Post-Quantum Authenticated Key Exchange via Signcryption with Ephemeral Key Masking

Mostefa Kara
LIAP Laboratory, University of El Oued,
PO Box 789, El Oued,
39000, Algeria
karamostefa@univ-eloued.dz

Konstantinos Karampidis
Department of Electrical and Computer Engineering,
Hellenic Mediterranean University,
Crete, Greece
karampidis@hmu.gr

Muath AlShaikh
Cyber Security Department,
College of Engineering, Al Ain University,
Abu Dhabi, UAE
muath.alshaikh@aau.ac.ae

***Abstract*—We present PQES-AKE, a novel two-party authenticated key exchange (AKE) protocol built upon the Post-Quantum Encryption and Signcryption Scheme (PQES) introduced by Kara et al. The protocol achieves mutual authentication, session key secrecy, and forward secrecy in a post-quantum adversarial model. The central design principle of PQES-AKE is the concealment of ephemeral Diffie-Hellman (DH) keys within affine masks derived from randomness generated internally by the PQES signcryption algorithm. The security of PQES-AKE rests on three hardness assumptions, including Learning With Errors (LWE) for ciphertext confidentiality, a Secret-Base Discrete Exponentiation (SBDE) assumption for signature unforgeability, and the Computational Diffie-Hellman (CDH) assumption for session key secrecy. The protocol completes in two network rounds, with a total communication cost of approximately 627 bytes, and requires 17 modular exponentiations. A Python prototype evaluated on a commodity Intel Core i7 laptop achieves an average execution time of 6.2 ms. These results confirm that PQES-AKE provides a competitive, single-primitive alternative to composite KEM-then-authenticate constructions in post-quantum secure channel establishment.**

***Index Terms*—authenticated key exchange, signcryption, Diffie-Hellman masking, forward secrecy, mutual authentication, ephemeral key hiding**

I. Introduction

The security of deployed public-key infrastructure rests on the hardness of integer factorisation and discrete logarithm, both of which are efficiently solvable by Shor's quantum algorithm [1]. As quantum hardware matures, protocols that expose classical public-key material in their transcripts become directly vulnerable, motivating an urgent transition to post-quantum alternatives [2], [3]. Authenticated key exchange (AKE) protocols are among the most critical targets; schemes such as TLS 1.3 [4] and IKEv2 transmit ephemeral Diffie-Hellman (DH) keys $X = g^x$ in the clear, allowing a quantum adversary to recover x via Shor's discrete-logarithm reduction and subsequently compute the session secret.

Following the NIST post-quantum standardisation process [5], the most widely adopted post-quantum AKE designs combine a lattice-based key encapsulation mechanism (KEM) such as ML-KEM (Kyber) [6] with a lattice-based signature such as ML-DSA (Dilithium) [7] in a KEM-then-Sign or hybrid composition [8], [9]. While these constructions inherit strong security foundations from the Module Learning With Errors (MLWE) problem [10], they are composite designs requiring two independent primitives, increasing implementation complexity and bandwidth, which limits their deployment in lightweight or real-time scenarios such as IoT and WSN [11], [12]. Recent hybrid TLS deployments [13] still transmit raw ECDH public keys alongside the KEM ciphertext, leaving them exposed to quantum discrete-logarithm attacks.

Signcryption [14] offers a single-primitive alternative that simultaneously achieves confidentiality and authentication at lower cost than separate encrypt-then-sign compositions [15]. Post-quantum signcryption schemes based on lattice problems [16] and other hard problems have been proposed, but none directly address the problem of protecting ephemeral DH keys from quantum adversaries within an AKE setting.

In this paper, we propose PQES-AKE, a two-party AKE protocol built on PQES (Post-Quantum Encryption and Signcryption) Scheme [17]. The central contribution is a novel affine masking technique; Alice's ephemeral DH key $X_A = g^{x_A}$ and Bob's ephemeral DH key $X_B = g^{x_B}$ are never transmitted in the clear. Instead, they are hidden as $M = r_1 X_A + r_2 \bmod p$ and $M' = r_2 X_B + r_1 \bmod p$, where (r_1, r_2) are session-fresh values recoverable only by a party that successfully performs PQES unsigncryption. This eliminates the discrete-logarithm attack surface from the transcript entirely, without requiring a second primitive. A fresh random nonce m is signcrypted by PQES, binding Alice's long-term identity to the session and producing the masked randomness (r_1, r_2) as a byproduct.

The security of PQES-AKE rests on three assumptions, including Learning With Errors (LWE) for ciphertext confidentiality, an introduced Secret-Base Discrete Exponentiation (SBDE) assumption for signature unforgeability, and Computational Diffie-Hellman (CDH) for session key secrecy. Unlike classical discrete logarithm, SBDE requires recovery of k_1, k_2 from $k_1^{k_1} \bmod p'$ and $k_1^{k_2} \bmod p'$ where both base and exponents are unknown, a strictly harder problem to which Shor's algorithm does not apply, as it requires a known fixed base. We prove SK-security and forward secrecy via game-hopping in the Bellare-Rogaway model [18], and establish mutual authentication by reduction to PQES unforgeability. A Python prototype achieves 6.2 ms average over 100 runs on

Preprint, 2026

a commodity Intel Core i7 laptop, with a two-round communication cost of $\approx$627 bytes, significantly more compact than KEM-then-Sign alternatives.

II. Post-Quantum Encryption and Signcryption Scheme (PQES)

The Post-Quantum Encryption and Signcryption Scheme (PQES) is a previously published cryptographic construction that simultaneously provides confidentiality and authenticity in a post-quantum setting. PQES operates over a large prime field $\mathbb{Z}_p$ augmented by a secondary prime p' that governs the signature component.

Each party holds a long-term key pair. The private key consists of small-norm scalars (s, e) and two unknown-base discrete logarithm exponents (k_1, k_2) over $\mathbb{Z}_{p'}$. The corresponding public key is the tuple $(a,\ b = as+e \bmod p,\ K_{11} = k_1^{k_1} \bmod p',\ K_{12} = k_1^{k_2} \bmod p')$, whose hardness relies on a combination of Learning-With-Errors (LWE)-style linear masking and unknown-base discrete logarithm exponent assumptions in $\mathbb{Z}_{p'}$.

