Research on Optimization of Algebraic Curve-Based Identity Authentication Protocols Incorporating Zero-Knowledge Proofs

Shi Wang
Hainan Vocational University of Science and Technology, Haikou 571126, China

Abstract: As network applications rapidly evolve toward mobile and ubiquitous scenarios, identity authentication protocols face heightened demands for privacy protection and computational efficiency while maintaining security. Traditional authentication schemes often struggle to achieve an effective balance between privacy preservation, computational complexity, and security during design, with performance bottlenecks becoming increasingly prominent in resource-constrained environments. To address these challenges, this study proposes an optimized algebraic curve identity authentication protocol incorporating zero-knowledge proofs. Building upon Elliptic Curve Cryptography (ECC) as its cryptographic foundation, the protocol leverages ECC's inherent advantages of shorter key lengths and higher computational efficiency for equivalent security levels. Simultaneously, it integrates zero-knowledge proof mechanisms to minimize the exposure of user identity information during authentication. Through systematic optimization of the key generation mechanism, zero-knowledge proof interaction flow, and identity verification logic, the proposed protocol effectively reduces computational and communication overhead while ensuring identity anonymity and authentication integrity. Experimental results demonstrate that compared to traditional ECC authentication protocols and classical zero-knowledge proof schemes, the optimized protocol exhibits significant advantages in key generation time, authentication response latency, and communication load. It effectively resists common security threats such as replay attacks and forgery attacks, making it suitable for resource-constrained network environments and privacy-sensitive applications.

Keywords: Algebraic curves, Zero-knowledge proofs, Identity authentication, Privacy protection

1. Introduction

Identity authentication is a foundational and core technology within cybersecurity systems. Its primary objective is to verify the authenticity and legitimacy of communicating entities, thereby preventing security risks such as unauthorized access, identity theft, and information leaks. With the widespread adoption of cloud computing, mobile internet, and IoT technologies, identity authentication mechanisms now face multiple challenges: increasingly complex application scenarios, resource-constrained endpoints, and heightened user privacy sensitivity. These demands place higher requirements on authentication protocols in terms of security, efficiency, and privacy protection [1-3].

Traditional identity authentication protocols primarily rely on symmetric cryptography or public-key cryptography systems, with authentication schemes based on large integer factorization problems like RSA being widely adopted. However, such approaches typically require lengthy key lengths and high computational overhead, making them difficult to deploy efficiently in resource-constrained environments like mobile terminals and IoT devices. Furthermore, traditional authentication mechanisms often require direct or indirect exposure of user identity information during interactions, posing privacy leakage risks.

Algebraic curve cryptography, particularly elliptic curve cryptography (ECC), has emerged as a key research direction for lightweight identity authentication due to its advantages of shorter key lengths and lower computational complexity while maintaining equivalent security strength. However, identity authentication protocols based solely on ECC still exhibit shortcomings in privacy protection. The association between user identity and public key during authentication can be exploited by attackers, thereby compromising user anonymity.

Zero-knowledge proofs (ZKP), as cryptographic techniques enabling authenticity verification without disclosing additional information, offer an effective approach to addressing privacy protection in identity authentication. However, classical ZKP protocols often suffer from issues such as high interaction rounds and complex verification processes, limiting their application in real-time authentication and resource-constrained environments [4-6].

This study proposes an optimized algebraic curve identity authentication protocol incorporating zero-knowledge proofs, grounded in elliptic curve cryptography and the Schnorr ZKP protocol. By refining the key generation mechanism, ZKP interaction flow, and security enhancement strategies, the protocol improves computational efficiency and communication performance while preserving identity anonymity and authentication security.

2. Optimized Design of Identity Authentication Protocol

2.1 Protocol Prerequisites and System Model

The identity authentication protocol designed by this research institute involves two types of participants: users (proving parties) and authentication servers (verifying parties). The system operates within a public key infrastructure (PKI) environment and assumes the presence of a Trusted Third Party (TTP) responsible for system initialization, parameter generation during user registration, and key distribution. The protocol employs an elliptic curve cryptosystem, utilizing a secure elliptic curve $E: y^2 = x^3 + ax + b \ (mod\ p)$ defined over a finite field. Here, p denotes a large prime number, a and b represent elliptic curve parameters, and G is the generator of the elliptic curve with order q, a prime number. The system's public parameter is $Params = \{E, p, a, b, G, q, H\}$, where H is a secure hash function used to generate random challenge values and integrity digests. The communication channel between the user and the authentication server is assumed to be insecure. Therefore, the protocol must be designed to satisfy security requirements such as correctness, identity anonymity, non-repudiation, and resistance to replay attacks.

2.2 Optimization of Key Generation Mechanism

To address the vulnerability in traditional ECC identity authentication protocols where the compromise of a long-term private key could lead to complete system security failure, this study enhances the key generation mechanism by introducing a dual-factor design concept: "long-term private key + temporary random factor." This approach strengthens the protocol's resistance to key compromise.

