

Adapting Ephemeral ECC Authentication for Decentralized Web 3.0: A Threat Model and Security Analysis

Adarsh S V Nair*, Rathnakar Achary

Alliance School of Advanced Computing, Alliance University, Bengaluru, India

Abstract—Web 3.0 shifts control of digital identity and data away from centralized platforms, moving authentication onto resource-constrained end-user devices, wallets, and smart contracts. RSA and ElGamal are computationally expensive at the key sizes required for adequate security, making them a poor match for this setting. This study presents a user-centric Web 3.0 authentication framework combining Elliptic Curve Cryptography with an ephemeral Elliptic Curve ElGamal (ECEG) character-encoding scheme, building on an ephemeral-encoding design for constrained IoT devices and its subsequent hash/XOR authentication extension. Unlike that prior design, the mutual-authentication step here binds each party to its long-term key through digital signatures and replaces static-key ECEG transport with a two-ephemeral-key ECEG transport, providing an erasure-conditional forward-secrecy property that the prior design lacks. We define a Dolev-Yao threat model with an in-scope/out-of-scope adversary boundary, an assumption-based security analysis, a formal-verification specification, and a decentralized-identifier/verifiable-credential mechanism binding long-term public keys to claimed identities, addressing the identity-binding gap that otherwise limits man-in-the-middle resistance without a common certificate authority. A protocol-level operation count and a complexity comparison against RSA are reported without claiming empirical superiority. We situate the framework against its closest prior IoT-oriented work and active 2025–2026 literature on ECC- and blockchain-based decentralized authentication. Recent 2026 analyses have reduced estimates of the quantum resources needed to solve the elliptic-curve discrete logarithm problem underlying secp256k1-style curves; we give a post-quantum outlook against current NIST migration guidance. The evaluation remains analytical rather than empirical; formal verification and hardware benchmarking remain necessary before production deployment.

Keywords—*Elliptic curve cryptography; decentralized authentication; Web 3.0 security; privacy-preserving authentication; self-sovereign identity; decentralized identifiers; verifiable credentials; forward secrecy; formal verification; resource-constrained devices; post-quantum cryptography*

I. INTRODUCTION

The development of the World Wide Web is generally described in three overlapping phases. Web 1.0 was a static medium in which users were essentially passive consumers of information provided by publishers. Web 2.0 swung the pendulum towards participation, allowing for social interaction and user-generated content, but then conversely centralized data, identity, and economic value in the hands of a few large

platforms. Web 3.0 is best understood as a correction to this concentration: a computing paradigm that uses blockchain infrastructure, cryptographic identity primitives, and peer-to-peer protocols to return control of data and digital identity to the people who created it.

This change of control does not remove the need for security; it moves it. In Web 2.0, a central identity provider can enforce authentication policy, patch vulnerabilities, and revoke compromised credentials for its users. In a decentralized environment, this burden falls on end-user devices, wallets, and smart contracts, which usually have limited computational power, memory, and energy budgets. Decentralized identity technologies such as Decentralized Identifiers (DIDs), Self-Sovereign Identity (SSI), and Zero-Knowledge Proofs (ZKPs) give individuals cryptographic control of credentials without a centralized issuing authority. These mechanisms are attractive because they minimize third-party trust, but they also raise the bar for the cryptographic building blocks that underpin them: authentication must be robust enough to withstand active adversaries, yet lightweight enough to run on constrained hardware such as embedded wallets, IoT gateways, and mobile edge devices participating in decentralized applications.

This tension has practical consequences that are visible in the attack landscape that has grown alongside Web 3.0. Social-engineering attacks such as rug pulls, Sybil attacks, and phishing trick users into giving up credentials or authorizing malicious transactions, and weak or spoofable authentication is a direct contributor to several of these: if the link between a user and their cryptographic identity can be spoofed, replayed, or intercepted, an attacker can impersonate a legitimate party or enrol duplicate identities under someone else's claimed identity. Smart contracts are self-executing and largely immutable once deployed, which makes them susceptible to logic errors and reentrancy attacks, and cryptojacking silently hijacks computing power for unauthorized mining; these two threats are primarily software-security and resource-abuse problems rather than authentication failures, and the framework proposed here does not address them (Section X). Strengthening the authentication layer is therefore not a complete defense for Web 3.0 but a necessary precondition for the identity-dependent parts of its security promises [1].

Public-key cryptosystems such as RSA and ElGamal have been the standard building blocks for mutual authentication between distributed entities for decades, but their computational profile, dominated by large-modulus modular exponentiation,

*Corresponding author.

and their reliance on large keys make them poorly suited to resource-constrained devices operating within decentralized networks. Elliptic Curve Cryptography (ECC) is an alternative that provides comparable security with a much smaller key size, resulting in less computation time, a smaller memory footprint, and lower communication overhead. These properties make ECC a natural candidate for authentication in Web 3.0 settings, provided that the protocol built around it is designed with the specific trust assumptions of decentralized environments in mind.

This study presents a user-centric authentication framework for Web 3.0 that uses Elliptic Curve Cryptography together with an ephemeral Elliptic Curve ElGamal (ECEG) character-encoding scheme, a two-ephemeral-key ECEG parameter transport, and a digital-signature-based mutual authentication step. The ephemeral ECEG character encoding builds on the encoding concept originally proposed for constrained IoT devices by Baccouri et al. [2], and the goal of binding the resulting encoding parameters to an authenticated exchange builds on that group's own subsequent extension of the idea, which transports the encoding parameters inside an ECEG ciphertext under the recipient's static public key and confirms them with a hash-and-XOR exchange chained to the previous session's parameters, verified with the Scyther tool and benchmarked against RSA [3]. As Section II-D discusses in detail, this study departs from [3] at the protocol level as well as the application level: mutual authentication here rests on digital signatures over each party's long-term key rather than on the hash-and-XOR confirmation of [3], and the encoding parameters are transported under a fresh, per-session ephemeral key rather than under the recipient's static public key. Under the stated erasure assumption, this removes the static-key forward-secrecy limitation of [3] (Section VI-D). The present study's contribution is therefore both this protocol-level adaptation and the Web 3.0-specific threat model, requirements analysis, and identity-binding mechanism built around it, for the setting in which two entities may share no prior relationship and no common certificate authority. The contributions of this work are summarized as follows.

- A review of the authentication and identity-management challenges specific to decentralized Web 3.0 architectures, linking them to concrete attack patterns reported in the literature.
- An explicit, requirement-driven threat model for Web 3.0 peer-to-peer authentication, together with an honest positioning of this work relative to its closest prior art (Section II-D).
- A Web 3.0-oriented adaptation of an existing ECEG character-encoding scheme [2,3], replacing its static-key encoding transport with a two-ephemeral-key ECEG exchange and its mutual-authentication step with digital signatures over each party's long-term key, reanalyzed for the trust assumptions of self-sovereign, peer-to-peer Web 3.0 identity. The adaptation is designed to operate without exposing long-term private keys and provides an erasure-conditional forward-secrecy property that the prior static-key design lacks.

- An informal, assumption-based security analysis against eavesdropping, man-in-the-middle, replay, and impersonation attacks – with man-in-the-middle resistance stated explicitly as conditional on authenticated public-key binding rather than as an unqualified guarantee – together with a concrete formal-verification plan (Section VI-F) that specifies the exact queries needed to move from an informal argument to a machine-checked one.
- A concrete decentralized-identifier/verifiable-credential (DID/VC) mechanism (Section VI-F) for binding long-term public keys to claimed identities, closing the residual key-to-identity binding gap that otherwise limits man-in-the-middle resistance in a peer-to-peer setting with no common certificate authority.
- A computational complexity comparison between RSA and ECC, checked against the current NIST SP 800-57 equivalence table, together with an honest discussion of the limits of the present analytical-only evaluation and concrete directions for hardware-based validation.
- A post-quantum outlook (Section IX) that situates the proposed ECC-based framework against the rapidly narrowing 2025–2026 quantum resource estimates for solving the elliptic-curve discrete logarithm problem, and against current NIST migration guidance for algorithm deprecation.

The remainder of this study is organized as follows. Section II reviews related work on public-key and elliptic-curve-based authentication for constrained and decentralized systems, and positions this work relative to its closest prior art. Section III introduces the mathematical preliminaries of elliptic curve cryptography used throughout the study. Section IV defines the threat model and security objectives. Section V presents the proposed authentication framework in detail. Section VI gives the security analysis, including the formal-verification plan. Section VII studies computational complexity relative to RSA. Section VIII discusses the results. Section IX gives a post-quantum outlook. Section X discusses limitations and future work, and Section XI concludes the study.

II. RELATED WORK

A. Public-Key Cryptography as a Baseline for Authentication

Traditionally, public-key cryptosystems such as RSA and ElGamal have been used to achieve mutual authentication between distributed entities, where the security of the scheme rests on the difficulty of factoring large integers or computing discrete logarithms in large multiplicative groups. These systems require key sizes of two to four thousand bits or more for an adequate security margin, and the modular exponentiation operations involved are computationally intensive. This overhead is manageable on servers and general-purpose computers, but becomes a bottleneck on the memory- and energy-constrained devices common in Internet of Things (IoT) deployments, and increasingly in Web 3.0 client software such as wallets and decentralized-application agents. This limitation motivates the search for cryptographic primitives that realize similar security with a smaller computational footprint.

B. Elliptic Curve Cryptography for Constrained and Decentralized Systems

Elliptic Curve Cryptography is increasingly the preferred choice for constrained environments because it offers security comparable to RSA with much shorter key lengths, owing to the higher per-bit strength of the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC has been used for authentication in resource-constrained settings in a large body of work. Dinarvand and Barati [4] proposed an efficient and secure ECC-based RFID authentication protocol; Lamrani Alaoui et al. [5] proposed a highly efficient ECC-based authentication protocol for RFID tags under similarly tight resource constraints. Noori et al. [6] extended this line of work to the healthcare Internet of Things with an ECC-based mutual authentication scheme for RFID-enabled medical devices, and Sowjanya et al. [7] designed an ECC-based authentication scheme specifically for the Internet of Medical Things. Khan et al. [8] proposed a secure framework combining authentication and encryption with an improved ECC construction for IoT-based medical sensor data, illustrating the recurring role of healthcare IoT as a proving ground for lightweight ECC protocols.

ECC authentication using smart cards and multi-factor schemes has also been heavily explored. Kumari et al. [9] proposed a smart-card-based mutual authentication protocol named ESEAP, and a related work by the same group [10] proposed a secure user authentication protocol based on ECC. Li et al. [11] proposed a robust and energy-efficient authentication protocol for the industrial Internet of Things, while Panja et al. [12] combined ECC with fingerprint biometrics to strengthen user binding. Ramasamy et al. [13] proposed an ECC-secured one-time-password scheme for mobile commerce; Sadri and Asaar [14] designed an anonymous two-factor authentication protocol for IoT applications with emphasis on user privacy; Elias et al. [15] implemented a one-time-password system on top of ECC key exchange; and Kaur et al. [16] proposed a two-factor authentication framework for cloud computing based on the same underlying elliptic-curve operations. A common observation across these contributions is that identity-based and certificate-based ECC schemes can fail either because they lack user anonymity, or because the authentication step itself remains vulnerable to replay and impersonation if ephemeral values are reused or predictable a gap this work directly addresses by using fresh ephemeral parameters on a per-session basis.

