, accounting for approximately 41% of total processing time. Format detection, LZ4 compression, and 2-bit encoding each contribute measurable but smaller proportions.

**Figure 5: Encryption pipeline stage timing for a 1 MB input file.**




A total processing time of approximately 244 ms for a 1 MB file yields an encryption throughput of:

Throughput = 1 MB / 244 ms = 1000 KB / 0.244 s ≈ 4.1 MB/s

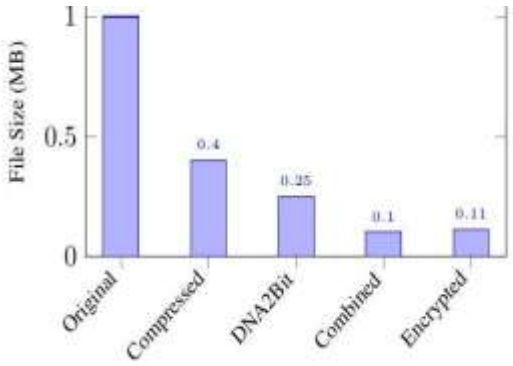
(34)

This throughput is sufficient for both interactive analysis and routine batch processing of genomic datasets of sizes typical in current research and clinical workflows.

**Storage Efficiency Results**

Figure 6 illustrates the progressive reduction in file size through the pipeline. Beginning from a 1 MB source file, combined 2-bit encoding and LZ4 compression reduce the pre-encryption data volume to approximately 0.10 MB, and the final encrypted output is 0.11 MB—an overall storage reduction of 89%.

**Figure 6: Progressive storage reduction through the encryption pipeline.**




The overall storage efficiency is:

$\eta_{total} = 1 - Size_{encrypted} / Size_{original} = 1 - 0.11/1.0 = 0.89 = 89\%$

(35)

**Security Analysis**

**Threat Model**

The security model addresses the following threat categories relevant to genomic data protection.

**Eavesdropping.** An adversary that intercepts encrypted files in transit is unable to recover the plaintext because AES-256-GCM provides computational indistinguishability under chosen-ciphertext attack (IND-CCA2) [9].

**Key interception.** Session keys are never transmitted in cleartext. Age-based key wrapping ensures that the raw key material remains protected throughout distribution.

**Data tampering.** Any modification to the ciphertext or to the associated header data is detected through verification of the 128-bit GCM authentication tag prior to decryption.

**Format confusion.** Strict input format validation is performed before any parsing commences, preventing adversarial inputs from exploiting format-handling code paths.

The current framework does not address quantum adversaries. X25519 is vulnerable to Shor's algorithm on a sufficiently powerful quantum computer. AES-256 retains approximately 128 bits of security under Grover's algorithm, but X25519 key exchange would require replacement with a post-quantum algorithm such as ML-KEM to maintain equivalent security guarantees [23].

**Cryptographic Security Bounds**

The advantage of any adversary A against the AES-256-GCM authenticated-encryption scheme is bounded by [9]:

$Adv^{AE}_{AES\text{-}GCM}(A) \le q^2/2^{128} + \sigma^2/2^{128} + Adv^{PRP}_{AES}(B)$

(36)

where q is the number of encryption queries, σ is the total plaintext length in 128-bit blocks, and $Adv^{PRP}_{AES}(B)$ is the advantage of a distinguisher against AES as a pseudorandom permutation.

The advantage of any polynomial-time adversary against the X25519 key exchange is bounded by:

$Adv^{CDH}_{Curve25519}(A) \le negl(\lambda)$

(37)

under the computational Diffie–Hellman assumption for Curve25519.

**Nonce Management and Collision Analysis**

GCM security requires that each nonce be unique for a given key. DNASSF generates each nonce by sampling uniformly from the 96-bit space:

$$N \leftarrow^{\$} \{0, 1\}^{96} \qquad (38)$$

The probability of a nonce collision among n independently generated nonces is bounded by the birthday approximation:

$Pr[\text{collision among n nonces}] \le n(n-1) / (2 \cdot 2^{96}) \approx n^2/2^{97}$

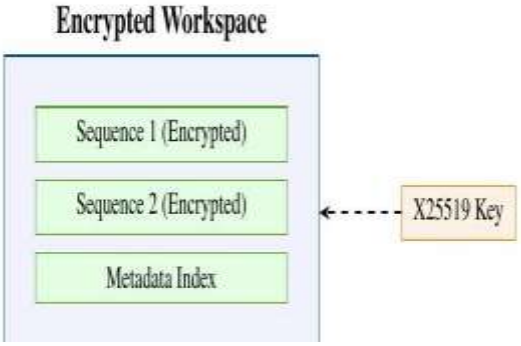
(39)

For $n = 2^{32}$ encryptions under a single key, the collision probability is approximately $2^{-33}$, which is negligible for practical deployment scales. Per-file key isolation further ensures that nonces from different files never share a key context.