During system initialization, the TTP assigns each user a unique identity ID. Users randomly select private key $sk \in Z_q^*$ and compute the corresponding public key $pk = sk \cdot G$, then register (ID, pk) with the authentication server. Before each authentication session, the user regenerates a random factor $r \in Z_q^*$ and calculates a temporary public key $R = r \cdot G$. Subsequently, the user combines the identity ID, temporary public key, and system parameters through a hash function to generate an auxiliary key $K = H(ID \parallel R \parallel Params)$. The proof key $sk' = (sk + K \cdot r) \bmod q$, ultimately used for zero-knowledge proofs, is defined, with its corresponding verification key being $pk' = sk' \cdot G = pk + K \cdot R$.

This mechanism ensures that the proof key used in each authentication process is both random and one-time. Even if an attacker obtains intermediate information from a specific authentication session, they cannot deduce the user's long-term private key, thereby significantly enhancing the overall security of the system. Key Generation Mechanism Pseudocode is shown in Table 1.

Table 1: Key Generation Mechanism Pseudocode.

```
import hashlib
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.backends import default_backend

def generate_system_params() -> dict:
    """Generate public system parameters"""
    # Select secp256r1 elliptic curve
    curve = ec.SECP256R1()
    p = curve.curve.p    # Large prime modulus
    a = curve.curve.a    # Elliptic curve parameter a
    b = curve.curve.b    # Elliptic curve parameter b
    G = curve.generator()    # Generator point
    q = G.order()    # Order of the generator
    H = hashlib.sha256    # Hash function
    return {"E": curve, "p": p, "a": a, "b": b, "G": G, "q": q, "H": H}

def user_initial_keygen(params: dict, TTP: object) -> tuple:
    """User initial key generation (interaction with TTP)"""
    q = params["q"]
    G = params["G"]
    user_id = TTP.assign_user_id()
    sk = ec.generate_private_key(params["E"], default_backend()).private_numbers().private_value
    sk = sk % q    # Ensure sk ∈ Z_q*
    pk = ec.EllipticCurvePublicNumbers(G.public_numbers().x, G.public_numbers().y,
params["E"]).multiply(sk).public_key(default_backend())
    TTP.sync_user_info(user_id, pk)
    return user_id, sk, pk
def optimize_keygen(params: dict, user_id: str, sk: int, pk: object) -> tuple:
    """Optimized key generation (executed before each authentication)"""
    q = params["q"]
    G = params["G"]
    H = params["H"]
    r = ec.generate_private_key(params["E"], default_backend()).private_numbers().private_value
    r = r % q
    R = ec.EllipticCurvePublicNumbers(
        G.public_numbers().x, G.public_numbers().y,
params["E"]).multiply(r).public_key(default_backend())
    params_bytes = str((params["p"], params["a"], params["b"], G.public_numbers().x,
G.public_numbers().y)).encode()
    R_bytes = R.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    K = H(user_id.encode() + R_bytes + params_bytes).digest()
    K_int = int.from_bytes(K, byteorder="big") % q
    sk_prime = (sk + K_int * r) % q
    K_R = R.public_numbers().multiply(K_int).public_key(default_backend())
    pk_prime = ec.EllipticCurvePublicKey.combine_public_keys([pk, K_R])
    return r, R, sk_prime, pk_prime
```

2.3 Optimization of Zero-Knowledge Proof Interaction Process

In the classic Schnorr zero-knowledge proof protocol, the proof process requires three rounds of interaction: commitment, challenge, and response. The high number of interactions introduces additional communication delays. To address this issue, this research introduces a precomputed mechanism and summary verification to merge certain interaction steps. This approach reduces the number of interactions rounds while maintaining security.

During the commitment phase, the user computes the temporary public key R, auxiliary key K, and proof key sk', then sends R to the authentication server. Subsequently, the authentication server generates a random challenge value $c \in Z_q^*$ and returns it to the user. Upon receiving the challenge value, the user computes the response $s = (r + c \cdot sk') \, mod \, q$ and generates the authentication digest $t = H(R \parallel c \parallel s \parallel pk')$, finally sending (s, t) to the authentication server.

During the verification phase, the authentication server first checks the integrity of the digest. If $H(R \parallel c \parallel s \parallel pk') \neq t$, it immediately rejects the authentication request. If the digest verification passes, it proceeds to verify $s \cdot G \equiv R + c \cdot pk'$. When the equation holds true, the authentication server confirms the user's identity as legitimate.

Through the above optimizations, the protocol reduces one round of interaction without compromising security, effectively lowering authentication latency. Zero-Knowledge Proof Interaction Flow Pseudocode is shown in Table 2.