On the implementation side, Islam et al. [17] proposed high-performance ECC processors with unified point addition on twisted Edwards curves, and Basu Roy and Mukhopadhyay [18] presented a high-speed implementation of ECC scalar multiplication over prime fields for generic Montgomery curves, both demonstrating that the scalar multiplication at the heart of ECC can be realized efficiently even on constrained hardware. Lara-Nino et al. [19] provide a comprehensive survey of elliptic-curve lightweight cryptography that catalogs the design trade-offs between security margin, key size, and implementation cost that motivate the present work. The most directly related prior work is that of Baccouri et al. [2], who proposed an ephemeral encoding scheme in which plaintext characters are mapped onto curve points using randomly generated ephemeral parameters rather than a fixed, exchanged lookup table. The framework

proposed in this study builds on that ephemeral-encoding idea; Section II-D discusses in detail how it relates to that group's own subsequent extension of the same idea into a full authentication protocol [3].

The field has remained active through 2025 and into 2026. Prakash and Ramesh [20] proposed a decentralized ECC-based authentication scheme with a dynamic difficulty-adjustment mechanism for IoT networks, correctness argued with BAN logic and evaluated against replay and man-in-the-middle attacks, architecturally the closest 2025 comparator to the framework proposed here. Liu et al. [21] introduced EdenDID, an edge-computing and blockchain-based decentralized identity system for Web3 applications that binds a human user, wallet address, and physical device together specifically to resist Sybil attacks on Web3 wallets, and Le et al. [22] combined blockchain smart contracts with AI-driven document recognition and Merkle-tree-based access control for decentralized identity management. These 2025 contributions confirm that ECC-based and blockchain-based decentralized authentication remains a fast-moving area, which strengthens the case, developed in Section II-D, for differentiating this study's contribution narrowly and explicitly rather than on the underlying cryptographic primitives alone.

C. Blockchain, Identity, and Verification

Beyond point-to-point authentication, several works consider the broader decentralized identity and trust ecosystem in which such protocols operate. Achary et al. [1] proposed an integrated blockchain model for personally identifiable information systems, examining how blockchain-based identity management can mitigate the social-engineering and identity-theft risks characteristic of Web 3.0 platforms, and a related study [23] investigated the integration of ECC with the Constrained Application Protocol (CoAP) to secure resource-constrained IoT communication, a combination architecturally close to the decentralized, low-power devices targeted here. A related blockchain-enabled IoT logistics system for high-value shipment tracking [52] similarly pairs a smart contract with blockchain-based access control, illustrating the same decentralized-trust pattern applied to a supply chain rather than an authentication setting, and a multi-authentication smart-locker design [53] combines several authentication modalities in a deployed IoT system; both are system-level rather than protocol-level contributions, but they reinforce the point that authentication should be evaluated alongside privacy mechanisms and deployment cost rather than through key-size comparisons alone. Zheng et al. [24] proposed a blockchain-based lightweight authentication and key-agreement scheme for the Internet of Vehicles, illustrating how ECC-style lightweight authentication is increasingly paired with distributed ledgers rather than centralized authentication servers. Formal verification of such protocols is itself an active area: Cremers [25] developed Scyther, a tool for the automated verification of security protocols, and Blanchet [26] developed ProVerif, based on the applied pi calculus; both tools are used to motivate the formal-verification plan of Section VI-F. Finally, K. K. G. V. et al. [27] explored machine-learning-based anomaly detection for cloud data, a complementary line of defense that could, in future work, be layered on top of the authentication mechanism

proposed here to flag statistically anomalous authentication attempts rather than only individually valid or invalid ones.

Within the same domain, Kanjanaputhipong and Boonkrong [28] proposed a blockchain-based decentralized authentication paradigm for closed environments, with security argued using Gong–Needham–Yahalom (GNY) logic rather than the Dolev–Yao-style symbolic verification pursued here, illustrating the range of formal approaches the field currently uses. Recent systematization work has also matured: a companion comparative survey [29] in this journal analyzes the scalability, security, and feasibility trade-offs between centralized and decentralized identity management, and a 2026 systematization of knowledge [30] organizes the open challenges of self-sovereign digital identities across a broad cross-section of the recent literature.

D. Positioning Relative to the Closest Prior Art

The protocol most closely related to the one proposed here is the lightweight authentication scheme of Baccouri et al. [3], which extends their own earlier ephemeral-encoding proposal [2] by exchanging the ephemeral encoding parameters (α , β) inside an ECEG ciphertext C encrypted under the recipient's static public key, sending C together with a hash value $h1$ in the first message and returning a hash-and-XOR confirmation $CH1 = h3 \oplus h1$ whose hash input includes the previous session's encoding parameters; the resulting protocol is verified with the Scyther tool — checking secrecy, liveness, weak agreement, non-injective agreement and synchronization, and commitment to the exchanged values — and benchmarked against RSA on a C implementation across 8-, 64-, and 256-bit inputs. At the level of the cryptographic mechanism, that scheme and the framework proposed in this study therefore share the ephemeral ECEG character-encoding step of [2]; the mutual-authentication mechanism itself differs, since [3] authenticates via a hash-and-XOR confirmation that depends on state carried over from the previous session rather than via digital signatures over the current transcript, and transports the parameters under a static recipient key rather than the two-ephemeral-key ECEG transport used here (Section V-C). Table I sets these differences out message by message. Based on the openly available full text of [3], that protocol is designed and evaluated for classical IoT and wireless-sensor-network deployments between two devices provisioned in advance, and it does not address the specific trust assumptions of Web 3.0: there is no notion of a self-sovereign, wallet-held identity, no discussion of decentralized identifiers, and no attempt to reason about anonymity in the sense required by self-sovereign identity systems.

The contribution of the present study relative to [3] operates at two levels. At the protocol level, mutual authentication here rests on digital signatures over each party's long-term key combined with a two-ephemeral-key ECEG transport of the encoding parameters (Section V-C), replacing the static-key ECEG transport and previous-session-chained hash/XOR confirmation of [3], and providing an erasure-conditional forward-secrecy property that [3] does not have (Section VI-D). At the requirements level, the contribution is: 1) an explicit, requirement-driven threat model formulated for the trust assumptions of decentralized Web 3.0 identity, where the two parties may share no prior relationship and no common

certificate authority (Section IV); 2) a concrete decentralized-identifier/verifiable-credential mechanism for binding long-term public keys to claimed identities (Section VI-D), together with an accompanying discussion of anonymity, pseudonymity, and forward secrecy framed in self-sovereign-identity terms rather than device-provisioning terms (Section VI-E); and 3) a concrete, ready-to-execute formal-verification plan with explicit secrecy and injective-agreement queries (Section VI-F), intended to be carried out as direct follow-up work. Readers should treat the ephemeral ECEG character-encoding step of Section V-B as a design pattern inherited from [2] and [3]; the signature- and two-ephemeral-key-ECEG-based mutual-authentication mechanism built around it (Section V-C), the Web 3.0-specific requirements analysis, the identity-binding mechanism, and the verification plan are this study's own contribution, summarized in Table I.

TABLE I. MESSAGE-LEVEL AND TRUST-MODEL DIFFERENCES BETWEEN THE CLOSEST PRIOR SCHEME [3] AND THE PROTOCOL OF SECTION V

Aspect	Baccouri et al. [3]	This study (Section V)
ECEG character encoding	Ephemeral (α, β) encoding of [2]	Retained unchanged; inherited, not claimed as new
Parameter transport	ECEG ciphertext C under recipient's static public key	ECEG ciphertext (C_1, C_2) under recipient's fresh per-session ephemeral point R_b
Message 1	$C, h1$	$C, N_A, \sigma_A = \text{Sign}_{\{k_A\}}(\text{hash}(C \parallel N_A \parallel Q_A \parallel Q_B \parallel R_b))$
Message 2	$CH1 = h3 \oplus h1$ (hash input includes previous session's parameters)	$N_B, \sigma_B = \text{Sign}_{\{k_B\}}(\text{hash}(M \parallel N_A \parallel N_B \parallel Q_A \parallel Q_B))$
Authentication primitive	Hash and XOR	Schnorr/BIP-340 signatures over the current transcript
First-session bootstrap	Depends on previous-session state	Stateless per session; valid signature required regardless of history
Replay protection	Session chaining	Fresh 128-bit nonces + bounded replay cache
Forward secrecy	Not provided (static-key transport)	Provided conditionally if A erases i_r and B erases r_b after single use (Section VI-D)
Identity binding	Devices provisioned in advance	DID/VC binding check before the handshake (Section VI-B)
Added per-session cost	—	+1 ephemeral point, +2 signature ops, +2 verifications (Table VI); not claimed lighter than [3]
Validation	Scyther-verified; benchmark	Analytical; verification plan (Section VI-F) and benchmarking pending

E. Research Gap

The literature makes two things clear: 1) ECC is well suited to authentication on constrained devices because of its favorable security-to-key-size ratio; and 2) ephemeral, non-repeating cryptographic parameters are important for resisting replay and impersonation. What is comparatively unexplored, as Table II summarizes, is the combination of these two properties analyzed against the specific trust requirements of Web 3.0 identity management, where authentication must also support user-

controlled, self-sovereign identity without a centralized issuing or verifying authority. Existing ECC-based authentication schemes, including the closest prior art discussed in Section II-D, are analyzed for classical IoT, RFID, or wireless-sensor-network environments rather than for the decentralized application and wallet interactions of Web 3.0. The framework proposed in this study targets that gap. This remains true of the most recent 2025–2026 literature identified during the

preparation of this revision (Table II): the closest 2025 comparators [20,21,28] each address either dynamic difficulty adjustment, blockchain-level decentralized authentication, or Sybil-resistant wallet binding, but none combines ephemeral character-level ECEG encoding with a signature- and two-ephemeral-key-ECEG-based mutual-authentication handshake analyzed against a Web 3.0-specific threat model, which is the specific combination this study targets.

TABLE II. SUMMARY OF REPRESENTATIVE ECC-BASED AUTHENTICATION SCHEMES

Ref.	Target Domain	Core Mechanism	Noted Limitation
[4,5]	RFID	ECC-based mutual authentication for tags	Tight tag-side resource budget limits protocol complexity
[6–8]	Healthcare IoT / IoMT	ECC authentication and encryption for sensor data	Emphasis on confidentiality; limited anonymity guarantees
[9,10]	Smart-card authentication	ECC-based mutual authentication with smart cards	Card loss/cloning risk; anonymity depends on card security
[12]	Biometric authentication	ECC combined with fingerprint features	Requires biometric sensing hardware
[13,15]	OTP-based authentication	ECC key exchange for one-time-password generation	OTP channel itself can be a phishing target
[14]	IoT two-factor authentication	Anonymous two-factor protocol	Second factor adds device/hardware dependency
[2]	IoT messaging	Ephemeral ECC/ECEG character encoding (no authentication handshake)	No explicit mutual authentication handshake
[3]	IoT / wireless sensor networks	Ephemeral ECEG encoding + hash/XOR confirmation chained to previous session; Scyther-verified; RSA-benchmarked	Targets classical IoT/WSN; no Web 3.0/SSI trust model or DID/VC binding
[20]	IoT networks (decentralized)	ECC + dynamic difficulty adjustment; BAN-logic proof	No ephemeral character-encoding layer; not Web 3.0/SSI-specific
[28]	Closed blockchain environments	Blockchain-based decentralized authentication; GNY-logic proof	ECC used only for validator-node key agreement, not for end-user authentication
[21]	Web 3.0 wallets / DePIN	Edge computing + blockchain identity binding human, wallet, and device	Targets Sybil resistance specifically; no ephemeral encoding or signature-based parameter-transport handshake
This work	Web 3.0 / decentralized identity	Ephemeral ECEG encoding + signature/two-ephemeral-key ECEG mutual authentication and DID/VC binding, adapted to an explicit Web 3.0-oriented threat model	Forward secrecy added over [3]; evaluated analytically; formal verification and hardware benchmarking pending (Section X)

III. ELLIPTIC CURVE CRYPTOGRAPHY

Anonymity is an essential requirement for authentication in open networks, especially where the privacy of the interacting parties must be protected. Early designs for mutual authentication in public-key cryptosystems such as RSA and ElGamal relied on resource-intensive modular exponentiation and a separate infrastructure for certificate management. Fig. 1 illustrates how steeply this modular-exponentiation cost grows with key size under a naive-multiplication model. This combination is difficult to justify on devices with limited processing power, memory, or battery capacity, which motivates the use of Elliptic Curve Cryptography as a lighter-weight alternative with an equivalent security guarantee.