The signcryption algorithm PQES_Enc&Sig takes a plaintext message m, the receiver's public key, and the sender's private key. It produces a ciphertext pair (u, v) and a signature pair (s_1, s_2), as well as two internal randomness values (r_1, r_2) that are used both during encryption and as verifiable witnesses in the signature equations. The ciphertext embeds m inside a randomised linear combination of the receiver's public-key components, while the signature binds the sender's identity to the message digest through a set of modular exponentiation relations involving k_1, k_2, and hash outputs.

The unsigncryption algorithm PQES_Dec&Verf inverts the process: the receiver uses its private key (s_b, e_b) to recover m, r_1, and r_2 from (u, v), and then verifies four independent modular-exponentiation equations against the sender's public key $(K_{11,A}, K_{12,A})$ to confirm authenticity. Successful verification guarantees both the integrity of m and the sender's identity. The correctness conditions on the key parameters ensure that the recovery of m, r_1, and r_2 is exact.

III. PQES-Based Post-Quantum Two-Round Authenticated Key Exchange (PQES-AKE)

A. Overview

We propose PQES-AKE, a novel two-round authenticated key exchange protocol that leverages PQES to achieve mutual authentication, session secrecy, and forward secrecy in a post-quantum setting. The core idea is to mask each party's ephemeral Diffie-Hellman (DH) public key with the randomness (r_1, r_2) produced internally by PQES, so that an eavesdropper who intercepts the transcript cannot separate the DH component from the signcryption without first breaking the PQES scheme. Session keys are derived from both the DH shared secret and the signcryption transcript, combining classical forward secrecy with post-quantum authentication.

Throughout this protocol, the values $X_A = g^{x_A} \bmod p$ and $X_B = g^{x_B} \bmod p$ are referred to as ephemeral DH keys, borrowing standard Diffie-Hellman terminology. However, unlike classical DH, these values are never transmitted in the clear. Publishing X_A or X_B directly would expose the scheme to quantum adversaries, who could recover the discrete logarithm x_A or x_B via Shor's algorithm and thereby compute the shared secret. To prevent this, X_A and X_B are always sent in masked form:

$$M = r_1 \cdot X_A + r_2 \bmod p, \qquad M' = r_2 \cdot X_B + r_1 \bmod p,$$

where (r_1, r_2) are secret, session-fresh values known only to the party who has successfully performed PQES unsigncryption. The masks M and M' are computationally indistinguishable from random elements of $\mathbb{Z}_p$ to any adversary that cannot first break the underlying PQES scheme. In this sense, the word "public" in "DH public key" is a naming convention inherited from the classical protocol; in PQES-AKE, no DH key material is ever made public.

B. Protocol Parameters

- p: a large prime defining the DH and LWE group.
- p': a secondary prime governing signature exponents.
- $g \in \mathbb{Z}_p^*$: a publicly known generator.
- (SK_A, PK_A), (SK_B, PK_B): long-term PQES key pairs of Alice and Bob, respectively.
- $x_A, x_B \in_R [g, p')$: ephemeral DH secrets chosen fresh per session.
- $X_A = g^{x_A} \bmod p$, $X_B = g^{x_B} \bmod p$: ephemeral DH public keys.
- r_1, r_2: ephemeral randomness produced internally by PQES_Enc&Sig and recovered by PQES_Dec&Verf.
- $m \in [2, g)$: a fresh random nonce chosen by Alice, authenticated by PQES.
- h_{X_A}, h_{X_B}: SHA commitments to the ephemeral DH keys, transmitted for integrity binding.
- KDF: a key derivation function (e.g. HKDF-SHA-256).

C. Algorithm Descriptions

Algorithm 1: AUTHA — *Alice's Initiator Step:* Input: Public parameters (p, p', g); Alice's private PQES key $(k_{1,A}, k_{2,A})$; Bob's public PQES key (a_B, b_B).
Output: PQES ciphertext/signature (u, v, s_1, s_2); masked DH key M; commitment h_{X_A}; ephemeral secrets x_A, r_1, r_2 (kept local).

1) Sample a fresh ephemeral DH secret $x_A \xleftarrow{R} [g,\ p')$.
2) Compute the ephemeral DH public key $X_A = g^{x_A} \bmod p$.
3) Sample a fresh random nonce $m \xleftarrow{R} [2,\ g)$.
4) Run PQES signcryption:
$$(u,\ v,\ s_1,\ s_2,\ r_1,\ r_2) \leftarrow \text{PQES_Enc\&Sig}(m,\ PK_B,\ SK_A).$$
5) Mask the ephemeral DH public key using (r_1, r_2):
$$M = (r_1 \cdot X_A + r_2) \bmod p.$$
6) Compute a hash commitment to X_A:
$$h_{X_A} = \text{SHA}(X_A).$$
7) Send to Bob: $(u,\ v,\ s_1,\ s_2,\ M,\ h_{X_A})$.
Retain locally: $(x_A,\ r_1,\ r_2)$.

Algorithm 2: AUTHB — *Bob's Responder Step:* Input: Received tuple $(u, v, s_1, s_2, M, h_{X_A})$; public parameters (p, p', g); Bob's private PQES key (s_B, e_B); Alice's public PQES key $(K_{11,A}, K_{12,A})$.
Output: Masked DH key M'; commitment h_{X_B}; local DH value DH_B.

1) Run PQES unsigncryption:

$$(m',\ t,\ r_1',\ r_2') \ \leftarrow\ \text{PQES_Dec\&Verf}(u,\ v,\ s_1,\ s_2,\ SK_B,\ PK_A).$$

If $t \neq 1$, abort (authentication failure).
2) Unmask Alice's ephemeral DH public key:

$$X_A' \ =\ {r_1'}^{-1} \cdot (M - r_2') \bmod p.$$

3) Verify the commitment: compute $h'_{X_A} = \text{SHA}(X_A')$ and check $h'_{X_A} \stackrel{?}{=} h_{X_A}$. If they differ, abort (replay/substitution detected).
4) Sample a fresh ephemeral DH secret $x_B \xleftarrow{R} [g,\ p')$.
5) Compute $X_B = g^{x_B} \bmod p$.
6) Mask Bob's ephemeral DH key using the recovered (r_1', r_2'):

$$M' \ =\ (r_2' \cdot X_B \ +\ r_1') \bmod p.$$

7) Compute a hash commitment to X_B:

$$h_{X_B} \ =\ \text{SHA}(X_B).$$

8) Compute the local DH value:

$$DH_B \ =\ {X_A'}^{x_B} \bmod p.$$

9) Send to Alice: $(M',\ h_{X_B})$.