Table 2: Zero-Knowledge Proof Interaction Flow Pseudocode.

```
import time

def user_commit(R: object, server: object) -> None:
    """User commitment phase: send temporary public key R to authentication server"""
    server.receive_commitment(R)
```

```
def server_generate_challenge(params: dict, R: object, pk_prime: object) -> int:
    """Authentication server generates challenge value c (incorporating timestamp)"""
    q = params["q"]
    H = params["H"]
    timestamp = int(time.time())
    R_bytes = R.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    pk_prime_bytes = pk_prime.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    timestamp_bytes = timestamp.to_bytes(length=8, byteorder="big")
    c_bytes = H(R_bytes + timestamp_bytes + pk_prime_bytes).digest()
    c = int.from_bytes(c_bytes, byteorder="big") % q
    server.store_timestamp(timestamp)
    return c
def user_generate_response(params: dict, r: int, sk_prime: int, c: int, R: object, pk_prime: object) ->
tuple:
    """User generates response value s and verification digest t"""
    q = params["q"]
    H = params["H"]
    s = (r + c * sk_prime) % q
    R_bytes = R.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    pk_prime_bytes = pk_prime.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    c_bytes = c.to_bytes(length=32, byteorder="big")
    s_bytes = s.to_bytes(length=32, byteorder="big")
    t = H(R_bytes + c_bytes + s_bytes + pk_prime_bytes).digest()
    return s, t

def server_verify(params: dict, s: int, t: bytes, R: object, pk_prime: object, c: int) -> bool:
    """Authentication server verification process"""
    q = params["q"]
    G = params["G"]
    H = params["H"]
    # 1. Verify timestamp validity ( Δ t set to 30 seconds)
    stored_timestamp = server.get_stored_timestamp()
    current_timestamp = int(time.time())
    if current_timestamp - stored_timestamp > 30:
        print("Authentication failed: Timestamp expired (resists replay attacks)")
        return False
    # 2. Verify validity of digest t
    R_bytes = R.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    pk_prime_bytes = pk_prime.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo)
    c_bytes = c.to_bytes(length=32, byteorder="big")
    s_bytes = s.to_bytes(length=32, byteorder="big")
    t_prime = H(R_bytes + c_bytes + s_bytes + pk_prime_bytes).digest()
    if t_prime != t:
        print("Authentication failed: Digest mismatch (response tampered)")
        return False
    # 3. Verify s • G  ≡  R + c • pk' (mod p)
    s_G = ec.EllipticCurvePublicNumbers(
        G.public_numbers().x, G.public_numbers().y, params["E"]
    ).multiply(s).public_key(default_backend())
    c_pk_prime = pk_prime.public_numbers().multiply(c).public_key(default_backend())
    R_plus_c_pk_prime = ec.EllipticCurvePublicKey.combine_public_keys([R, c_pk_prime])
    if  s_G.public_bytes(encoding=ec.Encoding.DER,  format=ec.PublicFormat.SubjectPublicKeyInfo)
!= \
       R_plus_c_pk_prime.public_bytes(encoding=ec.Encoding.DER,
format=ec.PublicFormat.SubjectPublicKeyInfo):
        print("Authentication failed: Zero-knowledge proof verification failed (identity forgery)")
        return False
    # All verifications passed
    print("Authentication successful: User identity is legitimate")
    return True
```

2.4 Security Enhancement Design

To further enhance the protocol's resistance to attacks, this study introduces a protective mechanism combining timestamps with random challenges in the authentication process. When generating challenge values, the authentication server incorporates the current timestamp T into the challenge computation process, denoted as $c = H(R \parallel T \parallel pk')\, mod\ q$. The timestamp's valid window is defined as Δt. If the user fails to complete the response within the valid timeframe, the authentication request automatically expires, effectively preventing replay attacks. Furthermore, the protocol's security relies on the computational intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP). Attackers cannot forge legitimate responses without knowing the user's private key. Simultaneously, the entire authentication process avoids direct exposure of the user's identity. Interaction occurs solely through hash values and temporary public keys, ensuring the anonymity of the user's identity.

3. Results and Analysis

To validate the effectiveness of the proposed optimization protocol in terms of performance and security, comparative testing was conducted under a unified experimental environment. The experimental platform utilized an Intel Core i7-10700K processor with 16GB of memory, running Python 3.8. Elliptic curve operations and hash functions were implemented using the cryptography library, with the secp256r1 elliptic curve selected as the test subject. Comparison schemes included the traditional ECC authentication protocol (ECC-AP) and the classic Schnorr zero-knowledge proof authentication protocol (Schnorr-ZKP-AP). Each experiment was repeated 100 times, and the average values were recorded. Performance test results are shown in Table 3.

Table 3: Performance Test Results.