An elliptic curve over a finite field $\mathbb{F}_p$ is formally the set of points (x, y) that satisfy $y^2 = x^3 + ax + b$, together with a distinguished point at infinity that serves as the identity element. The points on the curve form an Abelian group under a geometrically defined addition rule. In this group, scalar multiplication is repeated addition of a base point G to itself: $k \cdot G = G + G + \dots + G$ (k times). Fig. 2 plots the real points of a representative curve of this form to illustrate the group law geometrically.

The security of ECC rests on the computational hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP): given

the base point G and a point $Q = d \cdot G$, computing the scalar d is believed to require time exponential in the bit-length of the curve's order for well-chosen curves, with respect to the best algorithms currently known. This asymmetry easy to compute Q from d , hard to invert lets ECC bind a private key to a publicly verifiable value without disclosing the key itself. In this work, the elliptic-curve key pair is generated as follows.

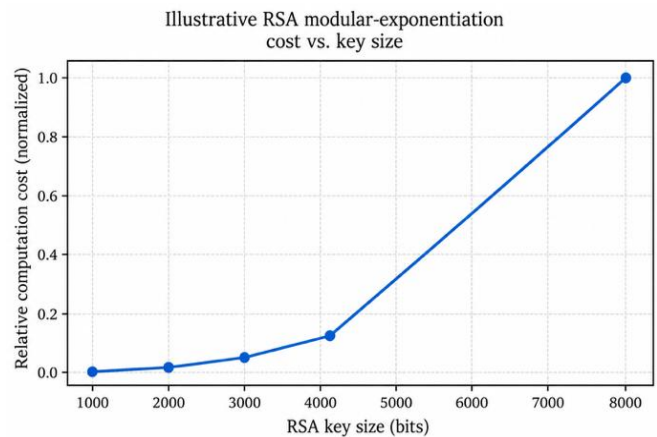


Fig. 1. Illustrative relative computation cost of RSA modular exponentiation as key size grows (naive-multiplication asymptotic model; not a measured benchmark).

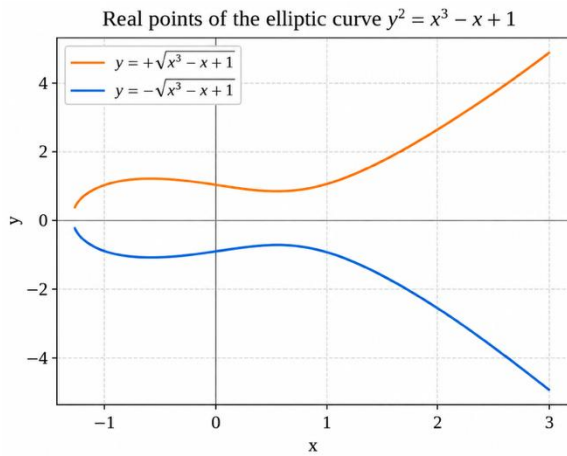


Fig. 2. Real-plane plot of the elliptic curve $y^2 = x^3 - x + 1$ (illustrative example of the group law; not the deployment curve, which is secp256k1).

Private key (d): a randomly chosen integer in the interval $[1, n - 1]$, where n is the order of the base point G .

Public key (Q): computed as $Q = d \cdot G$, a point on the curve.

The Elliptic Curve Integrated Encryption Scheme (ECIES) is a widely used hybrid construction that combines an ephemeral ECC key-agreement step with symmetric encryption and authentication. It is introduced here only as background context for the general pattern of ephemeral-key-plus-transport constructions; the protocol proposed in Section V is an ElGamal-based (ECG) construction, not an implementation of ECIES itself. Fig. 3 compares the same RSA/ECC asymptotic costs at equal operand length, complementing the key-size-equivalent comparison of Fig. 4.

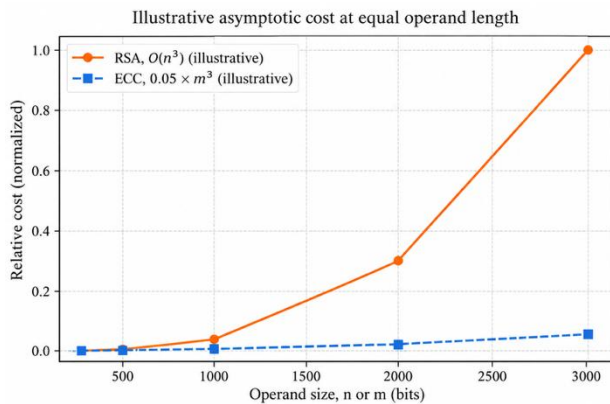


Fig. 3. Illustrative asymptotic cost of RSA and ECC at equal operand length (naive-multiplication model).

Encryption is performed by the sender using the recipient's public key Q , as follows.

- The sender generates an ephemeral private key r and the corresponding ephemeral public key $R = r \cdot G$.
- The sender computes the shared point $S = r \cdot Q$ and derives a shared secret byte string from it, typically the x -coordinate of S .

- A key derivation function (KDF) is applied to the shared secret to produce symmetric keys: K_{enc} for encryption and K_{mac} for integrity.
- The plaintext is encrypted with an authenticated symmetric cipher, such as AES-GCM or AES-CBC combined with HMAC, using K_{enc} .
- A message authentication code is computed over the ciphertext and any associated header data using K_{mac} .
- The tuple (R , ciphertext, MAC) is transmitted to the recipient.

Decryption, performed by the recipient using the private key d , mirrors this process.

- The recipient computes the shared point $S' = d \cdot R$.
- K_{enc} and K_{mac} are derived from S' using the same KDF and parameters used by the sender.
- The MAC is verified; the protocol aborts if verification fails.
- The ciphertext is decrypted with K_{enc} to recover the plaintext.

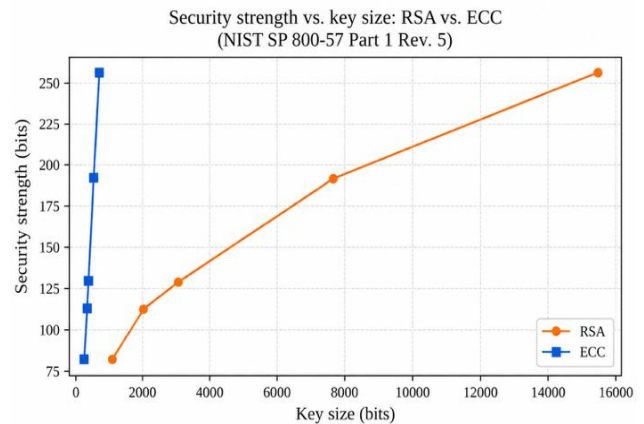


Fig. 4. Security strength versus key size for RSA and ECC, following NIST SP 800-57 Part 1 Rev. 5 [31].

As Fig. 4 illustrates, the security level obtainable per bit of key length is markedly higher for ECC than for RSA: a 256-bit ECC key (curve P-256, used here only as the NIST reference point for security-strength equivalence – the proposed protocol itself uses secp256k1, Section V-A) offers a security level comparable to a 3072-bit RSA key, both corresponding to a 128-bit security strength. This disparity widens further at higher security levels: at the 256-bit security level, a curve with a field size of 512 bits or more (e.g., P-521) is comparable to an RSA key of roughly 15360 bits, which is the central efficiency argument for adopting ECC on the resource-constrained devices considered in this study, and is analyzed quantitatively in Section VII. This comparison is a purely classical one: it assumes an adversary restricted to conventional computation. Section IX revisits the comparison under a quantum adversary, where the picture changes substantially, and situates both curves against NIST's current post-quantum migration guidance [32].

IV. THREAT MODEL AND SECURITY REQUIREMENTS

The proposed framework is evaluated under a Dolev–Yao-style adversary model [33], in which the attacker fully controls the communication channel: messages can be observed, intercepted, modified, replayed, or forged, but the attacker cannot break the underlying cryptographic hard problem; that is, the attacker cannot efficiently solve the ECDLP for the deployed curve parameters, nor invert the hash function used in the protocol. The attacker does not have access to the long-term private keys of legitimate parties and cannot read a device’s protected key storage. This classical Dolev–Yao adversary is consistent with the attack surface reported in a recent systematic review of Web 3.0 cybersecurity incidents [34], which documents network-level interception, replay, and impersonation alongside Sybil attacks and phishing as recurring threats to decentralized systems. A quantum-capable adversary, who could in principle solve the ECDLP efficiently, is outside this classical threat model and is treated separately in Section IX. Under the classical model, the framework is designed to satisfy the following security requirements, which also structure the analysis in Section VI.

1) *Mutual authentication*: Both communicating entities should be able to verify each other’s identity before any application data is exchanged.

2) *Confidentiality*: The character mapping and any plaintext exchanged should be unreadable to any party other than the intended recipient.

3) *Integrity*: any modification of the exchanged parameters or of the ciphertext during transmission must be detectable.

4) *Resistance to replay*: protocol messages captured in a previous session should not be reusable to impersonate a legitimate party in a new session.

5) *Resistance to man-in-the-middle attacks*: an adversary positioned between the two parties should not be able to establish independent, undetected sessions with each, provided the parties’ long-term public keys are authentically bound to their claimed identities (Section VI-B); absent that binding, this requirement reduces to key-possession authentication rather than identity authentication (Section VI-B).

6) *Pseudonymity and unlinkability*: the protocol should not leak information that lets an observer link protocol messages to a real-world identity beyond what the parties intentionally disclose (pseudonymity), and, more strongly, should not let an observer tell whether two sessions involve the same party (unlinkability). As Section VI-E makes precise, this protocol targets pseudonymity by design — protocol messages carry curve points and digests, not identity attributes — but does not by itself provide unlinkability, since the long-term public keys Q_A and Q_B are reused, and hence observable, across sessions.

7) *Ephemeral-session freshness*: the encoding parameters used to authenticate each session should be freshly generated and distinct from those of any other session, rather than static, reusable secrets.

8) *Forward secrecy*: later compromise of a party’s long-term private key should not allow an adversary to recover the

encoding parameters, and hence the message-key material, of previously completed sessions.

9) *Public-key-to-identity binding*: because the protocol targets parties with no common certificate authority, an explicit external mechanism (Section VI-B) must bind each long-term public key to its claimed identity; absent that binding, mutual authentication establishes only that a peer holds a particular private key, not that the key belongs to a particular real-world or wallet identity.

Sections IV and X already draw several of the distinctions above individually; Table III collects them into a single in-scope/out-of-scope boundary for the Web 3.0-specific adversary conditions raised in the discussion of related work (Section II).

TABLE III. IN-SCOPE AND OUT-OF-SCOPE ADVERSARY CAPABILITIES

Adversary Class	Treatment in This Study
Network (Dolev–Yao)	In scope: full control of the channel (intercept, modify, replay, inject, block); cannot break ECDLP or invert the hash (Section IV).
Endpoint/wallet compromise (malicious wallet software, key exfiltration)	Out of scope for Sections VI-A – VI-D; partially mitigated by explicit ephemeral-key erasure (Section VI-D) but not modeled as an active capability.
Signing-request substitution / malicious dApps	Out of scope; assumed handled by wallet-level user-interface security, not by this protocol.
Phishing / social engineering	Out of scope (Section X).
Smart-contract vulnerabilities (reentrancy, logic errors)	Out of scope; an application-layer concern distinct from the authentication protocol.
Public blockchain linkability of long-term keys	Partially in scope: acknowledged as a residual pseudonymity limitation (Section VI-E), not defended against as a network-adversary capability.
Quantum adversary (Shor’s algorithm)	Out of scope for the classical analysis of Sections IV - VIII; treated separately in Section IX.