Algorithm 3: AUTHA′ — *Alice's Finalisation Step:* Input: Received tuple (M', h_{X_B}); local state (x_A, r_1, r_2); public parameter p.
Output: Local DH value DH_A.

1) Unmask Bob's ephemeral DH public key:

$$X_B' \ =\ r_2^{-1} \cdot (M' - r_1) \bmod p.$$

2) Verify the commitment: compute $h'_{X_B} = \text{SHA}(X_B')$ and check $h'_{X_B} \stackrel{?}{=} h_{X_B}$. If they differ, abort.
3) Compute the local DH value:

$$DH_A \ =\ {X_B'}^{x_A} \bmod p.$$

D. Pseudocode

The following pseudocode listings present the three algorithms in a compact, implementation-oriented form. The notation $a \xleftarrow{R} [l, h)$ denotes uniform random sampling from the integer interval $[l, h)$.

Algorithm 1 AUTHA — Alice's Initiator Step

Require: Public parameters $(p,\ p',\ g)$; Alice's PQES private key $SK_A = (k_{1,A},\ k_{2,A})$; Bob's PQES public key $PK_B = (a_B,\ b_B)$
Ensure: Ciphertext/signature (u, v, s_1, s_2); masked DH key M; commitment h_{X_A}; local ephemeral state $(x_A,\ r_1,\ r_2)$

$x_A \xleftarrow{R} [g,\ p')$
$X_A \leftarrow g^{x_A} \bmod p$
$m \xleftarrow{R} [2,\ g)$
$(u,\ v,\ s_1,\ s_2,\ r_1,\ r_2) \leftarrow$ PQES_Enc&Sig$(m,\ PK_B,\ SK_A)$
$M \leftarrow (r_1 \cdot X_A + r_2) \bmod p$
$h_{X_A} \leftarrow \text{SHA}(X_A)$
Send $(u,\ v,\ s_1,\ s_2,\ M,\ h_{X_A})$ to Bob
Retain locally: $(x_A,\ r_1,\ r_2)$

Algorithm 2 AUTHB — Bob's Responder Step

Require: Received message $(u,\ v,\ s_1,\ s_2,\ M,\ h_{X_A})$; public parameters $(p,\ p',\ g)$; Bob's PQES private key $SK_B = (s_B,\ e_B)$; Alice's PQES public key $PK_A = (K_{11,A},\ K_{12,A})$
Ensure: Masked DH key M'; commitment h_{X_B}; local DH value DH_B

$(m',\ t,\ r_1',\ r_2') \leftarrow$ PQES_Dec&Verf$(u,\ v,\ s_1,\ s_2,\ SK_B,\ PK_A)$
if $t \neq 1$ **then**
  Abort ▷ PQES authentication failure
end if
$X_A' \leftarrow {r_1'}^{-} \cdot (M - r_2') \bmod p$
$h'_{X_A} \leftarrow \text{SHA}(X_A')$
if $h'_{X_A} \neq h_{X_A}$ **then**
  Abort ▷ Commitment mismatch: replay or substitution
end if
$x_B \xleftarrow{R} [g,\ p')$
$X_B \leftarrow g^{x_B} \bmod p$
$M' \leftarrow (r_2' \cdot X_B + r_1') \bmod p$
$h_{X_B} \leftarrow \text{SHA}(X_B)$
$DH_B \leftarrow {X_A'}^{x_B} \bmod p$
Send $(M',\ h_{X_B})$ to Alice

Algorithm 3 AUTHA′ — Alice's Finalisation Step

Require: Received message $(M',\ h_{X_B})$; local ephemeral state $(x_A,\ r_1,\ r_2)$; public parameter p
Ensure: local DH value DH_A

$X_B' \leftarrow r_2^{-} \cdot (M' - r_1) \bmod p$
$h'_{X_B} \leftarrow \text{SHA}(X_B')$
if $h'_{X_B} \neq h_{X_B}$ **then**
  Abort ▷ Commitment mismatch
end if
$DH_A \leftarrow {X_B'}^{x_A} \bmod p$

E. Protocol Flow

The three algorithms compose a two-round protocol as follows.
After both algorithms complete, $DH_A = DH_B = g^{x_A x_B} \bmod p$ is the shared Diffie-Hellman secret.

F. Session Key Derivation

Both parties derive the final session key by feeding the DH output together with the signcryption transcript and long-term public keys into a KDF:

$$Z \;=\; \text{KDF}(DH \parallel m \parallel PK_A \parallel PK_B \parallel (u, v) \parallel (s_1, s_2))$$

where $DH = DH_A = DH_B$. The inclusion of the ciphertext (u, v) and signature (s_1, s_2) in the KDF input binds the session key to the authenticated transcript, preventing key confirmation attacks. The fresh random nonce m prevents cross-session key reuse and strengthens the KDF input entropy.

G. Correctness

The masking and unmasking operations are summarised below.
Alice masks X_A: $M = r_1 \cdot X_A + r_2 \pmod p$
Bob recovers X_A: $X_A = r_1^{-1}(M - r_2) \pmod p$
Bob masks X_B: $M' = r_2 \cdot X_B + r_1 \pmod p$ Alice recovers X_B: $X_B = r_2^{-1}(M' - r_1) \pmod p$

The roles of r_1 and r_2 are swapped between the two masking operations, so each party uses a different linear combination of the same PQES randomness. The values r_1 and r_2 remain secret from any party that has not performed PQES unsigncryption, and they are fresh per session because PQES draws its internal randomness anew at each invocation.

