Computer Fraud and Security
ISSN (online): 1873-7056

Architecting Privacy-Preserving Distributed Systems for Handling User Data

Projjal Ghosh

Independent Researcher, USA

Abstract

Distributed systems that handle sensitive user data face concerning issues in ensuring the end-to-end encryption of the data lifecycle. This article introduces an architecture that guarantees encrypted user data transmission from the source to processing stages, till storage, and is only decrypted in cryptographically-secure environments. The system incorporates hardware security modules to guard encryption keys, trusted execution environments in confidential virtual machines to process encrypted data without exposing plaintext, and authenticated encryption to support persistent storage. Remote attestation protocols ensure the integrity of code before decryption, whereas binary transparency mechanisms permit independent verification of processing behavior. The architecture solves the threats to encrypted data, such as key compromise, memory extraction, storage tampering, and insider access, using layered cryptographic controls. Guidance on implementation includes the initiation of client-side encryption, secure key distribution hierarchies, isolated processing in hardware enclaves, and integrity-protected storage mechanisms. The resulting system allows enterprises to provide cryptographic security of user data over a distributed infrastructure and comply with regulatory data confidentiality requirements and privacy-preserving analytics with differential privacy methods.

Keywords: End-to-End Encryption, Trusted Execution Environments, Hardware Security Modules, Cryptographic Key Management, Binary Transparency

1. Introduction

To ensure the security of sensitive user data in distributed systems, it is necessary to ensure the existence of cryptographic boundaries in the whole data lifecycle. End-to-end encryption means that user information is initially encrypted at the client, encrypted over the network, and only processed in cryptographically secure environments, and is stored encrypted. Any interruption in this encryption chain will reveal user information to unauthorized access. Secure processor architectures provide the basis of processing encrypted user data by developing isolated memory spaces in which decryption may be performed without exposing plaintext to the host system or administrators.**[3]**

A study of secure processor implementations shows that enclave page cache sizes of 64 MB to 256 MB can be used to support realistic workloads and still provide cryptographic isolation. Protected regions with a larger size scale their overheads on initialization to about 0.8 milliseconds per megabyte [1]. These remote enclaves store the contents of memory encrypted with 128-bit AES keys based on processor-internal root secrets that are not accessible to software, meaning that even physical memory attacks cannot retrieve plaintext user data during execution. The encryption imposed by the hardware establishes a trusted boundary such that user data may be temporarily decrypted to compute without compromising the end-to-end encryption promise.

Attestation mechanisms are necessary to ensure that the processing of user data in these isolated environments is correct. These processes create cryptographic measurements prior to any user data decryption. The attestation process generates REPORT structures with 384 bytes of measurement information, 256-bit SHA-2 hashes of loaded code, and 512-bit RSA signatures linking measurements to platform keys [1]. These measurements can be checked by remote parties before the transmission of encrypted user data so that only approved code with confirmed behavior will be allowed to access decryption keys. This chain of attestation ensures that code that is malicious does not pose as legitimate processors and steal user data.

The encryption level that secures user information should be resistant to attacks throughout the life of the data. 128-bit symmetric encryption keys are sufficient to protect the user data to 2030, and 256-bit keys to 2050, given the current cryptanalytic capabilities [2]. In key exchange operations that define the encryption of user data, asymmetric cryptography needs longer keys: 2048-bit RSA is 112-bit strong and 3072-bit RSA is 128-bit strong [2]. The correct

choice of key lengths will make sure that the encrypted user data will not be compromised during the retention period, even with the growth of computing power.

The choice of cryptographic algorithms to protect user data should consider the changing threats. This need is reflected in the migration of SHA-1 to the SHA-2 family functions, where collision attacks on SHA-1 have shown weaknesses that need more robust primitives [2]. Hash functions with 256-bit outputs offer 128-bit collision resistance, which is adequate to secure valuable user data over decades. Key derivation functions allow hierarchical key structures where a master key is used to create operation-specific keys, such that the compromise of a derived key to access one data batch of user data does not compromise other encrypted data accessed with different derived keys.

Hardware security modules that are tamper-resistant are necessary to protect the keys that encrypt user data. These devices use physical security measures that identify intrusion attempts. The sensors in the environment cause protective actions in case of temperature changes more than 10 degrees Celsius, voltage changes more than 5 percent of nominal values, or electromagnetic emissions typical of side-channel attacks [2]. When tampering is detected, key material is erased in milliseconds during zeroization processes, and keys that would be used to decrypt user data are not extracted. The key protection is done by hardware so that even physical access to the cryptographic infrastructure cannot decrypt encrypted user data.