V. PROPOSED AUTHENTICATION FRAMEWORK

The main operation in the proposed framework is elliptic-curve scalar multiplication. In a decentralized Web 3.0 network, two entities must authenticate each other before any data exchange takes place, and the authentication step itself must be resilient to active attacks such as man-in-the-middle interception and eavesdropping-based replay. The framework presented below is designed for constrained devices; whether it is lightweight in practice remains an empirical question requiring implementation-level benchmarking.

A. System Parameters and Key Generation

The proposed framework targets secp256k1 as its deployment curve, consistent with its use throughout the Web3/blockchain wallet ecosystem discussed in Section IX; the standardized domain parameters for secp256k1 are used directly rather than being restated here. In general, the elliptic curve E over a finite prime field is defined by the domain parameters $T = (p, a, b, G, n, h)$, where p describes the underlying finite field, a and b define the shape of the curve, $G = (x_G, y_G)$ is the base

point, n is the order of G (the smallest positive integer such that $n \cdot G$ is the point at infinity), and h is the cofactor, $h = \#E(\mathbb{F}_p)/n$, where $\#E(\mathbb{F}_p)$ is the total number of points on E over $\mathbb{F}_p$. The curve used in this work is cubic and is given by

$$y^2 + a_1xy + a_3y \equiv x^3 + a_2x^2 + a_4x + a_6 \pmod{p} \quad (1)$$

For $p > 3$ prime, the curve in Weierstrass form is

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (2)$$

where, $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$. This condition ensures that the curve is nonsingular. Each participant generates a private key k_A as a random integer and computes the corresponding public key $Q_A = k_A \cdot G$. The other participant similarly generates k_B and $Q_B = k_B \cdot G$, and the two public keys are exchanged over the (possibly adversarial) channel.

In addition to their long-term key pairs, B generates a fresh, single-use ephemeral key pair for each session: a scalar r_b and the corresponding point $R_b = r_b \cdot G$. R_b is published alongside Q_B during this same parameter-exchange step (its binding to B is subject to the same external-identity assumption as Q_B itself, Section VI-B). This per-session ephemeral point, rather than the long-term Q_B , is what the ECEG transport of Section V-C encrypts to, which is what gives the protocol forward secrecy (Section VI-D); B discards r_b once it has been used to recover a single message, so a later compromise of B 's long-term key k_B does not expose past sessions' encoding parameters. In an interactive deployment, B generates R_b on demand for each new session; in a deployment where B must be able to receive a first message while offline, B instead publishes a small batch of one-time R_b values in advance and discards each one after its first use, following the same one-time-prekey pattern used in Signal's X3DH handshake.

This study adopts Schnorr signatures over secp256k1 (BIP-340 [44]) for $\text{Sign}_k(\cdot)$ in Section V-C, rather than leaving the signature primitive unspecified: SHA-256 as the hash function, tagged hashes for domain separation between the different hash uses in the protocol, and a 64-byte fixed-size signature encoding. Nonces are generated deterministically from the signing key and message following BIP-340's synthetic-nonce procedure [44] (optionally mixed with fresh auxiliary randomness for side-channel hardening), the same deterministic-nonce principle standardized for ECDSA in RFC 6979 [45], which removes the catastrophic key-exposure risk that a biased or repeated per-signature random nonce creates in ECDSA-style schemes, and sidesteps ECDSA's $(r, -s)$ malleability without requiring a separate canonical-encoding rule. All received points — Q_A, Q_B, R_b , and any signature-carried commitment — are validated on receipt: each is checked to satisfy the curve equation, to differ from the point at infinity, and to lie in the correct order- n subgroup, following standard curve-point validation practice (SEC 1 [46]; NIST SP 800-186 [47]), before being used in any further computation.

B. Ephemeral ECEG Character Encoding

Before encryption, the plaintext message is processed using an Elliptic Curve ElGamal (ECEG) style encoding, extending the ephemeral mapping approach of [2]. Rather than exchanging a fixed lookup table between the 8-bit byte alphabet and points

on the curve, each character is mapped, character by character, to a curve point using the scalar-multiplication relation

$$E_k = (\alpha \cdot k + \beta) \cdot G \quad (3)$$

where, k denotes the byte value (0–255) of the character being encoded, and α and β are two session-specific byte values jointly agreed upon by the sender and receiver during the encoding phase described below. Requiring a raw curve-point coordinate to already fall in (0, 255) is not usable on a 256-bit curve: both coordinates of a random point land in that range with probability on the order of 2^{-496} , making rejection sampling computationally infeasible. The sender therefore derives α and β from an ephemeral point with a hash-based key-derivation step, then re-encodes them as a curve point before transport (Section V-C requires a curve point, not an arbitrary integer pair), as follows.

Step 1: pick a random integer i .

Step 2: compute the ephemeral elliptic-curve point

$S = i \cdot G = (X_S, Y_S)$, where G is the generator point of the selected curve.

Step 3: derive the encoding parameters with a key-derivation function applied to the point's coordinates: $(\alpha, \beta) = \text{Trunc}_{16}(\text{hash}(X_S \parallel Y_S))$, i.e., take the first 16 bits of $\text{hash}(X_S \parallel Y_S)$ and split them into two bytes α and β . If $\alpha = 0$, the sender discards the derived pair and regenerates i until $\alpha \neq 0$, ensuring that the byte-to-point mapping remains injective over the 256-byte alphabet.

Step 4: combine the two bytes into a single 16-bit integer $m = 256 \cdot \alpha + \beta$, and encode m as a curve point $M = \text{EncodeToPoint}(m)$ using the integer-to-point technique of Koblitz [41]: for $j = 0, 1, 2, \dots$, form the candidate x -coordinate $x = m \cdot K + j$ for a fixed public padding constant K chosen so that $m \cdot K$ stays far below p , and stop at the first j for which $x^3 + ax + b$ is a quadratic residue mod p , setting y to a square root of $x^3 + ax + b \pmod{p}$. Decoding reverses this: recover $m = \lfloor x/K \rfloor$ from M 's x -coordinate, then split it back into $\alpha = \lfloor m/256 \rfloor$ and $\beta = m \pmod{256}$. Because roughly half of the field elements are quadratic residues, encoding succeeds within the first few values of j with overwhelming probability.

Only M — not α and β directly — is ever placed on the curve or on the channel; α and β themselves remain ordinary 16-bit integers used solely to build M , to form the signature input used in Section V-C, and to build the byte-to-curve-point mapping table. Because that reduction is only 16 bits wide, α and β are not by themselves a cryptographically strong secret — at that size they would be brute-forceable if ever exposed on their own. Their practical protection instead comes from the fact that they are only ever transported inside the authenticated ECEG ciphertext M of Section V-C, never sent in the clear; this trade-off is stated here explicitly rather than left implicit.

C. Signature- and Two-Ephemeral-Key ECEG-Based Mutual Authentication

Once α and β are generated and encoded as the curve point M of Section V-B, they must be transported to B and confirmed by both parties before either side uses them, without ever placing

(α, β) or M on the channel unencrypted, and with each party's authorship of its own messages bound to its long-term key pair rather than to any state carried over from a previous session. Every symbol below is defined once and used consistently with Fig. 5.

A picks a fresh ephemeral scalar r and forms the ECEG transport of M under B's per-session ephemeral point R_b (Section V-A) rather than under B's long-term Q_B : $C_1 = r \cdot G$, $C_2 = M + r \cdot R_b$, $C = (C_1, C_2)$. A also generates a fresh, single-use session nonce N_a and computes a signature $\sigma_a = \text{Sign}_{\{k_A\}}(\text{hash}(C \parallel N_a \parallel Q_A \parallel Q_B \parallel R_b))$ under its own long-term private key — binding the specific R_b used into the signed transcript so that a later substitution of R_b during the key-exchange step cannot go undetected — then sends (C, N_a, σ_a) to B. B first checks that N_a has not been seen before, rejecting the message as a replay if it has, and verifies σ_a against Q_A ; if either check fails, B aborts before doing any further work on C , in particular before performing the ECEG decoding.

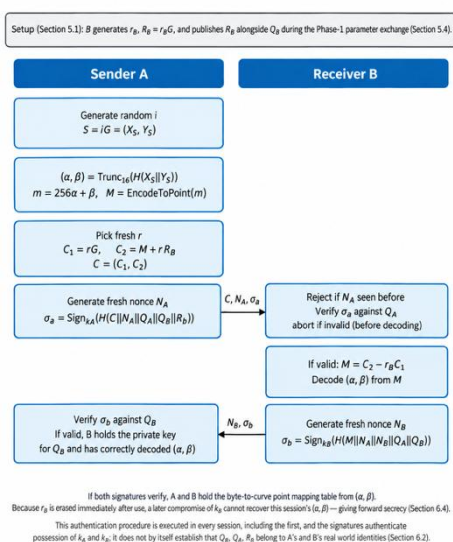


Fig. 5. Signature- and nonce-based mutual authentication and encoding parameter transport between sender A and receiver B.

If both checks succeed, B recovers $M = C_2 - r_b \cdot C_1$ using the ephemeral scalar r_b generated for this session (Section V-I), then immediately erases r_b — the step that gives the protocol forward secrecy (Section VI-D), since nothing derivable from B's long-term key k_B can later reconstruct r_b or M — and decodes (α, β) from M as in Section V-B. B then generates its own fresh nonce N_b and computes $\sigma_b = \text{Sign}_{\{k_B\}}(\text{hash}(M \parallel N_a \parallel N_b \parallel Q_A \parallel Q_B))$, and sends (N_b, σ_b) to A. A verifies σ_b against Q_B ; if verification succeeds, A concludes that whoever holds the private key for Q_B received and correctly decoded (α, β) , and both sides finalize the byte-to-curve-point mapping table used for subsequent character-by-character encryption.

This design authenticates each party as the holder of the long-term private key corresponding to its public key, in every session including the first: unlike a design that relies on shared history from a previous session, a valid signature cannot be produced without k_A or k_B regardless of whether A and B have interacted before, so it does not have the bootstrapping gap that a history-dependent check would have on a first encounter. It

does not by itself establish that Q_A or Q_B belongs to the real-world identity it is claimed to represent; that binding is the external identity layer discussed in Section VI-B, but it does rule out an adversary who has not compromised either long-term key from completing the handshake under a false claim to already hold one of these keys. Freshness and replay-resistance rest on the per-session nonces N_a, N_b , which B and A each check against a short-lived record of recently used values, rather than on synchronized state carried over from a previous session; this also avoids the resynchronization problems (e.g., after a device restart or an interrupted session) that a previous-session-dependent design would otherwise create.

Both nonces are generated as 128-bit values from a cryptographically secure random-number generator, giving a collision probability across k sessions on the order of $k(k-1)/2 \times 2^{-128}$ by the birthday bound — negligible for any realistic session count. Each party retains a replay cache of nonces it has already seen and evicts each entry a fixed window after first recording it (tens of seconds to a few minutes, matching the round-trip tolerance conventionally used for challenge-response freshness, e.g., in Kerberos), so the cache size stays bounded by the message-arrival rate rather than growing without bound over the life of a long-lived identity; a replayed message that arrives after its originating entry has already been evicted is accepted as if fresh only if the underlying nonce happens to collide with a genuinely new one, which the birthday bound above rules out. After a restart, a party simply begins from an empty cache; a validator or wallet backend spanning multiple nodes shares this cache (or replicates it with bounded staleness) across nodes so that a replay accepted at one node is rejected at the others.