Proposition 1 (Correctness of PQES-AKE)**.** *If both parties hold authentic long-term PQES key pairs and PQES Dec&Verf succeeds (i.e. $t = 1$), then $DH_A = DH_B$.*

Proof. Bob recovers $(r_1', r_2') = (r_1, r_2)$ via PQES unsigncryption. He then computes $X_A' = r_1^{-1}(M - r_2) = r_1^{-1}(r_1 X_A + r_2 - r_2) = X_A$. Alice computes $X_B' = r_2^{-1}(M' - r_1) = r_2^{-1}(r_2 X_B + r_1 - r_1) = X_B$. Hence $DH_A = X_B^{x_A} = (g^{x_B})^{x_A} = g^{x_A x_B} = (g^{x_A})^{x_B} = X_A^{x_B} = DH_B$. □

IV. Security Analysis

A. Hardness Assumptions

The security of PQES-AKE rests on three hardness assumptions, the first two inherited from the underlying PQES scheme and the third specific to the key structure.

Assumption 1 (Learning With Errors (LWE))**.** *Let p be a large prime and let s, e be secret scalars satisfying the PQES correctness conditions. Given $a \in \mathbb{Z}_p$ and $b = as + e \bmod p$, no probabilistic polynomial-time (PPT) adversary can recover s or e with non-negligible probability. The computational difficulty follows from the hardness of the Learning With Errors problem over $\mathbb{Z}_p$.*

Assumption 2 (Secret-Base Discrete Exponentiation (SBDE))**.** *Let p' be a large prime. Given $K_{11} = k_1^{k_1} \bmod p'$ and $K_{12} = k_1^{k_2} \bmod p'$, where $k_1, k_2 \xleftarrow{R} \mathbb{Z}_{p'}^*$ are secret, no PPT adversary can recover k_1 or k_2 with non-negligible probability.*

This assumption is strictly harder than the standard Discrete Logarithm (DL) assumption, in which the base is a publicly known generator and only the exponent is secret. In SBDE, both the base k_1 and the exponents k_1 and k_2 are unknown. Consequently, Shor's quantum algorithm does not apply; Shor's reduction to the hidden subgroup problem on $\mathbb{Z}$ requires a fixed, known base, and that structure is absent here. We conjecture that SBDE is hard against both classical and quantum adversaries.

It is worth noting that computing $\delta = k_2^- k_1$ in $K_{11} = K_{12}^{\delta}$ is of no use to the attacker, neither for compromising the encryption nor for forging the signature.

Assumption 3 (Collision Resistance of SHA-256/512 (CR-Hash))**.** *No PPT adversary can find two distinct inputs $x \neq x'$ such that $\text{SHA-256/512}(x) = \text{SHA-256/512}(x')$ with non-negligible probability.*

B. Security Model

We adopt a standard Bellare-Rogaway-style model for a two-party authenticated key exchange [18], extended to a post-quantum adversarial setting.

A finite set of parties $\mathcal{P} = \{A, B, \ldots\}$ hold long-term PQES key pairs. Each invocation of the protocol by a party P is called a session instance Π_P^i and maintains a local state comprising ephemeral secrets, received messages, and a session-key register. A session instance is accepted once it has completed all steps without aborting and has computed a session key Z.

A probabilistic polynomial-time adversary $\mathcal{A}$ controls the network and may issue the following oracle queries:

- $\mathsf{Send}(\Pi_P^i, \mu)$: deliver message μ to session instance Π_P^i and receive its response.
- $\mathsf{Reveal}(\Pi_P^i)$: obtain the session key of an accepted instance.
- $\mathsf{Corrupt}(P)$: obtain the long-term private key of party P.
- $\mathsf{Test}(\Pi_P^i)$: receive either the real session key or a uniformly random key (decided by a hidden bit b); this query is allowed at most once, on a fresh session.

A session instance Π_P^i is fresh at the time of a Test query if (i) neither Π_P^i nor its matching peer instance has been subject to a Reveal query, and (ii) the long-term keys of neither communicating party have been obtained via Corrupt before the session was initiated.

a) AKE security (SK-security).: The protocol is AKE-secure (SK-secure) if no PPT adversary $\mathcal{A}$ can guess the hidden bit b in a Test query with advantage non-negligibly greater than $\frac{1}{2}$, over all fresh sessions and for all polynomially bounded numbers of oracle queries.

C. Informal Security Analysis

We discuss informally how PQES-AKE achieves each of its claimed security properties.

TABLE I
2-ROUND

Alice		Bob
$(u, v, s_1, s_2, M, h_{X_A}, x_A, r_1, r_2) \leftarrow \text{AUTHA}(\cdot)$	$\xrightarrow{(u, v, s_1, s_2, M, h_{X_A})}$	$(M', h_{X_B}, DH_B) \leftarrow \text{AUTHB}(\cdot)$
	$\xleftarrow{(M', h_{X_B})}$	
$DH_A \leftarrow \text{AUTHA}'(\cdot)$		

1) Mutual Authentication: Alice authenticates to Bob by virtue of the PQES signcryption: the ciphertext (u, v) and signature (s_1, s_2) can only be produced by a party holding Alice's private key $(k_{1,A}, k_{2,A})$, and their validity is confirmed by Bob through four independent verification equations involving Alice's public key $(K_{11,A}, K_{12,A})$. An impersonator without SK_A cannot produce a valid (u, v, s_1, s_2) tuple for an arbitrary message m under Bob's public key, because doing so would require solving either the LWE problem (to forge the ciphertext) or the SBDE problem (to forge the signature).

Bob authenticates to Alice implicitly: Alice's ephemeral DH key X_A is masked by (r_1, r_2) values that are embedded inside the PQES ciphertext and recoverable only by a party that has successfully decrypted and verified the signcryption. Bob's reply $M' = r_2 \cdot X_B + r_1 \bmod p$ uses exactly those recovered values. If an active adversary had intercepted the first message and fabricated M', it would not know the correct (r_1, r_2), so Alice's commitment check $h'_{X_B} \stackrel{?}{=} h_{X_B}$ would fail, and Alice would abort.

2) Session Key Secrecy: The session key $Z = \text{KDF}(DH \| m \| PK_A \| PK_B \| \cdots)$ is a function of the shared DH value $DH = g^{x_A x_B} \bmod p$. A passive eavesdropper observing the transcript (u, v, s_1, s_2, M, M') obtains no information about DH because:

1) The ephemeral DH keys X_A and X_B are never transmitted in the clear; they appear only inside the affine masks M and M'.
2) Unmasking requires knowledge of (r_1, r_2), which in turn requires breaking the PQES scheme (LWE or SBDE hardness).
3) Even if an adversary recovered X_A and X_B, computing $g^{x_A x_B}$ from g^{x_A} and g^{x_B} alone requires solving the Computational Diffie-Hellman (CDH) problem in $\mathbb{Z}_p^*$.