2. Implementing Cryptographic Security of User Data

End-to-end encryption can be maintained by generating quality cryptographic keys that will secure user data during its lifecycle. Critical generation needs the sources of entropy that generate really random and unpredictable values. Entropy generation rates of over 100,000 bits per second are possible with hardware random number generators based on physical noise sources [3]. Statistical testing, such as frequency analysis, runs tests, and spectral analysis confirm that generated keys have no exploitable patterns that will compromise the encryption of user data.

Client-side key generation is used to guarantee that the encryption keys of the user data are never stored in plaintext within the control of the user. To encrypt bulk data, users locally generate symmetric encryption keys, usually choosing between 128-bit and 256-bit encryption key lengths depending on the sensitivity of the data and the duration of protection. The length of keys that are to be used to secure user data after 2030 must be 256 bits in order to have sufficient security margins over future cryptanalytic improvements [3]. Computational overhead of longer keys is not very problematic: software implementations demonstrate a 256-bit operation about 40 percent slower than a 128-bit operation, which is a sensible tradeoff to achieve longer security.

The distribution of encryption keys of user data to distributed systems necessitates key exchange protocols to create common secrets without the transmission of keys in plaintext. Elliptic curve cryptography offers an efficient key exchange, and 256-bit curves are as secure as 3072-bit RSA but use much less computational resources and smaller key representations [4]. This performance is essential in resource-limited settings that deal with encrypted user information. Embedded processor measurements indicate that point multiplication operations on NIST P-256 curves take about 23 milliseconds with 32-bit arithmetic, and optimized Curve25519 implementations take only 15 milliseconds [4]. These performance features allow key establishment in real time, even on low-processing-performance devices.

The key hierarchy and derivation must be considered when managing encryption keys during their lifetime. Hardware security modules hold master keys that are cryptographically derived to produce operation-specific derived keys. This top-down design implies that keys that encrypt batches of user data can be re-generated on command using master secrets instead of being stored in a persistent manner [3]. Revocation only impacts the individual user data that was encrypted using a compromised derived key, and does not impact other data encrypted using different derived keys. Hardware protection does not lose the master keys, so that even a large-scale attack on derived keys does not allow decryption of all user data.

The major rotation policies trade the benefits of security for the complexity of operation of systems that keep user data encrypted. The frequent rotation also constrains the amount of user data that is exposed in case of a key compromise, yet rotation necessitates the coordination of updates across distributed systems. The key rotation of high-sensitivity user data is generally 90 days, whereas less sensitive data can be done on an annual rotation schedule [3]. Systems also use dual-key windows of 24 to 48 hours during rotation periods, with both old and new keys being valid to allow seamless transition without service interruption to access encrypted user data.

Key Exchange Operation	Performance Metric
Random Key Generation Rate for User Data	100,000+ bits/second
Performance Cost of Stronger User Data Encryption	40% overhead (256 vs 128-bit)
Elliptic Curve Key Exchange Security Equivalence	256-bit ECC = 3072-bit RSA
User Data Key Exchange (NIST P-256)	23 milliseconds
User Data Key Exchange (Curve25519)	15 milliseconds
Key Distribution Architecture	Hierarchical derivation
High-Sensitivity User Data Key Rotation	90-day intervals
Key Transition Window for User Data	24-48 hours

Table 1: Key Exchange and Distribution Timing for User Data Encryption Initiation [3,4]

3. Handling Encrypted User Data in Isolated Environments

The processing of user data with end-to-end encryption and maintaining secrecy of the data must be done in environments where it can be decrypted without revealing plaintext beyond cryptographic security. Trusted execution environments provide hardware-isolated memory regions that encrypt all the contents at the memory controller level. Recent memory controllers can encrypt over 25 GB/s per channel with specialized AES engines, and encryption overhead is insignificant with most workloads [5]. This is a hardware-enforced encryption that even physical access to memory modules will not be able to retrieve plaintext user data during processing.

Memory protection of different granularities provides different tradeoffs between security and overhead in processing encrypted user data. Page-level encryption of individual 4 KB pages allows fine-grained access control but needs about 12.5 percent more memory to store integrity metadata. Metadata overhead is raised to 25 percent by cache-line granularity protection [5]. The option varies based on the threat model: page-level protection is more resistant to more advanced attacks that would seek to corrupt certain data structures; coarser protection is only necessary when memory isolation is sufficient to ensure user data security.