D. Protocol Execution

The protocol runs in three phases, as shown in Fig. 6. First, the shared elliptic-curve domain parameters are agreed upon by sender A and receiver B; long-term public keys Q_A and Q_B are exchanged, and B additionally generates and publishes a fresh, session-specific ephemeral point R_b (Section V-A); the binding of Q_A, Q_B , and R_b to A's and B's real-world identities is treated as an external assumption rather than established by this exchange itself, as discussed in Section VI-B. Second, the ephemeral encoding procedure of Section V-B is performed to produce α and β , which are transported under R_b rather than under B's long-term key and mutually confirmed using the signature- and nonce-based handshake of Section V-C; this step proves that each party holds the private key for its claimed long-term public key, in addition to confirming that B correctly recovered (α, β) , and at the end of this stage both parties share the same session-specific encoding parameters, B has erased r_b (Section VI-D), and neither has exchanged the full byte-to-curve-point mapping table itself. Third, A encrypts the message character by character with the derived mapping and sends it to B as ciphertext; B has already recovered and decoded (α, β) in the previous phase and uses the resulting byte-to-curve-point mapping table to recover the original plaintext character by character.

This design reduces the risk of replay attacks, since authentication in each session depends on encoding parameters that combine fresh randomness with values distinct from any previous session; resistance to man-in-the-middle substitution

additionally requires the public-key binding discussed in Section VI-B. Even if an adversary intercepts the ciphertext components during transmission, recovering B's private key k_B , and, separately, recovering B's per-session ephemeral scalar r_b , from the intercepted values is computationally infeasible under the ECDLP hardness assumption discussed in Section VI, since the ephemeral points $i \cdot G$, $R_b = r_b \cdot G$, and the ciphertext component $C_1 = r \cdot G$ reveal only curve points, not the underlying scalars i , r , r_b , or k_B . The session's encoding attributes α and β are a different case, addressed in Section VI-A: at only 16 bits, they are not ECDLP-hard on their own, and are protected instead by never being sent except inside the encrypted transport M of Section V-C.

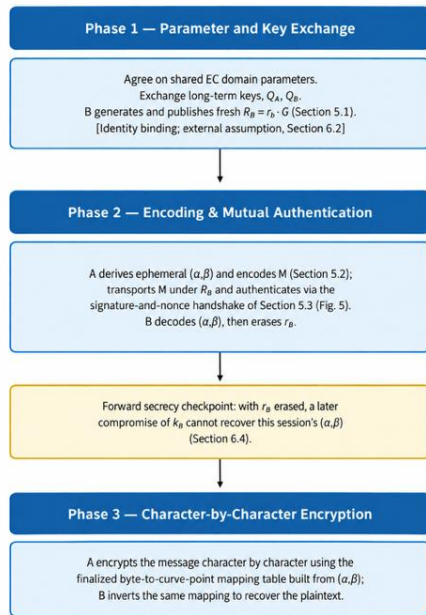


Fig. 6. End-to-end execution flow of the proposed ECEG authentication framework, including per-session signature verification.

VI. SECURITY ANALYSIS

This section gives an informal, assumption-based analysis of the protocol with respect to each requirement identified in Section IV. The analysis rests on the Dolev–Yao adversary model [33] and on the standard assumptions that the ECDLP is computationally hard for the chosen curve parameters and that the ECEG transport is IND-CPA secure under the relevant group assumption (e.g., DDH), while the hash function used in the signature inputs is preimage- and collision-resistant. Section VI-F then specifies a concrete plan for replacing this informal argument with a machine-checked one.

A. Resistance to Eavesdropping

An adversary eavesdropping on the channel sees only public curve points — e.g., Q_A , Q_B , ciphertext components, and hash/signature outputs. Recovering the long-term or ephemeral private scalars k_A , k_B , r , r_b , or i from these observations requires solving the ECDLP, which is assumed computationally infeasible for curves of appropriate size. The 16-bit pair (α, β) is a different matter: at only 2^{16} possible values it is not itself ECDLP-hard, and would be recoverable by exhaustive search from even a single known plaintext/ciphertext pair under a fixed

encryption key. Its confidentiality therefore rests entirely on the semantic security (IND-CPA) of the ECEG transport that carries it — specifically, on the pair (r, R_b) never repeating across sessions (Sections V-A, V-C) — rather than on the small parameter space itself being hard to search; framed differently, the 16-bit encoding provides no obfuscation-level security on its own and should not be read as doing so. Because r and R_b are both freshly generated per session and never reused, each transport instance is independently randomized, so an eavesdropper who cannot invert the transport does not obtain (α, β) directly, and passive monitoring does not reveal the mapping used for the actual message.

B. Resistance to Man-in-the-Middle Attacks

The signature step of Section V-C means a man-in-the-middle adversary cannot forge a valid session on behalf of A or B without the corresponding long-term private key k_A or k_B : substituting tampered values for C , N_a , or N_b causes signature verification to fail, and the session aborts before either party acts on the forged data. What this check cannot do is establish that Q_A or Q_B belongs to the real-world identity it is claimed to represent in the first place. If an adversary presents its own key pair as Q_B at the very start of the exchange — before any signature has been exchanged — and A has no independent way to confirm that this Q_B is genuinely B's, A can complete a validly signed handshake with the adversary under the adversary's own key. This residual gap is a binding problem between identity and public key, not a weakness in the signature or key-transport steps themselves; closing it requires the external identity layer described next.

A concrete instantiation of this external identity layer, rather than an abstract placeholder, is available using the W3C Decentralized Identifier (DID) Core [42] and Verifiable Credential Data Model [43] standards, and integrates directly into the parameter exchange of Section V-D rather than sitting alongside it as a separate system. Each party publishes its long-term public key as the verification method of a DID document — for instance, under the did: key method, which encodes the public key directly in the identifier and needs no registry lookup, or under a ledger-anchored method such as did:ethr when linkage to an existing wallet address is required. A credential issuer that the verifying party already trusts (a prior counterparty, an organization, or any other chosen trust anchor) signs a verifiable credential attesting that a given DID controls a given real-world or wallet identity. During the exchange of Q_A , Q_B , and R_b in Section V-D's first phase, each party additionally presents its DID document and this credential; the peer checks the credential's issuer signature and confirms that the DID document's verification method matches the presented public key before proceeding to the signature-and-key-transport steps of Section V-C. This binds the specific public key used later in the handshake to a checked identity claim up front, rather than accepting whichever key arrives first, closing the substitution gap described above; the pattern of folding a DID/VC presentation directly into an existing handshake, rather than treating it as a separate protocol layer, follows the approach recently described for integrating DIDs and verifiable credentials into the TLS handshake [50].

This mechanism does not remove every trust assumption — it shifts the question from “is this public key really B's” to “is

this credential issuer trustworthy and is its signature currently valid,” the same reduction any certificate- or credential-based scheme makes — but the resulting trust anchor is a credential issuer or prior relationship chosen by the parties themselves rather than a centralized certificate authority, consistent with the self-sovereign-identity setting motivating this study (Section I). Formally analyzing the combined system — the DID/VC presentation step together with the signature-and-key-transport handshake of Section V-C, as a single verified protocol rather than as two independently analyzed components — is identified as follow-up work in Section 10.

C. Resistance to Replay Attacks

Each session includes a fresh, single-use nonce (N_a from A, N_b from B, Section V-C), and both the signature and the confirmation checks are computed over values that include these nonces. An adversary who records a complete protocol transcript from one session and replays it in a later session presents a nonce that the recipient has already seen, so B’s nonce check (or A’s, for the return message) rejects it before any signature verification or ECEG decoding is attempted. Because this check happens first, a replayed message is cheap to reject; a flood of freshly-constructed but invalidly-signed messages is more expensive, since it still costs a signature verification per message, and defending against that kind of resource-exhaustion load is outside the assumption-based analysis in this section — rate limiting or a lightweight pre-filter would be needed in a production deployment (Section X).

D. Forward Secrecy and Key Compromise

Because α , β , the ephemeral scalar i used to derive them, and the per-session ephemeral scalars r and r_b (Section V-A, V-C) are all generated freshly for each session, none of them is a deterministic function of the long-term private keys k_A or k_B alone, so a device that is captured while offline does not thereby leak the encoding parameters of past sessions from its stored long-term key material. Unlike the static-key ECEG transport used in [3] and in an earlier draft of this protocol, the transport of Section V-C encrypts (α, β) under B’s per-session ephemeral point $R_b = r_b \cdot G$ rather than under B’s static Q_B , and B erases r_b immediately after using it to recover a single message. Consequently, an adversary who has recorded C_1 and C_2 from a past session and later obtains B’s long-term private key k_B gains no advantage in reconstructing that session’s r_b , since r_b is chosen independently of k_B and is not recoverable from R_b under the ECDLP hardness assumption; without r_b , the adversary cannot compute $r_b \cdot C_1$ and so cannot recover (α, β) from the recorded transcript. The protocol therefore provides forward secrecy against later compromise of a long-term private key, conditional on B genuinely erasing r_b after each session’s single use — an explicit key-erasure assumption stated here rather than left implicit, and one that a production implementation must enforce (e.g., by zeroing the scalar’s memory immediately after the single point multiplication that consumes it) for the property to hold in practice. This conditional guarantee, together with its formal counterpart in the verification plan of Section VI-F, replaces the unconditional absence of forward secrecy that an earlier, static-key version of this design would have inherited from [3]. The same forward-secrecy guarantee also requires A to erase its session-specific scalars i and r after the session completes; retaining either scalar

would allow recovery of past session material from recorded data. Accordingly, the erasure condition is a protocol-state requirement for both parties, not only for B.

E. Anonymity and Residual Risks

The protocol limits what an on-path observer can learn about the identities involved in a given exchange, since the exchanged values are curve points and hash digests rather than identity attributes. Full anonymity in the sense used by self-sovereign identity systems, however, also depends on how public keys are bound to real-world identities at the application layer, for instance, through the decentralized-identifier/verifiable-credential mechanism integrated into the exchange in Section VI-B, which establishes identity binding but does not by itself change what an observer of the resulting public keys can infer. This distinction matters specifically for the Web 3.0 setting targeted here (Section II-D): unlike a provisioned IoT device with a fixed operator-known identity, a Web 3.0 wallet’s long-term public key may itself be linkable across sessions through blockchain analysis even if this protocol’s ephemeral parameters are not. Reasoning about that residual linkability is an application layer concern outside this study’s scope, but it should be addressed jointly with the SSI layer in any deployment. The analysis in this section is informal rather than a machine-checked proof; Section VI-F identifies automated protocol verification as an important next step before the protocol is considered ready for production deployment.

F. Formal Verification Specification and Reproducibility Artifact

Sections VI-A – VI-E give an informal, assumption-based argument. To move beyond that argument, this subsection specifies a concrete formal-verification model and its required security queries, precisely enough to be encoded in Tamarin [54] as a supplementary reproducibility artifact without further design decisions. No such encoding has yet been produced or executed, and accordingly no machine-checked verification result is claimed in this study; preparing and running that encoding is identified as the immediate next step in Section X.

1) *Adversary model*: The standard Dolev–Yao network adversary of Section IV, in full control of the network and able to intercept, block, modify, replay, and inject messages, and able to compromise any number of protocol sessions but not the internal state of an honest party during a session, nor the long-term private keys k_A , k_B .

2) *Tool and primitives*: The specification below abstracts the two-ephemeral-key ECEG transport using a symbolic shared-point relation, rather than attempting to encode concrete secp256k1 point arithmetic. It treats the hash function as one-way and digital signatures through the verifier’s signature primitives. Because the protocol no longer includes an XOR operation, the resulting model would not require XOR equational reasoning. Tamarin [54] is used as the reproducibility target; Scyther [25] remains relevant as the tool used to verify the closest prior art [3].