Thus session key secrecy reduces to the joint hardness of PQES security (LWE + SBDE) and CDH.

3) Forward Secrecy: Forward secrecy ensures that the compromise of long-term keys does not expose past session keys. In PQES-AKE, the session key Z depends critically on the ephemeral secrets x_A and x_B, which are chosen fresh per session and discarded after DH is computed. Even if an adversary later obtains SK_A and SK_B, it cannot recompute DH without knowing x_A or x_B. Recovering the ephemeral secrets from the transcript requires unmasking M and M', which needs (r_1, r_2); recovering (r_1, r_2) from the ciphertext without the long-term private keys requires breaking PQES; and even given X_A, X_B, computing $g^{x_A x_B}$ requires solving CDH. The forward secrecy of PQES-AKE therefore follows from the ephemeral nature of x_A, x_B combined with CDH hardness.

4) Resistance to Replay Attacks: The PQES signcryption randomness (r_1, r_2) is drawn fresh at each invocation of PQES_Enc&Sig, making every ciphertext tuple (u, v, s_1, s_2) unique even for the same message m. Additionally, the hash commitments h_{X_A} and h_{X_B} bind each flow to the specific ephemeral DH keys of that session. An adversary replaying a recorded first message would cause Bob to recover the same (r_1, r_2) and the same X_A, but the ephemeral x_B chosen by Bob in the new session would differ, producing a different DH_B. Alice's session state (x_A, r_1, r_2) from the original session would not match, so the protocol would not complete or would produce a different session key, effectively neutralising the replay.

5) Resistance to Man-in-the-Middle Attacks: An active adversary $\mathcal{A}$ attempting a man-in-the-middle attack must either: (a) produce a valid PQES signcryption under Alice's key without holding SK_A, which requires breaking LWE or SBDE; or (b) intercept the first flow, strip the signcryption, re-signcrypt a modified message under its own key, and relay the result, but then Bob would verify against $\mathcal{A}$'s public key rather than Alice's, and Alice's public key would not match, so the session keys would diverge; or (c) substitute a different masked value $\tilde{M}$ or $\tilde{M}'$, but the hash commitments h_{X_A} and h_{X_B} would detect any substitution of X_A or X_B, causing an abort.

None of these attack paths succeeds under the stated hardness assumptions.

6) Post-Quantum Security of the Masking: As mentioned, the ephemeral DH keys $X_A = g^{x_A}$ and $X_B = g^{x_B}$ are concealed inside the affine masks M and M'. A quantum adversary running Shor's algorithm would need access to the raw values X_A or X_B as a starting point for the discrete logarithm computation. Since these values are not available in the transcript, Shor's algorithm cannot be applied. The post-quantum security of the masking therefore reduces entirely to the post-quantum security of PQES (LWE + SBDE), both of which are conjectured to be quantum-resistant.

D. Formal Security Analysis

We now state and prove the main security theorem for PQES-AKE.

Definition 1 (PQES Security)**.** *The PQES scheme is IND-CCA2 and EUF-CMA (Existential Unforgeability under Cho-*

sen Message Attack) secure if no PPT adversary can distinguish encryptions of two chosen plaintexts after adaptively querying a decryption oracle (IND-CCA2), or produce a valid ciphertext-signature pair for a message not previously submitted to a signing oracle (EUF-CMA) with non-negligible advantage. The security of PQES reduces to Assumptions 1 and 2.

Definition 2 (CDH Assumption in $\mathbb{Z}_p^*$)**.** *Given $g, g^a, g^b \in \mathbb{Z}_p^*$ for uniformly random a, b, no PPT adversary can compute $g^{ab} \bmod p$ with non-negligible probability.*

Theorem 1 (SK-Security of PQES-AKE)**.** *Let $\mathcal{A}$ be a PPT adversary against the SK-security of PQES-AKE that makes at most q_s* Send *queries, q_r* Reveal *queries, and one* Test *query against n parties running at most ℓ sessions each. If PQES is IND-CCA2 and EUF-CMA secure (Definition 1), the CDH assumption holds in $\mathbb{Z}_p^*$ (Definition 2), and* SHA-256/512 *is collision resistant (Assumption 3), then:*

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{AKE}} \;\le\; \frac{1}{2} \;+\; 2n\ell\,\epsilon_{\mathrm{PQES}} \;+\; n^2\ell^2\,\epsilon_{\mathrm{CDH}} \;+\; q_s\,\epsilon_{\mathrm{Hash}},$$

where ϵ_{PQES}, ϵ_{CDH}, and ϵ_{Hash} are the respective advantages of the best PPT adversary against PQES, CDH, and SHA-256/512.

Proof. We proceed via a sequence of games, starting from the real AKE security game and ending in a game where the adversary's advantage is at most $\frac{1}{2}$.

Game G_0: The original AKE security game. The adversary $\mathcal{A}$ interacts with honest parties via the oracle queries defined in Section IV-B. By definition, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{AKE}} = \Pr[\mathcal{A} \text{ wins } G_0] - \frac{1}{2}$.

Game G_1: Abort if PQES forgery occurs. We modify G_0 so that the simulator aborts and outputs a random bit if $\mathcal{A}$ causes any session to accept a PQES ciphertext/signature tuple (u, v, s_1, s_2) that was not produced by the legitimate sender. By the EUF-CMA security of PQES (Definition 1), any such forgery yields a PPT adversary against PQES with advantage ϵ_{PQES}. Over n parties and ℓ sessions:

$$|\Pr[\mathcal{A} \text{ wins } G_1] - \Pr[\mathcal{A} \text{ wins } G_0]| \;\le\; 2n\ell\,\epsilon_{\mathrm{PQES}}.$$

Game G_2: Abort if hash commitment collision occurs. We further modify the game to abort if $\mathcal{A}$ causes a commitment check $h'_X = h_X$ to pass with $X' \neq X$ (i.e., a SHA-256/512 collision). By Assumption 3, this event occurs with probability at most $q_s\,\epsilon_{\mathrm{Hash}}$:

$$|\Pr[\mathcal{A} \text{ wins } G_2] - \Pr[\mathcal{A} \text{ wins } G_1]| \;\le\; q_s\,\epsilon_{\mathrm{Hash}}.$$

In G_2, any session that accepts has correctly authenticated both parties' ephemeral DH keys X_A and X_B.