**Encrypted Workspace System**

The workspace system enables management of multiple encrypted sequences within a single protected container, supporting collaborative research teams while enforcing cryptographic access controls over each individual sequence. The workspace is implemented as an encrypted SQLite database that stores independently encrypted sequences alongside their associated metadata index.

**Figure 7: Encrypted workspace architecture for multi-sequence collaborative management.**




Access to the workspace is restricted to holders of the container's X25519 private key. The system supports workspace creation, labeled sequence insertion, enumeration of stored sequences, and extraction of individual sequences for analysis. Each sequence is encrypted independently, preserving per-file key isolation even within a shared container.

**Applications And Deployment Scenarios**

BIOCRYPT supports a range of operational contexts in genomic research, clinical diagnostics, and commercial biotechnology.

**Research institutions.** Secure workspace containers facilitate collaborative sharing of proprietary sequence data among distributed research teams while maintaining confidentiality and access control.

**Clinical laboratories.** The framework supports compliance with healthcare data protection regulations, including HIPAA requirements to encrypt protected health information [30] and GDPR Article 9 obligations governing special categories of personal data [29].

**Biotechnology firms.** Encrypted storage of proprietary genetic sequences supports patent prosecution and trade secret protection for commercially sensitive intellectual property.

**Multi-site collaboration.** In a representative workflow, the sending institution encrypts sequence data under the recipient's public key, producing an encrypted data file and a separately transmitted wrapped key file. The receiving institution applies its private key to recover the session key and decrypts the data for analysis. End-to-end audit trails support compliance monitoring and access accountability.

**Future Development Directions**

Several extensions are planned for subsequent versions of BIOCRYPT.

**Post-quantum cryptography.** Replacing X25519 and related primitives with quantum-resistant algorithms—specifically ML-KEM for key encapsulation and ML-DSA for digital signatures—will protect long-lived genomic archives against future quantum adversaries [23].

**Hardware security module integration.** Storing private keys in tamper-resistant hardware, including YubiKey devices and TPM chips, will provide a higher assurance key storage option for clinical and regulated environments.

**OS-integrated secure storage.** Leveraging platform-native key stores—Windows DPAPI, macOS Keychain, and the Linux Secret Service—will align cryptographic key storage with the security model of each deployment platform.

**Distributed storage backend.** Extending the storage layer to support cloud storage providers and decentralized storage networks will broaden applicability while preserving the cryptographic guarantees of the core protocol.

## Conclusion

This paper has presented BIOCRYPT, a cryptographic framework for protecting genomic data at rest and in transit through the DNASSF protocol. The framework applies AES-256-GCM authenticated encryption, X25519 elliptic-curve key exchange, and HKDF-SHA256 key derivation to protect the confidentiality and integrity of sensitive genetic sequences. A 2-bit nucleotide encoding scheme combined with LZ4 compression reduces storage requirements by up to 89% without compromising cryptographic security. The Age-based key wrapping protocol enables secure session key distribution across organizational boundaries without exposing raw key material during transmission.

Implementation in Rust provides compile-time memory safety guarantees, eliminating buffer overflow and use-after-free vulnerabilities that commonly affect security-critical codebases. Performance evaluation demonstrates throughput of approximately 4.1 MB/s, sufficient for the interactive and batch genomic workloads typical of current research and clinical

settings. The modular architecture permits independent auditing of each security-critical component and supports future extension, including the integration of post-quantum cryptographic algorithms.

BIOCRYPT addresses a genuine and growing need for cryptographic methods tailored to the structural and operational requirements of genomics. It offers researchers, clinicians, and biotechnology professionals a principled approach to protecting sensitive genetic data throughout the complete data lifecycle.

## References


[1] M. Naveed, E. Ayday, E. W. Clayton, et al., "Privacy in the Genomic Era," ACM Computing Surveys, vol. 48, no. 1, pp. 6:1–6:44, 2015.