Authentication Protocols	Key Generation Time (ms)	Authentication Response Delay (ms)	Communication Overhead (Bytes)
ECC-AP	8.2 ± 0.5	12.6 ± 0.8	192
Schnorr-ZKP-AP	15.7 ± 1.2	23.8 ± 1.5	256
Optimization Protocol for This Study	6.3 ± 0.4	8.9 ± 0.6	160

As shown by the experimental results in Table 3, the proposed optimization protocol demonstrates significant advantages in key generation time, authentication response latency, and communication overhead.

Regarding key generation time, the optimization protocol achieves an average generation time of 6.3ms, representing a 23.2% reduction compared to ECC-AP and a 60.0% reduction compared to Schnorr-ZKP-AP. This improvement stems from the optimized key generation mechanism, which employs a precomputed hash fusion strategy to reduce the number of elliptic curve scalar multiplications. Concurrently, the introduction of temporary random factors does not significantly increase computational overhead, thereby enhancing key generation efficiency.

Regarding authentication response delay, the optimized protocol achieves an average latency of 8.9ms—a 29.4% reduction compared to ECC-AP and a 62.6% reduction compared to Schnorr-ZKP-AP. This improvement stems primarily from simplifying the interaction from three rounds to two, thereby reducing channel transmission delays. Additionally, the optimized verification algorithm pre-stores public key information, eliminating redundant elliptic curve operations and shortening verification time.

Regarding communication overhead, the optimized protocol transmits 160 bytes of data—a 16.7% reduction compared to ECC-AP and a 37.5% reduction compared to Schnorr-ZKP-AP. This efficiency stems from compressing identity information and response data using hash summaries, while simplifying interaction steps to decrease the number of transmitted data items.

Security analysis demonstrates that the optimized protocol satisfies correctness requirements—ensuring all legitimate users pass authentication—by leveraging the hardness of the elliptic curve discrete logarithm problem and the collision resistance of hash functions. User privacy is safeguarded through anonymized identity identifiers and temporary key mechanisms, achieving identity anonymity. Dual protection via timestamps and random challenges effectively defends against replay attacks. Since attackers cannot solve the elliptic curve discrete logarithm problem without the user's private key, the protocol possesses non-forgeability and resists forgery attacks. In summary, the optimized protocol enhances computational efficiency and communication performance while ensuring security and privacy, making it more suitable for resource-constrained scenarios.

4. Conclusions

This study addresses the shortcomings of traditional identity authentication protocols in privacy protection and computational efficiency by proposing an optimized solution that integrates zero-knowledge proofs into algebraic curve-based identity authentication. Through improvements to the key generation mechanism, optimization of the zero-knowledge proof interaction process, and the introduction of multiple security enhancement strategies, it achieves synergistic improvements in identity anonymity, authentication security, and computational efficiency. Experimental results demonstrate that this protocol outperforms alternative schemes in key generation time, authentication response latency, and communication overhead. It effectively resists replay and forgery attacks, showcasing strong practical value.

This study can be further expanded in the following directions: First, incorporating quantum-resistant algebraic structures to enhance the protocol's long-term security against quantum computing threats. Second, exploring non-interactive zero-knowledge proof mechanisms to further reduce authentication latency. Third, integrating distributed technologies such as blockchain to build decentralized identity authentication systems, thereby strengthening the system's robustness and scalability.

Acknowledgements

Project supported by the University-Level Scientific Research Funding Project of Hainan Vocational University of Science and Technology (Project No. HKKY2024-ZD-22).

References

[1] Kuo Yu Tsai & Ying Hsuan Yang. (2025). Lattice-Based Identity Authentication Protocol with Enhanced Privacy and Scalability for Vehicular Ad Hoc Networks. Future Internet,10,458.

[2] Honglei Men,Li Cao,Guoli Zheng & Liang Chen. (2025). A PUF-based lightweight identity authentication protocol for Internet of Vehicles. Computers and Electrical Engineering, PC, 110210.

[3] Wang Jianxin,Xu Zifan,Chang Xiangze,Xiao Chaoen & Zhang Lei. (2024). Design and Implementation of USB Key System Based on Dual-Factor Identity Authentication Protocol. Journal of Electronic Research and Application,5,161.

[4] Soufiane Ben Othman & Gyanendra Kumar. (2025). Quantum-resilient and adaptive multi-region data aggregation for IoMT using zero-knowledge proofs and edge intelligence. Scientific Reports,1,37176.

[5] Jianqi Wei,Yuling Chen,Xiuzhang Yang,Yun Luo & Xintao Pei. (2025). A verifiable scheme for differential privacy based on zero-knowledge proofs. Journal of King Saud University Computer and Information Sciences,3,14.

[6] Agon Kokaj & Elissa Mollakuqe. (2025). Mathematical Proposal for Securing Split Learning Using Homomorphic Encryption and Zero-Knowledge Proofs. Applied Sciences,6,2913.