Game G_3: Replace DH with a random group element. In sessions involving the Test query, we replace the honest DH value $g^{x_A x_B}$ with a uniformly random element $R \xleftarrow{R} \mathbb{Z}_p^*$. The IND-CCA2 security of PQES ensures that (r_1, r_2), and hence the unmasked DH keys, remain computationally hidden from $\mathcal{A}$. Combined with the CDH assumption over $\mathbb{Z}_p^*$, any PPT distinguisher for this replacement can be turned into a CDH solver:

$$|\Pr[\mathcal{A} \text{ wins } G_3] - \Pr[\mathcal{A} \text{ wins } G_2]| \;\le\; n^2\ell^2\,\epsilon_{\mathrm{CDH}}.$$

Game G_4: Replace session key with a random key. Since $Z = \mathrm{KDF}(DH \| \cdots)$ and DH is now a uniformly random group element (from G_3), the output of KDF is uniformly distributed and independent of the transcript. The Test oracle, therefore, returns either the real Z or a fresh random key, and these are identically distributed from $\mathcal{A}$'s view:

$$\Pr[\mathcal{A} \text{ wins } G_4] = \frac{1}{2}.$$

Combining the bounds across all game transitions:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{AKE}} \;\le\; \frac{1}{2} + 2n\ell\,\epsilon_{\mathrm{PQES}} + n^2\ell^2\,\epsilon_{\mathrm{CDH}} + q_s\,\epsilon_{\mathrm{Hash}},$$

which is negligible under Assumptions 1, 2, 3, and the CDH assumption. □

Theorem 2 (Forward Secrecy of PQES-AKE)**.** *PQES-AKE provides perfect forward secrecy: the session key Z of any completed, fresh session remains computationally indistinguishable from random even after the long-term private keys SK_A and SK_B are fully revealed to the adversary, provided the adversary did not actively interfere with that session.*

Proof. Let $\mathcal{A}$ obtain SK_A and SK_B after a session has been completed. The session key is $Z = \mathrm{KDF}(g^{x_A x_B} \| \cdots)$. Computing $g^{x_A x_B}$ from the transcript requires recovering either x_A from X_A or x_B from X_B. The values X_A and X_B are not present in the transcript; only their masks $M = r_1 X_A + r_2 \bmod p$ and $M' = r_2 X_B + r_1 \bmod p$ are transmitted. Unmasking requires (r_1, r_2), which are recoverable from the PQES ciphertext only using SK_B (for r_1, r_2) together with SK_A (to verify the signature). However, even if $\mathcal{A}$ now holds both SK_A and SK_B and can thus recover X_A and X_B, computing $g^{x_A x_B}$ from g^{x_A} and g^{x_B} alone remains a CDH instance in $\mathbb{Z}_p^*$. Since the ephemeral secrets x_A and x_B are erased after the session, $\mathcal{A}$ cannot compute Z without solving CDH. □

Proposition 2 (Mutual Authentication)**.** *Under the EUF-CMA security of PQES and the collision resistance of* SHA-256/512*, PQES-AKE provides mutual authentication. If two session instances Π_A^i and Π_B^j both accept and agree on a session key Z, then each instance was genuinely interacting with the intended peer.*

Proof. Alice authenticates to Bob: Bob accepts only after PQES_Dec&Verf returns $t = 1$, which certifies that (u, v, s_1, s_2) was produced using SK_A. By EUF-CMA security of PQES, no party other than Alice can forge such a tuple with non-negligible probability.

Bob authenticates to Alice: Alice verifies $h'_{X_B} = h_{X_B}$. The value $M' = r_2 X_B + r_1 \bmod p$ can only be correctly formed by a party that knows (r_1, r_2). These are embedded in the PQES ciphertext and recoverable only by a party

holding SK_B that has successfully unsigncrypted Alice's first message. By the IND-CCA2 security of PQES, (r_1, r_2) are computationally hidden from any party not holding SK_B. An adversary substituting a different $\tilde{M}'$ would produce a different $\tilde{X}_B$ and therefore a different hash $h_{\tilde{X}_B} \neq h_{X_B}$, causing Alice to abort. □

E. Comparison with Classical AKE Protocols

Table II summarises the security properties of PQES-AKE in comparison with a classical ephemeral Diffie-Hellman (ECDH)-based AKE and a generic lattice-based KEM-then-authenticate construction.

TABLE II
SECURITY PROPERTY COMPARISON.

Property	ECDH-AKE	Lattice-KEM+Auth	PQES-AKE
Mutual authentication	✓	✓	✓
Session key secrecy	✓	✓	✓
Forward secrecy	✓	✓	✓
Post-quantum secure	✗	✓	✓
DH keys hidden in transit	✗	N/A	✓
Quantum-safe masking	✗	N/A	✓
Integrated signcryption	✗	✗	✓
Single-primitive design	✗	✗	✓

V. Performance Evaluation

A. Test Environment

The prototype implementation of PQES-AKE was written in Python and executed on a commodity laptop with the following hardware and software configuration.

TABLE III
TEST ENVIRONMENT SPECIFICATION.

Component	Specification
Processor	Intel Core i7
RAM	16 GB
Storage	SSD
Operating system	Windows 11 64-bit
Language / runtime	Python 3 (standard library only)
Arithmetic library	Python built-in arbitrary-precision integers
Parameter p	995-bit prime
Parameter p'	257-bit prime
Generator g	200-bit integer
Ephemeral DH secrets	$x_A, x_B \in [g,\ p')$, i.e. ≈ 256 bits
Hash function	SHA-256 (hashlib)

Timing was measured using Python's `time.perf_counter()`, which provides sub-microsecond resolution on the test platform. The reported figure covers the complete protocol execution: Algorithm 1 (AUTHA), Algorithm 2 (AUTHB), and Algorithm 3 (AUTHA′), including all modular exponentiations, hash evaluations, masking, and unmasking operations. Network transmission time is excluded, as the benchmark targets computational cost only.