[2] Grand View Research, "Genomics Market Size Report, 2023–2030," Technical Report, 2023.

[3] J. Kim et al., "Security Practices in Genomic Research Institutions," Nature Genetics, vol. 54, pp. 1128–1134, 2022.

[4] IAPP, "Genetic Data Breach Report 2020," Privacy Technical Report, 2020.

[5] E. Ayday, J. L. Raisaro, J. P. Hubaux, "Privacy-Preserving Processing of Raw Genomic Data," Proc. DPM/SETOP, pp. 133–147, 2013.

[6] M. M. Aziz et al., "DNA-Based Cryptographic Methods: A Survey," Biosystems, vol. 145, pp. 28–37, 2016.

[7] S. Chen et al., "CryptGenome: Secure Genomic Data Sharing," Bioinformatics, vol. 36, no. 12, 2020.

[8] F. Valsorda, "age: A Simple, Modern, and Secure File Encryption Tool," GitHub, 2019. [Online]. Available: https://github.com/FiloSottile/age

[9] NIST SP 800-38D, "Recommendation for Block Cipher Modes of Operation: GCM and GMAC," National Institute of Standards and Technology, 2007.

[10] NIST SP 800-56C, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes," Rev. 2, National Institute of Standards and Technology, 2020.

[11] D. J. Bernstein, "Curve25519: New Diffie–Hellman Speed Records," Proc. PKC, pp. 207–228, 2006.

[12] R. Jung et al., "RustBelt: Securing the Foundations of the Rust Programming Language," Proc. POPL, pp. 66:1–66:34, 2018.

[13] Y. Collet, "LZ4: Extremely Fast Compression Algorithm," 2011. [Online]. Available: https://lz4.github.io/lz4/

[14] W. R. Pearson, D. J. Lipman, "Improved Tools for Biological Sequence Comparison," Proceedings of the National Academy of Sciences, vol. 85, pp. 2444–2448, 1988.

[15] P. J. Cock et al., "The Sanger FASTQ File Format for Sequences with Quality Scores," Nucleic Acids Research, vol. 38, no. 6, pp. 1767–1771, 2010.

[16] H. Li et al., "The Sequence Alignment/Map Format and SAMtools," Bioinformatics, vol. 25, no. 16, pp. 2078–2079, 2009.

[17] P. Danecek et al., "The Variant Call Format and VCFtools," Bioinformatics, vol. 27, no. 15, pp. 2156–2158, 2011.

[18] K. Lauter et al., "Healthentic: Privacy-Preserving Medical Data Analysis Using Homomorphic Encryption," Proc. Cloud Computing Security Workshop, 2011.

[19] S. Simmons et al., "Realizing Privacy-Preserving Genome-Wide Association Studies," Bioinformatics, vol. 32, no. 9, pp. 1293–1300, 2016.

[20] A. Shae, J. Tsai, "A Blockchain-Based System for Secure Sharing of Genomic Data," Proc. IEEE Engineering in Medicine and Biology Society, 2017.

[21] M. Chen et al., "Princess: Privacy-Preserving Rare Disease Genetic Testing in a Public Cloud," Bioinformatics, vol. 33, no. 16, pp. 2503–2511, 2017.

[22] D. Stein et al., "Securing the Sequence: Security and Privacy in the Age of Cloud Computing Genomics," Trends in Genetics, 2023.

[23] T. Lepoint et al., "Post-Quantum Cryptography for Genomic Privacy," Journal of Cryptographic Engineering, 2022.

[24] S. Gueron, "Intel's New AES-GCM Implementation," IEEE Security & Privacy, 2010.

[25] A. Biryukov et al., "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications," Proc. IEEE European Symposium on Security and Privacy, 2016.

[26] dalek-cryptography Team, "x25519-dalek: Pure-Rust Implementation of Curve25519," GitHub, 2023.

[27] H. Krawczyk, P. Eronen, "HMAC-Based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, Internet Engineering Task Force, 2010.

[28] ISO/TC 215, "Health Informatics—Genomic Sequence Variation Reporting Selection," ISO 25720, 2023.

[29] European Parliament, "General Data Protection Regulation (GDPR) Article 9: Processing of Special Categories of Personal Data," 2016.

[30] U.S. Department of Health and Human Services, "HIPAA Privacy Rule and Sharing Information Related to Mental Health," 2013.