3) *Protocol encoding*: The roles of A and B are encoded as two processes exchanging exactly the messages of Section V-D, with fresh session-specific values (i , r , r_b) declared using

the tool's freshness construct so that each session uses independent randomness, while the long-term private keys k_A , k_B remain persistent across sessions, and with r_b scoped so the process has no further access to it once it has been used for a single decryption, modeling the explicit erasure required by Section VI-D.

Queries.

- Secrecy of α , β , r_b , k_A , and k_B : the adversary process must never derive these values, regardless of how many sessions run concurrently.
- Injective agreement (correspondence) between A and B on the tuple of session-specific values (α , β , R_b , and the session identifier), so that every session in which A completes with B corresponds to a distinct completed session of B with A — the standard formalization of both mutual authentication and replay resistance, with R_b included so that a substitution of the ephemeral point between phases is itself caught as a disagreement.
- Forward secrecy under long-term key compromise: secrecy of α and β from an already-completed session even after both parties' long-term private keys are

revealed to the adversary once the session has ended, directly encoding the erasure-conditional guarantee argued informally in Section VI-D, rather than treated as an optional strengthening as it would have been for the earlier static-key design.

- Expected deliverable: A machine-checked verdict verified, or a concrete attack trace for each query above, together with the tool version and the model source, to be released as a supplementary artifact. Executing this plan and reporting its outcome, whichever it is, is identified in Section X as the immediate next step before the protocol can be considered validated beyond the assumption-based argument given in Sections VI-A – VI-E.

G. Security-Claim Summary

Table IV records, for each property discussed in Sections 6.1–6.6, the strongest claim this study makes and the assumption or limitation on which it rests. It is intentionally conservative: properties that depend on external identity binding or on an unexecuted validation step are marked conditional or not claimed rather than folded into an unqualified security statement.

TABLE IV. SECURITY-CLAIM SUMMARY FOR THE PROTOCOL OF SECTION V

Property	Status	Basis and limitation
Key-possession authentication	Claimed	Schnorr signatures over the current transcript (Section V-C); assumes uncompromised long-term keys and point validation (Section V-A).
Identity authentication	Conditional	Holds only after the DID/VC binding check of Section VI-B; depends on the chosen credential issuer / DID method.
MITM resistance	Conditional	Signatures block tampering after key selection; initial key substitution is prevented only by the identity binding of Section VI-B.
Replay resistance	Claimed	Fresh 128-bit nonces checked against a bounded replay cache (Section V-C); requires the stated cache window.
Confidentiality of (α, β)	Claimed	Rests on IND-CPA security of the ECEG transport with fresh (r , R_b) per session (Section VI-A); the 16-bit values are not secrets in themselves.
Forward secrecy	Conditional	Requires erasure of i and r by A and r_b by B after the session (Section VI-D).
Pseudonymity	Claimed	Messages carry curve points and digests, not identity attributes (Section VI-E).
Unlinkability/anonymity	Not claimed	Long-term keys, DIDs, and wallet addresses remain linkable across sessions (Sections IV, VI-E).
Machine-checked verification	Not claimed	Verification plan specified (Section VI-F); no verifier run reported in this study.
Empirical lightweight performance	Not claimed	Analytical operation count only (Table VI); no latency, energy, memory, or throughput measurements performed.
Post-quantum security	Not claimed	Classical ECC only; migration path outlined in Section IX.

VII. COMPUTATIONAL COMPLEXITY ANALYSIS

The efficiency analysis in this section separates generic RSA/ECC complexity from the protocol-level costs of the proposed scheme. The generic comparison is intended to show the security-to-key-size relationship; it is not a substitute for implementation-level benchmarking.

A. RSA Modular Exponentiation

The main computation of RSA is the modular exponentiation $c = m^e \bmod N$. Its cost under repeated squaring depends on n , the bit length of the modulus N (an upper bound on the number of modular multiplications), and $M(n)$, the cost of a single big-

integer multiplication or modular reduction. The total cost of modular exponentiation is therefore roughly $O(M(n) \cdot n)$. School-book multiplication gives $M(n) = O(n^2)$ and hence a rough bound of $O(n^3)$; more advanced algorithms such as Karatsuba or FFT-based multiplication improve the constant, and the exponent on $M(n)$, but the dependence on n remains large, and n must typically be 2048 bits or more to reach an acceptable security level.

B. ECC Scalar Multiplication

The main computation in ECC is scalar multiplication $Q = k \cdot P$, realized through repeated point additions and doublings. Its cost depends on m , the bit length of the field elements used to

represent coordinates (the number of point operations is proportional to m), and $M(m)$, the cost of one field multiplication. This gives a total scalar-multiplication cost of order $O(M(m) \cdot m)$, a bound of similar form to $O(m^3)$ under naive field multiplication. The key difference is that m can be substantially smaller than n while still achieving the same security level, as Table V shows, which reduces key representation and can reduce the cost of corresponding public-key operations; actual runtime and energy benefits depend on implementation, hardware, and library choices and are not measured here. Values follow NIST SP 800-57 Part 1 Revision 5 [31].

TABLE V. COMPARABLE SECURITY LEVEL: ECC KEY SIZE VERSUS RSA KEY SIZE

Security Level (bits)	ECC Field Size (bits)	RSA Key Size (bits)
256	512+	15,360
192	384–511	7,680
128	256–383	3,072
112	224–255	2,048
80	160–223	1,024

As Table V and Fig. 7 show, the RSA key size is 3072 bits for a 128-bit security level, while the corresponding ECC key size is only 256 to 383 bits. The difference widens further at higher security levels: a 256-bit security level requires an RSA key of roughly 15,360 bits, versus an ECC key of 512 bits or more. Under the naive-multiplication model, both schemes have cost growing with the cube of the key length, but the ECC key length required grows much more slowly than the RSA key length required as the target security level increases. This supports the use of ECC where storage and communication overhead matter, but it does not establish lower end-to-end runtime for the complete protocol; that requires empirical measurement.

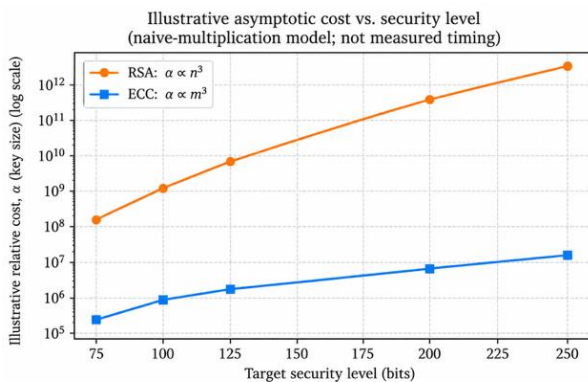


Fig. 7. Illustrative asymptotic cost of RSA and ECC as a function of target security level.

C. Protocol-Level Operation Count and Communication Cost

The comparison in Sections VII-A – VII-B evaluates generic RSA and ECC operations rather than the protocol proposed in Section V itself. Table VI instead counts, directly from the specification of Sections V-A – V-D, every EC scalar multiplication, signature operation, hash evaluation, message,

and byte of state that one completed session requires of each party.

TABLE VI. PER-SESSION OPERATION COUNT FOR THE PROTOCOL OF SECTION V

Operation (per completed session)	A	B
EC scalar multiplications, total (handshake, incl. Schnorr)	6	5
Byte-to-point table construction (one-time per session)	256	256
Signature generations	1	1
Signature verifications	1	1
Explicit SHA-256 hash/KDF evaluations (protocol)	2	1
Messages sent (handshake only)	1	1
Bytes transmitted (handshake only)	≈ 146 B	≈ 80 B
Long-term key state	$k_A + Q_B$	$k_B + \text{replay cache}$

The scalar-multiplication total breaks down as follows: for A, $i \cdot G$ (encoding, Section V-B) + $r \cdot G + r \cdot R_b$ (Section V-C) + 1 scalar multiplication for generating $\sigma_A + 2$ for verifying σ_B (Schnorr verification counted conservatively as two scalar multiplications in the unbatched verification equation) = 6. For B, $r_b \cdot G$ (Section V-A) + $r_b \cdot C_1$ (Section V-C) + 1 scalar multiplication for generating $\sigma_B + 2$ for verifying $\sigma_A = 5$. These totals therefore include the elliptic-curve work performed inside the Schnorr signature operations; the signature-generation and signature-verification columns in Table VI report those operations separately for clarity. An implementation using multi-scalar-multiplication techniques can reduce verification cost. The 256-entry byte-to-point table is built once per session and then reused for every character of the message by table lookup, so its construction cost is independent of message length L rather than growing with it. Persistent state is dominated by the 32-byte long-term private key on each side; B additionally holds a bounded replay cache (Section V-C). Table VI's message and byte counts cover only the cryptographic authentication phase and exclude Phase-1 domain/key exchange and DID/VC presentation overhead.

A separate communication cost, not included in the handshake byte count in Table VI, is the bandwidth expansion of the character-level ECEG encryption in Section V-D's third phase: each plaintext byte is represented on the wire as a full secp256k1 curve point, 33 bytes compressed or 65 bytes uncompressed, a $33\times$ to $65\times$ expansion over the one byte of plaintext it carries. A conventional hybrid alternative — using the already-established per-session shared point $r \cdot R_b$ (Section V-C) to key a symmetric authenticated cipher such as AES-GCM or ChaCha20-Poly1305 for the bulk message, as Section III describes for ECIES — would instead add only a small constant-size tag (typically 16 bytes) regardless of message length, with no per-character point encoding. This study retains character-level ECEG encoding because it is the specific mechanism inherited from [2,3] that the threat model and verification plan of Sections IV and VI-F are built around; a deployment that prioritizes bandwidth over continuity with that

prior design should instead use the hybrid AEAD construction for the bulk message while keeping the ECEG-over-two-ephemeral-key transport of Section V-C for the (α, β) handshake itself.

Published implementation-level benchmarks corroborate the direction, though not the exact magnitude, of the efficiency argument above: Hammi et al. [48] propose a lightweight ECC-based mutual authentication scheme for IoT and report it as computationally and communication-efficient relative to prior IoT authentication schemes on constrained hardware, in the same general direction as the asymptotic argument of Sections VII-A – VII-B, though that study does not benchmark against RSA directly; Li and Hu [49] report a fresh ECC-based IoT authentication and key agreement protocol reducing computational overhead by more than 37% and communication cost by more than 5% relative to prior schemes, using a per-session operation-count methodology similar in spirit to Table VI. Neither benchmark is of the protocol proposed in this study, so they are cited as external corroboration of the general ECC-efficiency direction rather than as a substitute for the hardware benchmarking identified as future work in Section X.

VIII. DISCUSSION

The analyses of Sections VI and VII lead to two complementary conclusions. From a security standpoint, tying authentication to ephemeral, session-unique parameters instead of static secrets, and grounding that design in the ECDLP hardness assumption, is intended to make eavesdropping and replay harder for an adversary limited to the Dolev–Yao model of Section IV, without requiring a common certificate authority; resistance to man-in-the-middle interception additionally depends on the public-key binding discussed in Section VI-B. From an efficiency standpoint, the asymptotic and key-size comparisons in Section VII support the well-known ECC security-to-key-size advantage, though actual end-to-end performance depends on implementation, hardware, and library choices not measured here.

These security and representation-size properties are complementary rather than independent: the relatively small ECC keys and points make per-session ephemeral operations practical to consider on constrained devices, while the actual cost of the proposed ECEG encoding and signature-based handshake remains an empirical question. The security properties described in Section VI therefore should not be inferred from key-size ratios alone.

