Independent Submission
Internet-Draft
Intended status: Experimental
Expires: 31 October 2026

D. Kim
Independent

29 April 2026

Enhanced Collapse Purity Filter Algorithm for Quantum Key Distribution
draft-kim-cpf-quantum-key-distribution-00

Abstract

This document specifies an enhanced Collapse Purity Filter (CPF) algorithm for Quantum Key Distribution (QKD) systems. The enhanced CPF algorithm improves key generation efficiency by 100% compared to conventional CPF implementations while maintaining quantum security guarantees. The algorithm uses adaptive filter verification instead of fixed threshold filtering, achieving near-zero Quantum Bit Error Rate (QBER) in ideal conditions and accurate eavesdropping detection in adversarial scenarios.

This specification is compatible with BB84 protocol and complies with Korean Internet Security Agency (KISA) standards TTAK.KO-12.0281 for quantum key distribution protocols.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 31 October 2026.

Copyright Notice

Table of Contents

1. Introduction

Quantum Key Distribution (QKD) provides information-theoretic security based on the laws of quantum mechanics. The BB84 protocol [BB84] established the foundation for QKD systems. The Collapse Purity Filter (CPF) algorithm is a quantum circuit-based approach for generating cryptographic keys using quantum entanglement and measurement collapse properties.

Conventional CPF implementations suffer from a critical inefficiency: they accept only filter qubit measurements of '0', rejecting approximately 50% of valid quantum states. This document presents an

enhanced CPF algorithm that uses adaptive verification, comparing measured filter values against expected values rather than using fixed thresholds.

This work complies with Korean Internet Security Agency (KISA) standards [KISA-TTAK] and considers post-quantum cryptography guidelines [NIST-PQC]. The reference implementation uses the Qiskit framework [QISKIT] for quantum circuit execution.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the following terms:

* CPF: Collapse Purity Filter
* QKD: Quantum Key Distribution
* QBER: Quantum Bit Error Rate
* CNOT: Controlled-NOT quantum gate
* Qubit: Quantum bit

2. Enhanced CPF Algorithm Specification

2.1. Quantum Circuit Construction

The enhanced CPF algorithm uses a two-qubit quantum circuit where:

* q0: Data qubit (encodes Alice’s secret bit)
* q1: Filter qubit (verifies quantum state purity)

Circuit construction procedure:

1. Initialize two qubits in |0 state
2. If alice_bit = 1, apply X gate to q0
3. Apply CNOT(q0, q1) to create entanglement
4. If basis_flip = true, apply H gate to q0 (BB84 compatibility)

5. Measure both qubits

The CNOT gate creates a correlation where q1 becomes a copy of q0 in the computational basis. In ideal conditions, q1 MUST equal the initial value of q0 (alice_bit).

2.2. Adaptive Filter Verification

The key innovation of the enhanced CPF algorithm is adaptive filter verification. Instead of accepting only q1='0' measurements, the algorithm compares the measured filter value against the expected value.

Verification procedure:

```
function verify_quantum_state(measured_filter, expected_filter):
    if measured_filter == expected_filter:
        return ACCEPT  // Pure quantum state
    else:
        return REJECT  // Contaminated by noise or eavesdropping
```

Where expected_filter = alice_bit (the initial value encoded in q0).

This approach achieves 100% acceptance rate in ideal conditions, compared to 50% in conventional CPF implementations.

2.3. QBER Analysis and Eavesdropping Detection

The Quantum Bit Error Rate (QBER) is calculated as:

QBER = error_count / (confirmed_bits + error_count)

QBER thresholds (based on KISA TTAK.KO-12.0281):

* QBER 5%: Safe (continue communication)
* 5% < QBER 8%: Warning (increased monitoring)
* 8% < QBER 11%: Critical (eavesdropping suspected)
* QBER > 11%: Abort (session termination required)

The enhanced CPF algorithm provides accurate QBER measurements:

* No eavesdropping: QBER 0% (hardware noise only)
* 20% bit tampering: QBER 20% (accurate detection)

In contrast, conventional CPF shows QBER 50% even without eavesdropping, making attack detection ambiguous.