TABLE IV
PQES-AKE EXECUTION TIME (100 INDEPENDENT RUNS).

Metric	Value (seconds)
Average total execution time	6.2×10^{-3}
Number of test runs	100
Approximate per-algorithm breakdown (estimated)	
AUTHA (Alice initiates)	$\sim 2 \times 10^{-3}$
AUTHB (Bob responds)	$\sim 3 \times 10^{-3}$
AUTHA′ (Alice finalises)	$\sim 0.2 \times 10^{-3}$

B. Experimental Results

The protocol was executed 100 times under identical conditions. Table IV reports the observed timing.
The dominant cost is borne by AUTHB, which performs the full PQES unsigncryption (eight modular exponentiations in $\mathbb{Z}_{p'}$). AUTHA performs PQES signcryption (five modular exponentiations in $\mathbb{Z}_{p'}$). AUTHA′ performs a single DH exponentiation and is therefore the lightest of the three steps.

An average total time of $\mathbf{6.2}$ ms places PQES-AKE well within the performance envelope of practical authentication protocols. For reference, TLS 1.3 handshake computational costs on comparable hardware are typically reported in the range of 2–10 ms for the cryptographic operations alone, and NIST-standardized post-quantum KEMs such as Kyber-768 complete key encapsulation in under 1 ms on optimized implementations. The additional cost of PQES-AKE relative to a bare KEM is attributable to the integrated signcryption, which simultaneously provides confidentiality, authentication, and the masking of ephemeral DH keys, operations that would otherwise require separate primitives.

C. Computational Complexity Analysis

We analyse the asymptotic and concrete computational complexity of each algorithm in terms of the dominant operation: modular exponentiation using the square-and-multiply method, whose cost for an n-bit exponent and an ℓ-bit modulus is $O(n)$ multiplications of ℓ-bit integers, each costing $O(\ell^2)$ bit operations, for a total of $O(n\ell^2)$ bit operations per exponentiation.

In PQES-AKE, all exponentiations in the signature component use the p'-domain ($\ell \approx 257$ bits, $n \leq p' \approx 257$ bits), while DH exponentiations use the p-domain ($\ell \approx 995$ bits, $n \leq p' \approx 257$ bits because ephemeral secrets are drawn from $[g, p')$).

TABLE V
OPERATIONS IN AUTHA.

Operation	Count	Domain
Modular exponentiation (PQES signcryption)	5	$\mathbb{Z}_{p'}$
DH exponentiation g^{x_A}	1	$\mathbb{Z}_p$
SHA-256 hash evaluations	3	—
Modular multiplications / additions	$O(1)$	$\mathbb{Z}_p$

1) Algorithm AUTHA *(Alice):* Total cost of AUTHA:

$$T_{\text{AUTHA}} = 5 \cdot C_{\exp}(p') + 1 \cdot C_{\exp}(p) + 3 \cdot C_{\text{hash}} + O(C_{\text{mul}}),$$

where $C_{\exp}(q)$ denotes the cost of one modular exponentiation modulo q.

TABLE VI
OPERATIONS IN AUTHB.

Operation	Count	Domain
PQES decryption (division, modinv, modmul)	$O(1)$	$\mathbb{Z}_p$
PQES verification exponentiations	8	$\mathbb{Z}_{p'}$
DH exponentiation g^{x_B}	1	$\mathbb{Z}_p$
DH exponentiation $X_A^{x_B}$	1	$\mathbb{Z}_p$
SHA-256 hash evaluations	4	—
Modular inverse (extended GCD)	1	$\mathbb{Z}_p$

2) Algorithm AUTHB *(Bob):* Total cost of AUTHB:

$$T_{\text{AUTHB}} = 8{\cdot}C_{\exp}(p')+2{\cdot}C_{\exp}(p)+4{\cdot}C_{\text{hash}}+O(C_{\text{mul}}+C_{\text{gcd}}).$$

TABLE VII
OPERATIONS IN AUTHA′.

Operation	Count	Domain
DH exponentiation $X_B^{x_A}$	1	$\mathbb{Z}_p$
SHA-256 hash evaluation	1	—
Modular inverse + multiplication	1	$\mathbb{Z}_p$

3) Algorithm AUTHA′ *(Alice Finalises):* Total cost of AUTHA′:

$$T_{\text{AUTHA'}} = 1 \cdot C_{\exp}(p) + C_{\text{hash}} + O(C_{\text{mul}}).$$

Therefore, the total protocol complexity is:

$$T_{\text{PQES-AKE}} = 13{\cdot}C_{\exp}(p')+4{\cdot}C_{\exp}(p)+8{\cdot}C_{\text{hash}}+O(C_{\text{mul}}+C_{\text{gcd}}).$$

Using the square-and-multiply complexity formula $C_{\exp}(q) = O(\log q \cdot M(q))$, where $M(q) = O((\log q)^2)$ is the cost of a modular multiplication modulo q:

$$C_{\exp}(q) = O\big((\log q)^3\big) \text{ bit operations.}$$

With $\log_2 p \approx 995$ and $\log_2 p' \approx 257$:

$$13 \cdot C_{\exp}(p') = 13 \cdot O(257^3) \approx 13 \times 1.70 \times 10^7 = O(2.21 \times 10^8) \text{ bit ops,}$$

$$4 \cdot C_{\exp}(p) = 4 \cdot O(995^3) \approx 4 \times 9.85 \times 10^8 = O(3.94 \times 10^9) \text{ bit ops.}$$

The DH exponentiations in $\mathbb{Z}_p$ dominate the total bit-operation count, consistent with p being nearly four times wider than p' (the cubic scaling amplifies this ratio significantly). The overall asymptotic complexity of the full protocol is:

$$T_{\text{PQES-AKE}} = O\big((\log p)^3\big) \text{ bit operations,}$$

which is the same asymptotic class as a standard two-round ephemeral Diffie-Hellman exchange, confirming that the additional PQES signcryption and masking layers introduce only a constant-factor overhead.

4) Communication Complexity: Table VIII summarises the size of each transmitted message.