As discussed in Section II-B, the framework’s relationship to the closest prior art [3] is best understood as an adaptation at two levels rather than a wholesale new cryptographic mechanism: both protocols inherit the same ephemeral ECEG character-encoding step from [2], rather than a static, exchanged lookup table or a static identity-bound key exchange as in several of the RFID and smart-card schemes reviewed in Section II [4,5,9,10]. Beyond that shared encoding step, however, the two protocols diverge: [3] transports the encoding parameters under a static recipient key and confirms them with a hash-and-XOR value chained to the previous session, whereas this study authenticates each party with digital signatures over its long-term key and transports the encoding parameters under a fresh per-session ephemeral key (Section V-C), which is what closes

the forward-secrecy gap identified in Table I. What this study adds on top of that protocol-level change is the explicit threat model, the pseudonymity/unlinkability analysis, the concrete DID/VC identity-binding mechanism (Section VI-B), and the verification plan needed to argue that the resulting design is appropriate specifically for Web 3.0 peer-to-peer interactions, where the two interacting parties may have no previously established trust relationship and no common certificate authority.

IX. POST-QUANTUM OUTLOOK

The security analysis in Section VI and the efficiency case in Section VII both rest on a classical adversary: one that cannot efficiently solve the Elliptic Curve Discrete Logarithm Problem (ECDLP). A quantum adversary running Shor’s algorithm can, in principle, solve the ECDLP in polynomial time, and 2025–2026 has seen rapid, concrete progress in estimating exactly how many quantum resources that would take for the curve sizes relevant to Web 3.0. This section situates the proposed framework against that fast-moving picture, states plainly what it implies for the protocol proposed here, and outlines a migration path consistent with current NIST guidance.

A. The Narrowing Quantum Resource Gap for ECDLP

TABLE VII. SUCCESSIVE 2020–2026 ESTIMATES OF THE LOGICAL QUBITS NEEDED TO SOLVE THE 256-BIT ELLIPTIC-CURVE DISCRETE LOGARITHM PROBLEM ON A PRIME-FIELD CURVE

Source	Year	Venue / Status	Logical Qubits (256-bit curve)
Häner et al. [PQCrypto 2020]	2020	Peer-reviewed (PQCrypto 2020)	≈ 2124
[35]	2026	Peer-reviewed (PRX Quantum 2026)	$< 1200\text{--}1450$
[36]	2026	Peer-reviewed (EUROCRYPT 2026)	≈ 1193
[37]	2026	Preprint (arXiv)	≈ 1333
Häner et al. [51]	2026	Preprint (IACR ePrint / arXiv)	≈ 1450 (19,397 physical qubits; ~ 25.7 days)

Table VII summarizes successive 2026 reductions in the estimated number of logical qubits needed to solve the 256-bit ECDLP on a prime-field curve such as secp256k1, the curve family underlying most Web3 wallets and curves in the same security class as those discussed in this study. Babbush et al. [35], a Google Quantum AI, Ethereum Foundation, and Stanford collaboration, reported that Shor’s algorithm for this problem can execute with either fewer than 1200 logical qubits and fewer than 90 million Toffoli gates, or fewer than 1450 logical qubits and fewer than 70 million Toffoli gates, running in minutes on fewer than half a million physical qubits at a 10^{-3} error rate. Cheignard, Fouque, and Schrottenloher [36], published in the peer-reviewed EUROCRYPT 2026 proceedings, independently reduced the estimate to roughly 1193 logical qubits, below the qubit count separately estimated for factoring 3072-bit RSA; Luo et al. [37] independently reduced the earlier estimate of roughly 2124 qubits reported by Häner et al. to about 1333. None of these results has been demonstrated on physical hardware at the required scale, but the direction is unambiguous: multiple independent groups converged on roughly the same 1200–1450-qubit range within a few months of each other in

2026, down from the prior benchmark of over 2100 qubits. A September 2026 estimate by Häner et al. [51] translates a comparable logical-qubit count into a physical-hardware figure for a trapped-ion architecture: roughly 19,397 physical qubits solving the same problem in approximately 25.7 days, down from prior trapped-ion estimates of 1.2 to 9.4 million physical qubits. As preprints or very recent results, these figures should be read as a fast-moving direction rather than a settled number; the recommendation in Section IX. C is unaffected by exactly which proves most durable.

Figures are as reported by each source; none has been demonstrated on physical quantum hardware at the scale required, and estimation methodology differs slightly across sources.

B. Implications for this Framework

The framework proposed in this study is, like essentially all currently deployed Web 3.0 authentication and wallet infrastructure, built entirely on classical ECC and therefore offers no protection against a sufficiently capable quantum adversary: such an adversary could recover a party's private key k_A or k_B from its public key, and could therefore forge the mutual-authentication handshake of Section V-C outright, independent of the ephemeral, session-unique design that protects against the classical replay and man-in-the-middle attacks analyzed in Section VI. A further, distinct risk is "harvest-now, decrypt-later": an adversary who records protocol transcripts or on-chain public keys today could decrypt or forge against them retroactively once a cryptographically relevant quantum computer exists, which matters specifically for long-lived Web 3.0 identities that are expected to remain valid for years. It is important to be precise about what current estimates do and do not show: no cryptographically relevant quantum computer capable of running these circuits exists today, and the qubit counts in Table VII, while dropping quickly, remain roughly three orders of magnitude beyond current hardware. The risk this section addresses is therefore one of migration planning, not of an immediate break.

C. Standards Guidance and a Migration Path

NIST finalized the first three post-quantum cryptography standards in August 2024: FIPS 203 (ML-KEM, a module-lattice key-encapsulation mechanism), FIPS 204 (ML-DSA, a module-lattice digital signature scheme), and FIPS 205 (SLH-DSA, a stateless hash-based digital signature scheme) [38]. NIST Internal Report 8547 (initial public draft) sets out a migration timeline in which algorithms providing 112-bit classical security, including RSA-2048 and ECC P-256, are to be deprecated after 2030 and disallowed after 2035, with the same disallowance date applying to higher-strength classical public-key algorithms including the curves discussed in this study [32]. Read against that timeline, the ECC-based framework proposed here remains an appropriate near- and medium-term choice, consistent with current NIST guidance, but it is not a long-term end state.

A practical migration path, consistent with how the wider field is approaching this problem, is a hybrid construction rather than a wholesale redesign: retaining the ephemeral ECEG encoding and signature-based handshake of Section V for their efficiency and their resistance to classical replay and man-in-

the-middle attacks, while wrapping the long-term key-establishment step in a post-quantum key-encapsulation mechanism such as ML-KEM, so that an adversary must break both the classical and post-quantum components to succeed. Recent quantum-resistant self-sovereign-identity frameworks [39,40] establish precedent for this pattern by layering NIST-standardized post-quantum primitives onto a decentralized identity architecture; designing and evaluating a concrete hybrid variant of the present framework along these lines, rather than merely outlining the path as this section does, is identified as future work in Section X.

X. LIMITATIONS AND FUTURE WORK

The evaluation in this study is analytic rather than empirical: the security discussion in Section VI is an informal argument based on standard cryptographic hardness assumptions rather than a machine-checked proof, and the efficiency discussion in Section VII is based on asymptotic complexity and established key-size equivalences rather than measured execution time on physical hardware. This limitation directly implies five concrete directions for follow-up work.

1) *Formal verification of the protocol.* The plan of Section VI-F should be encoded in Tamarin [54] and run against the documented queries for mutual authentication, replay resistance, secrecy, and erasure-conditional forward secrecy. The present manuscript reports no machine-checked verdict because that encoding has not yet been produced or executed.

2) *Formal and empirical validation of the identity-binding mechanism.* The DID/VC binding of Section VI-B is specified concretely but has not been analyzed as part of a single verified system together with the signature-and-key-transport handshake of Section V-C, nor implemented against a live DID method (e.g., did:key or did:ethr) to measure the latency and message-size overhead the credential-presentation step adds to the base handshake of Table VI.

3) *Hardware benchmarking.* To confirm the asymptotic gains of Section VII under practical operating conditions rather than theoretical bounds, the protocol should be implemented on representative resource-constrained platforms, such as microcontroller-class IoT devices or embedded wallet hardware, with measured wall-clock execution time, memory usage, and energy consumption, reported against the same curve parameters and RSA baseline used by the closest prior art [3] to allow a direct, like-for-like comparison rather than a comparison of asymptotic bounds alone.

4) *Integration with anomaly detection.* The authentication layer could be supplemented with a separate anomaly-detection mechanism, such as the machine-learning-based approach studied for cloud data in [27], to recognize statistically unexpected authentication patterns, for example, an unusually high number of failed handshakes even when each individual attempt is cryptographically valid.

5) *Hybrid post-quantum migration.* As Section IX discusses, a concrete hybrid variant of the protocol that wraps the long-term key-establishment step in a NIST-standardized post-quantum key-encapsulation mechanism such as ML-KEM

[38] should be designed and evaluated, following the precedent of recent quantum-resistant self-sovereign-identity frameworks [39,40], so that the protocol's classical efficiency advantages are retained in the near term without leaving it exposed once quantum resource estimates such as those in Table VII are realized in hardware.

The present framework addresses authentication and identity binding only. It does not address smart-contract-level vulnerabilities such as reentrancy or logic errors, nor the social-engineering component of attacks such as rug pulls, which rely on manipulating user decisions rather than defeating the cryptographic layer. These are complementary areas of defense that remain necessary for a complete Web 3.0 security posture and are outside the scope of the authentication mechanism proposed here. Finally, as Section VI-E notes, full anonymity in the self-sovereign-identity sense also depends on the application-layer binding between public keys and real-world identity, which this protocol does not itself provide and which should be addressed jointly with a DID/verifiable-credential layer in any deployment.

XI. CONCLUSION

This study presented an authentication framework for decentralized Web 3.0 environments, built around Elliptic Curve Cryptography, an ephemeral Elliptic Curve ElGamal encoding scheme, and a mutual-authentication handshake that combines digital signatures over each party's long-term key with a two-ephemeral-key ECEG transport of the encoding parameters. The framework was evaluated against an explicit Dolev–Yao threat model with an explicit in-scope/out-of-scope adversary boundary (Table III); the informal security analysis indicates resistance to passive eavesdropping, replay, and, conditional on ephemeral-key erasure, later compromise of a long-term key, attributable to session-unique ephemeral parameters bound by the hardness of the Elliptic Curve Discrete Logarithm Problem. Resistance to man-in-the-middle substitution additionally requires the authenticated binding between long-term public keys and identities that Section VI-B supplies through a concrete decentralized-identifier/verifiable-credential mechanism. A protocol-level operation count and a computational complexity comparison against RSA, checked against the current NIST SP 800-57 equivalence table, showed the ECC security-to-key-size advantage while making clear that complete-protocol runtime, energy, and bandwidth performance remain to be established experimentally.

As Section II-D makes explicit, the ephemeral ECEG character-encoding step builds directly on prior work [2,3]; this study's own contribution is the protocol-level replacement of that prior work's static-key parameter transport with a signature-and-two-ephemeral-key-ECEG construction, together with the explicit Web 3.0-specific threat model, the DID/VC identity-binding mechanism, the requirements analysis, and the concrete formal-verification plan of Section VI-F built around it. Section IX further situated the framework's classical ECC foundation against the rapidly narrowing 2025–2026 quantum resource estimates for the elliptic-curve discrete logarithm problem — including a September 2026 physical-hardware estimate of roughly 19,397 qubits — and against current NIST post-quantum migration guidance, concluding that the framework

remains an appropriate near- and medium-term choice but should be extended with a hybrid post-quantum key-establishment step ahead of the 2030–2035 deprecation horizon. Because the present evaluation is analytical, the five next steps identified in Section X — formal protocol verification, hardware-based benchmarking, anomaly-detection integration, hybrid post-quantum migration, and formal/empirical validation of the identity-binding mechanism — are necessary before the framework can be considered validated for production deployment. Subject to that validation, the proposed approach offers a practical building block for strengthening authentication and mitigating identity theft, phishing, and unauthorized access in self-sovereign, user-centric Web 3.0 identity ecosystems.