Memory encryption has performance effects that depend on the workload properties during user data processing. Operations that are compute-intensive and have high rates of cache hits have low overhead (less than 3 percent) because the majority of data accesses are in processor caches that do not need any encryption operations [5]. The workloads with high memory demands and poor cache locality incur 15 to 30 percent overhead as a result of the encrypted memory traffic and integrity checks. User data that has been encrypted by applications processing must maximize the use of the cache in order to reduce the performance impact without compromising the cryptographic protection.

Attestation protocols need to ensure that only approved code is executed in the isolated environment before any encrypted user data is processed. The attestation process produces evidence structures that store platform configuration measurements, enclave identity data, and cryptographic signatures relating these measurements to platform keys [6]. Generation of evidence takes less than 50 milliseconds,s even with ECDSA signing operations based on P-256 curves that produce signatures of 64 bytes. This evidence can be verified by remote parties before they can issue decryption keys, which means that user data will not be decrypted outside of a verified, trusted environment.

Transformation of attestation evidence into remotely verifiable quotes allows distributed verification of environments that process encrypted user data. Quote generation services authenticate local attestation evidence, retrieve pertinent measurements, and generate signed quotes with attestation keys supplied when the platform is manufactured [6]. This process introduces about 100 milliseconds of latency but generates 4 KB quote structures with platform firmware versions, enclave measurements, and full certification chains. These quotes can be verified without access to the platform, which is essential in distributed systems where user data processing is done on several independent nodes.

To verify attestation quotes, it is necessary to verify several cryptographic properties prior to the release of user data decryption keys. Verification checks more than 30 different fields in the structure of quotes, authenticating ECDSA signatures, comparing measurement hashes with reference values, and verifying attestation key revocation status [6]. Full

verification can be finished in less than 200 milliseconds on typical server processors, allowing real-time verification when connecting. The cryptographic binding of measured code and platform state guarantees that encrypted user data will not be decrypted in environments that execute verified, unmodified code that cannot perform unauthorized data exfiltration.

Authorization Stage	Timing/Specification
Memory Encryption Speed for User Data	25+ GB/s per channel
User Data Processing Memory Unit	4 KB pages
Metadata Overhead (Page-level Protection)	12.5% additional
Metadata Overhead (Cache-line Protection)	25% additional
Cryptographic Evidence Generation	50 milliseconds
Signature Algorithm for User Data Access	ECDSA P-256
Authorization Signature Size	64 bytes
Remote Authorization Quote Generation	100 milliseconds
Complete Authorization Data Package	4 KB
Verification Checkpoints per Request	30 fields
Total Authorization Verification Time	200 milliseconds

Table 2: Encrypted User Data Processing Authorization Timeline [5,6]

4. Encryption of Persistent User Data

Any user data that is not required to be processed should be re-encrypted before storage to ensure end-to-end encryption. An authenticated encryption scheme that provides confidentiality and integrity protection prevents unauthorized access and undetected tampering. AEAD paradigm encrypts user data and related metadata, and the encrypted data is generated with authentication tags that are verified during decryption [7]. Tag verification. The tag verification is an automatic rejection of corrupted or tampered data before any processing, eliminating attacks that alter encrypted user data in storage.

Authenticated encryption of user data has a large variation in performance depending on the availability of hardware support. Implementations of GCM mode with the use of PCLMULQDQ instructions to accelerate the arithmetic of polynomials have a throughput of more than 1 GB/s, sufficient to process large volumes of user data [7]. Multiplication Software-only implementations with a lookup table-based multiplication are about 100 MB/s throughput. This performance gap of ten times highlights the need for hardware acceleration in systems where encrypted user data is processed in large volumes in production.

Effective nonce management is essential in achieving authenticated encryption security of user information. The reuse of nonces with the same key is disastrous to the confidentiality assurances and can reveal several encrypted records of user data. Birthday-bound collision probability is about 2-32 when encryption is performed on 232 blocks with one key, and requires key rotation before this probability approaches it [7]. Operational policies should restrict the amount of user data that is encrypted with individual keys, and security margins should be maintained significantly lower than theoretical. Normal deployments spin keys every time a portion of the theoretical limit is encrypted, which guarantees sufficient safety factors in the protection of user data over time.

Trusted platform modules offer hardware-isolated security over keys used to encrypt stored user data. TPM specifications specify about 25 different commands that assist key generation, encryption, signing, and attestation tasks, each of which takes multiple parameter configurations [8]. This command interface allows flexible security policy and hardware-enforced access controls to prevent unauthorized use of keys that can lead to the loss of encrypted user data. The hardware isolation is such that even privileged software is not able to extract keys with which user data in storage is encrypted.