The total bandwidth of approximately 627 bytes over two rounds is competitive with post-quantum KEM-based handshake schemes; for example, a Kyber-768 key encapsulation message is 1088 bytes, and a Dilithium2 signature is 2420 bytes, so a KEM-then-sign approach at a comparable security level would require roughly 3.5 KB, about four times the bandwidth of PQES-AKE.

D. Complexity Comparison with Related Schemes

Table IX contextualises the computational cost of PQES-AKE against representative authenticated key exchange constructions.

The 6.2 ms average execution time on an unoptimized Python prototype is noteworthy; production implementations typically achieve one to two orders of magnitude of speedup through compiled languages (C/C++, Rust) and hardware acceleration (e.g., Montgomery multiplication and SIMD). A conservative estimate, therefore, places an optimized native implementation of PQES-AKE well below 1 ms on the same hardware class, making it suitable for latency-sensitive applications such as TLS handshakes, IoT device provisioning, and real-time secure messaging.

VI. CONCLUSION

We have presented PQES-AKE, a two-party authenticated key exchange protocol that integrates ephemeral Diffie-Hellman key agreement with PQES signcryption into a unified, two-round protocol. The design addresses a fundamental vulnerability of classical DH-based key exchange in the quantum era, which is the exposure of ephemeral DH public keys in the network transcript, which allows a quantum adversary equipped with Shor's algorithm to recover the shared secret. The protocol achieves many security properties, including mutual authentication, session key secrecy, forward secrecy, and post-quantum resistance. From a practical standpoint, PQES-AKE completes in an average of 6.2 ms on an unoptimized Python prototype running on a commodity Intel Core i7 laptop, with a total communication cost of approximately 627 bytes over two rounds. These figures compare favorably with composite post-quantum handshake constructions (e.g., Kyber + Dilithium at ≈3.5 KB), confirming that the single-primitive design of PQES-AKE yields concrete efficiency benefits alongside its security guarantees. Future work intended to extend PQES-AKE to a group key exchange setting, where $n > 2$ parties establish a common session key.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[2] D. J. Bernstein, "Post-quantum cryptography," in *Encyclopedia of Cryptography, Security and Privacy*. Springer, 2025, pp. 1846–1847.

[3] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[4] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Tech. Rep., 2018.

TABLE VIII
COMMUNICATION COMPLEXITY OF PQES-AKE.

Round	**Direction**	**Fields**	**Approx. size (bits)**
Round 1	Alice $\rightarrow$ Bob	u, v (ciphertext), s_1, s_2 (signature), M, h_{X_A}	$\approx 3 \times 995 + 2 \times 257 + 256$ $= 3755$ bits ≈ 470 bytes
Round 2	Bob $\rightarrow$ Alice	M', h_{X_B}	$995 + 256 = 1251$ bits ≈ 157 bytes
Total			≈ 627 bytes

TABLE IX
COMPUTATIONAL COMPLEXITY COMPARISON (E_q: ONE MODULAR EXPONENTIATION MODULO A q-BIT PRIME; KEM: ONE LATTICE-BASED KEM ENCAPSULATION/DECAPSULATION).

Scheme	**Exp. ops**	**Post-quantum**	**Integrated auth.**	**Hidden DH keys**
Classic DH-AKE (e.g. SIGMA)	$4\,E_{2048}$	✗	✗	✗
ECDH-AKE (e.g. TLS 1.3)	$4\,E_{256}$	✗	✗	✗
KEM + Signature (Kyber + Dilithium)	$O(\mathsf{KEM})$	✓	✗	N/A
PQES-AKE (ours)	$11\,E_{257} + 4\,E_{995}$	✓	✓	✓

[5] A. T. Olutimehin, S. Joseph, A. J. Ajayi, O. C. Metibemu, A. Y. Balogun, and O. O. Olaniyi, "Future-proofing data: Assessing the feasibility of post-quantum cryptographic algorithms to mitigate 'harvest now, decrypt later'attacks," *Decrypt Later'Attacks (February 17, 2025)*, 2025.

[6] R. Chhetri, "Benchmarking post-quantum cryptography on resource-constrained iot devices: Ml-kem and ml-dsa on arm cortex-m0+," *arXiv preprint arXiv:2603.19340*, 2026.

[7] M. IDRASSI, "Kilobyte-bandwidth subliminal channels in fips 204 ml-dsa via packed-commitment embedding," *Cryptology ePrint Archive*, 2026.

[8] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid key encapsulation mechanisms and authenticated key exchange," in *International Conference on Post-Quantum Cryptography*. Springer, 2019, pp. 206–226.

[9] F. Giacon, F. Heuer, and B. Poettering, "Kem combiners," in *IACR International Workshop on Public Key Cryptography*. Springer, 2018, pp. 190–218.

[10] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[11] A. Habib, A. Laouid, and M. Kara, "Secure consensus clock synchronization in wireless sensor networks," in *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*. IEEE, 2021, pp. 1–6.

[12] H. Aissaoua, A. Laouid, M. Kara, A. Bounceur, M. Hammoudeh, and K. Chait, "Integrating homomorphic encryption in iot healthcare blockchain systems," *Ingenierie des Systemes d'Information*, vol. 29, no. 5, p. 1667, 2024.

[13] O. Egbuagha and E. Ikwunna, "Post-quantum cryptography in practice: A literature review of protocol-level transitions and readiness," *Cryptology ePrint Archive*, 2025.

[14] Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) less than cost (signature)+ cost (encryption)," in *Annual international cryptology conference*. Springer, 1997, pp. 165–179.

[15] M. E. Kahla, M. Beggas, A. Laouid, M. Kara, and M. AlShaikh, "Asymmetric image encryption based on twin message fusion," in *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*. IEEE, 2021, pp. 1–5.

[16] Z. Wang, H. Bao, T. Zhao, F. Du, and X. Chen, "New lattice-based short proxy signature scheme with fewer system parameters in the standard model," *IEEE Access*, vol. 10, pp. 57 180–57 191, 2022.

[17] M. Kara, M. Hammoudeh, A. Alamri, and S. Alamri, "A post-quantum public-key signcryption scheme over scalar integers based on a modified lwe structure," *Sensors*, vol. 25, no. 15, p. 4728, 2025.

[18] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 62–73.