REFERENCES

- [1] Achary, R.; Raj, S.; Shelke, C.J. Integrated blockchain model for personally identifiable information system (IBPIS): A secure approach. In Proceedings of the 2024 3rd International Conference for Innovation in Technology (INOCON), Bangalore, India, 1–3 March 2024; pp. 1–7.
- [2] Baccouri, S.; Farhat, H.; Gharbi, N.; Tahar, R.; Azzabi, T.; Attia, R. Ephemeral encoding message in the elliptic curve cryptography for IoT. In Proceedings of the 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET), Hammamet, Tunisia, 22–25 March 2022; pp. 172–177.
- [3] Baccouri, S.; Farhat, H.; Azzabi, T.; Attia, R. Lightweight authentication scheme based on Elliptic Curve El Gamal. *J. Inf. Telecommun.* 2024, 8, 231–261.
- [4] Dinarvand, N.; Barati, H. An efficient and secure RFID authentication protocol using elliptic curve cryptography. *Wirel. Netw.* 2019, 25, 415–428.
- [5] Lamrani Alaoui, H.; El Ghazi, A.; Zbakh, M.; Touhafi, A.; Braeken, A. A highly efficient ECC-based authentication protocol for RFID. *J. Sens.* 2021, 2021, 8876766.
- [6] Noori, D.; Shakeri, H.; Niazi Torshiz, M. An elliptic curve cryptosystem-based secure RFID mutual authentication for the Internet of things in healthcare environment. *EURASIP J. Wirel. Commun. Netw.* 2022, 2022, 64.
- [7] Sowjanya, K.; Dasgupta, M.; Ray, S. Elliptic curve cryptography based authentication scheme for internet of medical things. *J. Inf. Secur. Appl.* 2021, 58, 102761.
- [8] Khan, M.A.; Quasim, M.T.; Alghamdi, N.S.; Khan, M.Y. A secure framework for authentication and encryption using improved ECC for IoT-based medical sensor data. *IEEE Access* 2020, 8, 52018–52027.
- [9] Kumari, A.; Jangirala, S.; Abbasi, M.Y.; Kumar, V.; Alam, M. ESEAP: ECC-based secure and efficient mutual authentication protocol using smart card. *J. Inf. Secur. Appl.* 2019, 51, 102443.
- [10] Kumari, A.; Yahya Abbasi, M.; Kumar, V.; Khan, A.A. A secure user authentication protocol using elliptic curve cryptography. *J. Discrete Math. Sci. Cryptogr.* 2019, 22, 521–530.
- [11] Li, X.; Peng, J.; Niu, J.; Wu, F.; Liao, J.; Choo, K.R. A robust and energy efficient authentication protocol for industrial internet of things. *IEEE Internet Things J.* 2018, 5, 1606–1615.
- [12] Panja, A.; Mondal, S.; Islam, K.A.; Ghosh, T.K.; Karforma, S. An efficient and secure fingerprint-based authentication scheme using elliptic curve cryptography. *Webology* 2022, 19.
- [13] Ramasamy, P.; Ranganathan, V.; Palanisamy, V.; Kadry, S. Securing one-time password generation using elliptic-curve cryptography with self-portrait photograph for mobile commerce application. *Multimed. Tools Appl.* 2020, 79, 17081–17099.
- [14] Sadri, M.J.; Asaar, M.R. An anonymous two-factor authentication protocol for IoT-based applications. *Comput. Netw.* 2021, 199, 108460.
- [15] Elias, E.P.; Santhanavijayan, A.; Janet, B.; Arul, K.R.J. OTP system based on ECC key exchange. In Proceedings of the 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), Chennai, India, 3–4 March 2022; pp. 1–6.

- [16] Kaur, S.; Kaur, G.; Shabaz, M. A secure two-factor authentication framework in cloud computing. *Secur. Commun. Netw.* 2022, 2022, 7540891.
- [17] Islam, M.M.; Hossain, M.S.; Hasan, M.K.; Shahjalal, M.; Jang, Y.M. Design and implementation of high-performance ECC processors with unified point addition on twisted Edwards curve. *Sensors* 2020, 20, 5148.
- [18] Basu Roy, D.; Mukhopadhyay, D. High-speed implementation of ECC scalar multiplication in GF(p) for generic Montgomery curves. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 2019, 27, 1587–1600.
- [19] Lara-Nino, C.A.; Diaz-Perez, A.; Morales-Sandoval, M. Elliptic curve lightweight cryptography: A survey. *IEEE Access* 2018, 6, 72514–72550.
- [20] Prakash, M.; Ramesh, K. Blockchain-enabled decentralized lightweight authentication scheme with dynamic difficulty adjustment using ECC for IoT networks. *Cluster Comput.* 2025, 28, 898.
- [21] Liu, H.; Cao, J.; Cao, Y.; Bai, D.; Liang, J.; Li, R. EdenDID: an edge computing and blockchain-based decentralized identity system for Web3 applications and DePIN. *Cluster Comput.* 2025, 28, 1048.
- [22] Le, H.V.A.; Nguyen, Q.D.N.; Tadashi, N.; Tran, T.H. Blockchain-Based Decentralized Identity Management System with AI and Merkle Trees. *Computers* 2025, 14, 289.
- [23] Achary, R.; Shelke, C.J.; Marx, K.; Rajesh, A. Security implementation on IoT using CoAP and elliptical curve cryptography. *Procedia Comput. Sci.* 2023, 230, 493–502.
- [24] Zheng, J.; Wang, X.; Yang, Q.; Xiao, W.; Sun, Y.; Liang, W. A blockchain-based lightweight authentication and key agreement scheme for internet of vehicles. *Connect. Sci.* 2022, 34, 1430–1453.
- [25] Cremers, C.J.F. Scyther: Semantics and Verification of Security Protocols. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
- [26] Blanchet, B. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Found. Trends Priv. Secur.* 2016, 1, 1–135.
- [27] K. K. G. V.; Achary, R.; Shelke, C.J. Cloud data anomaly detection using machine learning. In Proceedings of the 2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS), Gobichettipalayam, India, 18–19 April 2024; pp. 1245–1251.
- [28] Kanjanaputhipong, W.; Boonkrong, S. Blockchain-Based Decentralised Authentication in Closed Environments. *Future Internet* 2025, 17, 98.
- [29] Goel, A.; Rahulamathavan, Y. A Comparative Survey of Centralised and Decentralised Identity Management Systems: Analysing Scalability, Security, and Feasibility. *Future Internet* 2025, 17, 1.
- [30] Ambati, S.; Adeel, K.; Myers, J.; Ivanov, N. SoK: Self-Sovereign Digital Identities. In Proceedings of the 31st ACM Symposium on Access Control Models and Technologies (SACMAT 2026), New York, NY, USA, 2026.
- [31] National Institute of Standards and Technology. *Recommendation for Key Management: Part 1 General*; NIST Special Publication 800-57 Part 1 Revision 5; NIST: Gaithersburg, MD, USA, 2020.
- [32] National Institute of Standards and Technology. *Transition to Post-Quantum Cryptography Standards*; NIST Internal Report 8547 (Initial Public Draft); NIST: Gaithersburg, MD, USA, 2024.
- [33] Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* 1983, 29, 198–208.
- [34] Alotaibi, B. Cybersecurity Attacks and Detection Methods in Web 3.0 Technology: A Review. *Sensors* 2025, 25, 342.
- [35] Babbush, R.; Zalcman, A.; Gidney, C.; Broughton, M.; Khattar, T.; Neven, H.; Bergamaschi, T.; Drake, J.; Boneh, D. Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations. *PRX Quantum* 2026, 7, 031001.
- [36] Chevignard, C.; Fouque, P.-A.; Schrottenloher, A. Reducing the Number of Qubits in Quantum Discrete Logarithms on Elliptic Curves. In *Advances in Cryptology EUROCRYPT 2026*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2026; pp. 371–401.
- [37] Luo, H.; Yang, Z.; Wang, Z.; Su, Y.; Li, T. Space-Efficient Quantum Algorithm for Elliptic Curve Discrete Logarithms with Resource Estimation. arXiv:2604.02311, 2026. Preprint.
- [38] National Institute of Standards and Technology. *Announcing Issuance of Federal Information Processing Standards (FIPS) 203, 204, and 205*; Federal Register 2024-17956; NIST: Gaithersburg, MD, USA, 2024.
- [39] Fiaz, F.; Muhammad, Z. QR-MetaSSI: A Quantum-Resistant Self-Sovereign Identity Framework for Metaverse Platforms. *J. Cybersecur. Priv.* 2026, 6, 111.
- [40] Radanliev, P.; Maple, C.; Santos, O. Complying with the NIST post-quantum cryptography standards and decentralizing artificial intelligence: methodology for quantum-resistant and privacy-preserving digital identity systems. *Front. Blockchain* 2025, 8, 1702066.
- [41] Kobitz, N. Elliptic curve cryptosystems. *Math. Comput.* 1987, 48, 203–209.
- [42] W3C. *Decentralized Identifiers (DIDs) v1.0*. W3C Recommendation, 19 July 2022.
- [43] W3C. *Verifiable Credentials Data Model v2.0*. W3C Recommendation, 2025.
- [44] Wuille, P.; Nick, J.; Ruffing, T. *BIP-340: Schnorr Signatures for secp256k1*. Bitcoin Improvement Proposals, 2020.
- [45] Pornin, T. *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. IETF RFC 6979, 2013.
- [46] Standards for Efficient Cryptography Group (SECG). *SEC 1: Elliptic Curve Cryptography, Version 2.0*. 2009.
- [47] National Institute of Standards and Technology. *Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters*. NIST SP 800-186, 2023.
- [48] Hammi, B.; Fayad, A.; Khatoun, R.; Zeadally, S.; Begriche, Y. A lightweight ECC-based authentication scheme for internet of things (IoT). *IEEE Syst. J.* 2020, 14, 3440–3450.
- [49] Li, M.; Hu, S. A Lightweight ECC-Based Authentication and Key Agreement Protocol for IoT with Dynamic Authentication Credentials. *Sensors* 2024, 24, 7967.
- [50] Rodriguez Garzon, S.; Natusch, D.; Philipp, A.; Küpper, A.; Einsiedler, H.J.; Schneider, D. DID Link: Authentication in TLS with Decentralized Identifiers and Verifiable Credentials. In Proceedings of the 2024 21st Annual International Conference on Privacy, Security and Trust (PST), 2024.
- [51] Häner, T.; Tripier, F.; Young, J.; Naehrig, M.; Maksymov, A.; Alam, S.; Maslov, D.; Parrott, M.; de Sereville, Y.; Sullivan, J.; Webster, M.; Delfosse, N.; Gamble, J.; Roetteler, M. Computing 256-Bit Elliptic Curve Discrete Logarithms in 26 Days on a Fault-Tolerant Trapped-Ion Quantum Computer with 20,000 Qubits. arXiv:2609.05625, 2026. Preprint.
- [52] Balfagih, M.; Balfagih, Z.; Lytras, M.D.; Alfawaz, K.M.; Alshdadi, A.A.; Alsolami, E. A Blockchain-Enabled IoT Logistics System for Efficient Tracking and Management of High-Price Shipments: A Resilient, Scalable and Sustainable Approach to Smart Cities. *Sustainability* 2023, 15, 13971.
- [53] Balfagih, M. Enhancing Security and Flexibility in Smart Locker Systems: A Multi-Authentication Approach with IoT Integration. In Proceedings of the 2024 21st Learning and Technology Conference (L&T), 2024; pp. 325–329.
- [54] Tamarin Prover. *Tamarin Manual, Version 1.12*; Tamarin Prover Project, 2026.