Measurements of platform configuration held in TPMs generate cryptographic chains between encrypted user data and particular states of the system. PCR extended operations combine new measurements with the current register values with SHA-1 or SHA-256 to generate irreversible measurement chains [8]. These chains include boot firmware, operating system components, and application software. Any unauthorized changes to systems that handle encrypted user data generate different measurements, which can be detected by comparing them to the expected sequences, and then the data will be decrypted.

Sealing mechanisms tie encrypted user data to particular platform configurations and cannot be decrypted on an inappropriate platform. Encrypted user data and policy specifications are stored in sealed blobs that specify the necessary PCR values [8]. Unsealing is only successful when the current platform state is consistent with sealed policies, and encrypted user data is not accessible on attacked or tampered systems. This cryptographic binding offers high security against offline attacks where attackers have access to the physical storage media holding encrypted user information. Unauthorized modifications to systems leave them inaccessible, and the confidentiality of systems is enforced with cryptography instead of access controls that can be bypassed.

Storage Protection Feature	Implementation Details
Encryption Paradigm	AEAD: confidentiality + integrity combined
Automatic Corruption Rejection	Tag failures trigger rejection before processing
Hardware-Accelerated Performance	GCM with PCLMULQDQ: exceeding 1 GB/s
Software-Only Performance	Lookup table approach: approximately 100 MB/s
Hardware Isolation Guarantee	Prevents privileged software key extraction
Measurement Chain Mechanism	PCR extended with SHA-1 or SHA-256
Measurement Scope	Boot firmware through OS to applications
Sealing Binding Approach	User data bound to platform configurations
Sealed Blob Components	Encrypted data + policy specifications
Unsealing Condition	Platform state must match policy requirements
Self-Protection Characteristic	Auto-inaccessible after unauthorized modifications

Table 3: Cryptographic Storage Protection for Persistent User Data [7,8]

5. Protection of Encrypted User Data against Attack Vectors

Timing attacks with implementation vulnerabilities affect key exchange operations that create encryption of user data. When using Diffie-Hellman, the modular exponentiation operations have timing variations, which may reveal some secret exponents with careful measurement [9]. Conditional branches and variable-iteration loops incur timing differences of microseconds to milliseconds based on bit patterns in secret exponents. Constant-time implementations that remove timing differences based on key values are important in preventing the extraction of user data encryption keys by timing attacks.

Unauthenticated key exchanges can be attacked by a man-in-the-middle to intercept and decrypt user data by connecting to each of the parties separately. The success of attacks occurs when parties have no mechanisms that confirm that negotiated keys are those that are intended to be used in communication with the partners [9]. The need to ensure that the key material is bound to known identities is demonstrated by authentication protocols that are used to protect the encryption of user data during key establishment. Authentication using certificates or pre-shared secrets allows detection of intermediary manipulation during exchange phases, which prevents adversaries from creating a position to decrypt user data.

Small subgroup attacks are attacks that take advantage of poor parameter selection in key exchange protocols that coerce agreement into subgroups in which discrete logarithm problems are solvable. Small-factor groups allow opponents to retrieve some of the private exponents by solving smaller-complexity problems in subgroups of size 220 or less [9].

Protecting encrypted user data involves a comprehensive validation of received public keys, whereby, before key derivation, membership in large prime-order subgroups is validated. The rejection of values that do not pass the membership test does not allow adversaries to compel weak keys that can be used to decrypt user data.

Differential privacy protocols allow analytics to be done on encrypted user information, and re-identification is avoided. The implementations add noise to the query output, which is proportional to the privacy budget parameter epsilon in an inverted manner [10]. It is found that the epsilon of 0.5 to 2.0 has the best tradeoffs, offering meaningful privacy protection and data utility of over 80 percent of unperturbed baselines. This allows useful analytics on encrypted sets of user data without breaking down the privacy of individuals by re-identification attack.

Disclosure of privacy safeguards is a major determinant of readiness to submit encrypted user data. Experiments show that the rate of sharing increases by 15-25 percent when the mechanisms of differential privacy are explained, as opposed to when there are no descriptions of privacy mechanisms [10]. This effect of behavior suggests that the perception of trust and the ensuing data-sharing decisions are affected by the clear communication of technical safeguards of user data. Companies that deal with encrypted user information ought to offer clear descriptions of protection systems to gain the trust of users.

It is important to consider the effects of query composition on encrypted user data when managing privacy budget. Sequential query processing loses privacy linearly with the number of queries, whereas parallel query processing of independent queries incurs fixed privacy budgets [10]. System architectures that cluster similar queries to be executed in parallel optimize the utility of data under a given privacy constraint. According to statistical analysis, epsilon values of 1.0 allow about 100 consecutive queries with noise per query scaled to query sensitivity, a tradeoff between long-term usability and high privacy assurances to encrypted user data over long periods of operation.