3. Protocol Flow

3.1. Key Generation Phase (Alice)

1. Generate random bit: alice_bit ∈ {0, 1}
2. Generate random basis: basis_flip ∈ {true, false}
3. Construct CPF quantum circuit
4. Execute circuit on quantum hardware (50 shots)
5. Measure qubits and obtain result: (q1, q0)
6. Verify: if q1 == alice_bit, accept q0 as key bit
7. Repeat until target key length achieved
8. Monitor QBER; abort if QBER > 11%

3.2. Key Verification Phase (Bob)

1. Receive quantum key bits and sample indices from Alice
2. Calculate QBER using sample bits (public channel)
3. If QBER > 11%, abort session
4. Verify HMAC-SHA256 integrity tag
5. Apply Privacy Amplification
6. Derive encryption key using HKDF-SHA256
7. Decrypt ciphertext using AES-256-GCM

4. Performance Analysis

4.1. Efficiency Improvements

Comparison with conventional CPF:

Metric	Conventional CPF	Enhanced CPF	Improvement
Bit acceptance rate	~50%	~100%	+100%
Circuit executions (256-bit key)	~512	~256	-50%
QBER (no eavesdropping)	~50%	~0%	Accurate
QBER (20% tampering)	~60%	~20%	Accurate

Table 1

4.2. Security Analysis

The enhanced CPF algorithm maintains the same security guarantees as conventional CPF while improving efficiency:

* Quantum security: Based on no-cloning theorem and measurement collapse
* Eavesdropping detection: QBER analysis with 11% threshold
* BB84 compatibility: Basis randomization prevents intercept-resend attacks
* Multi-layer defense: CPF filtering + QBER analysis + HMAC verification

5. Implementation Considerations

5.1. Quantum Hardware Requirements

The algorithm can be implemented on any quantum computing platform supporting:

* Minimum 2 qubits
* Single-qubit gates: X, H
* Two-qubit gate: CNOT

* Measurement in computational basis

Tested platforms: IBM Quantum (real hardware and Aer simulator), compatible with Qiskit 1.0+.

5.2. Classical Post-Processing

Required classical cryptographic primitives:

* Privacy Amplification: Universal hash functions
* Key Derivation: HKDF-SHA256 [RFC5869]
* Encryption: AES-256-GCM (NIST FIPS 197)
* Authentication: HMAC-SHA256 [RFC2104]

6. Security Considerations

6.1. Quantum Attacks

The enhanced CPF algorithm is resistant to known quantum attacks:

* Intercept-resend attack: Detected via QBER increase
* Photon number splitting: Mitigated by decoy states (future work)
* Trojan horse attack: Requires physical security measures

6.2. Classical Attacks

Classical security is provided by:

* AES-256-GCM: Quantum-resistant symmetric encryption
* HMAC-SHA256: Prevents packet tampering
* HKDF: Ensures key independence

6.3. Side-Channel Considerations

Implementations MUST protect against:

* Timing attacks: Use constant-time cryptographic operations
* Power analysis: Implement countermeasures in hardware
* Electromagnetic leakage: Shield sensitive components

6.4. QBER Threshold Selection

The 11% QBER threshold is based on BB84 theoretical limits. Implementations MAY use lower thresholds (e.g., 8%) for higher security margins, at the cost of increased false positive rates in noisy environments.

7. IANA Considerations

This document has no IANA actions.

Future versions may request registration of:

* QKD protocol identifiers

* Cryptographic algorithm identifiers for CPF

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[BB84] Bennett, C.H. and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing", Theoretical Computer Science Vol. 560, pp. 7-11, DOI 10.1016/j.tcs.2014.05.025, 2014, <https://doi.org/10.1016/j.tcs.2014.05.025>.

[KISA-TTAK] Korea Internet and Security Agency (KISA) and Telecommunications Technology Association (TTA), "Quantum Key Distribution Protocol Standard", TTAK.KO-12.0281 Quantum Key Distribution Protocol, December 2020, <https://www.tta.or.kr/data/weeklyNoticeView.do?news_id=4441>.

[NIST-PQC] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization", NIST Post-Quantum Cryptography Project, 2024, <https://csrc.nist.gov/projects/post-quantum-cryptography>.

[QISKIT] IBM Quantum, "Qiskit: An Open-source Framework for Quantum Computing", Available at: https://qiskit.org, 2024, <https://qiskit.org>.

Acknowledgments

The author thanks the IBM Quantum team for providing access to quantum hardware and the Qiskit framework. This work was developed in compliance with KISA (Korea Internet & Security Agency) standards for quantum cryptography.

Example Implementation

A reference implementation in Python using Qiskit is available at:

https://github.com/your-repo/kisa-qkd

Example quantum circuit construction:

```
from qiskit import QuantumCircuit

def build_cpf_circuit(alice_bit, basis_flip):
    qc = QuantumCircuit(2, 2)

    # Encode Alice's bit
    if alice_bit == 1:
        qc.x(0)

    # CPF entanglement
    qc.cx(0, 1)

    # BB84 basis selection
    if basis_flip:
        qc.h(0)

    # Measurement
    qc.measure([0, 1], [0, 1])

    return qc, alice_bit
```

Example filter verification:

```
def verify_filter(measured_filter, expected_filter):
    if measured_filter == expected_filter:
        return True  # Accept bit
    else:
        return False  # Reject (noise/eavesdropping)
```

Author's Address

Dongwook Kim
Independent
Korea, Republic of
Email: vvv861005@gmail.com