Security Concern	Mitigation Strategy/Configuration
Implementation Vulnerability Source	Conditional branches and variable-iteration loops
Man-in-the-Middle Attack Vector	Unauthenticated exchanges enable session interception
Authentication Requirement	Certificate-based authentication or pre-shared secrets
Differential Privacy Purpose	Analytics without user re-identification
Noise Calibration Approach	Noise is inversely proportional to epsilon
Optimal Privacy Budget Range	Epsilon: 0.5 to 2.0
Transparency Communication Effect	15-25% increase in user data sharing
Sequential Query Privacy Loss	Linear accumulation with query count
Parallel Query Privacy Loss	Constant regardless of parallelism
Query Noise Adjustment	Scaled to individual query sensitivity
Long-term Protection Balance	Usability with strong privacy guarantees maintained

Table 4: Attack Mitigation and Privacy Preservation for User Data [9,10]

Conclusion

To ensure that sensitive user data in distributed systems is maintained with end-to-end encryption, cryptographic protection must be coordinated at all the data lifecycle phases. The presented architectural framework defines cryptographic boundaries between the first encryption at client devices and processing in hardware-isolated environments to long-lasting storage with integrity protection. Trusted execution environments can be used to execute encrypted user data by providing hardware-enforced isolation such that temporary decryption cannot reveal plaintext beyond the boundaries of verified code. Hardware security modules secure the encryption keys that allow the user to protect their data so that it cannot be extracted by either physical or logical attack. Binary transparency schemes allow external verification that user data in code processing is acting as intended, and give empirical evidence to privacy claims. Authenticated encryption keeps the stored user data safe, and the differential privacy methods allow useful analytics

without violating the privacy of a person. The multi-tiered strategy means that several independent defenses have to fail before encrypted user data can be revealed, and the risk is minimal enough to allow deploying the system in an enterprise. Hardware acceleration of cryptographic operations makes performance characteristics viable for production workloads. Organizations that adopt these architectural patterns provide verifiable bases to manage sensitive user data, which allows innovation in data-driven services without sacrificing cryptographic guarantees across the entire data lifecycle.

References

[1] Victor Costan et al., "Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture", Massachusetts Institute of Technology, 2017. Available: https://people.csail.mit.edu/devadas/pubs/part_2.pdf

[2] Elaine Barker, "Recommendation for Key Management: Part 1 – General", NIST, 2020. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

[3] Abhishake Reddy Onteddu, Rahul Reddy Bandhela; RamMohan Reddy Kundavaram. Enhancing E-Commerce Product Recommendations through Data Engineering and Machine Learning. ES 2024, 20 (1), 171-183. https://doi.org/10.69889/vqgzw857.

[4] Elaine Barker and Quynh Dang, "Recommendation for Key Management - Part 3: Application-Specific Key Management Guidance", NIST, 2015. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

[5] Vinayak Tanksale, "Efficient Elliptic Curve Diffie–Hellman Key Exchange for Resource-Constrained IoT Devices", MDPI, 2024. Available: https://www.mdpi.com/2079-9292/13/18/3631

[6] Saeid Mofrad et al., "A Comparison Study of Intel SGX and AMD Memory Encryption Technology", HASP'18 - ACM, 2018. Available: https://dl.acm.org/doi/epdf/10.1145/3214292.3214301

[7] Intel, "Intel®SGX Data Center Attestation Primitives (Intel®SGX DCAP)". Available: https://www.intel.com/content/dam/develop/public/us/en/documents/intel-sgx-dcap-ecdsa-orientation.pdf

[8] Phillip Rogaway, "Authenticated-Encryption with Associated-Data", University of California at Davis, 2002. Available: https://web.cs.ucdavis.edu/~rogaway/papers/ad.pdf

[9] Ms. Poojashree et al., "Trusted Platform Module for Security in Cloud Computing," IJERT, 2015. Available: https://www.ijert.org/research/trusted-platform-module-for-security-in-cloud-computing-IJERTCONV3IS14007.pdf

[10] Iraj Fathirad et al., "Network-Specific Attacks on Diffie-Hellman Key-Exchange in Commercial Protocols", International Journal of Computer Theory and Engineering, 2016. Available: https://www.ijcte.org/vol8/1031-C092.pdf

[11] Michael Khavkin and Eran Toch, "Investigating the impact of differential privacy obfuscation on users' data disclosure decisions", ScienceDirect, Sep. 2025. Available: https://www.sciencedirect.com/science/article/pii/S0167923625000752