**Optimizing ML-KEM for IoT Devices: Replacing Keccak with Ascon for Reduced Memory Footprint**

**Abstract:**
This work proposes replacing Keccak with Ascon in ML-KEM to optimize post-quantum cryptography for IoT devices. Ascon's lightweight design reduces memory usage and improves efficiency, addressing IoT constraints like limited RAM and processing power. The research demonstrates significant memory savings while maintaining IND-CCA2 security.

---

# 1. INTRODUCTION

- **Context Evolution**
  - **PQC Emergence**: Quantum computing threats → Need for quantum-resistant algorithms
  - **NIST Standardization**: Kyber selected → ML-KEM as standard
  -

  **Problem Statement**

  - PQC applications: various
  - **IoT Adoption**: growth Need for lightweight, efficient PQC
  - **Howerver, IoT Constraints**:
    - Limited RAM, processing power, energy efficiency
    - Memory footprint restrictions → Critical for constrained devices
  - **ML-KEM Challenges**:
  - Performance bottleneck for constrained devices → Hash operations
    - Keccak’s 1600-bit state → High memory usage
    - Memory bandwidth → Implementation inefficiency

  **Research Objectives**

  - Proposal: replace Keccak with much lighter hash function, i.e., Ascon
  - Introduction of Ascon: standardized lightweight block cipher, but can be used as hash function as well.
  - Contribution of this work
    - **Primary**: Optimize ML-KEM for IoT devices
      - Algorithm optimizations for constrained devices
    - **Secondary**: Comprehensive performance analysis
      - Reduce memory footprint
    - **Third**: Mathmetical security proof
      - Maintain IND-CCA2 security guarantees

---

# 2. PRELIMINARIES

### 2.1 ML-KEM Overview

- **Parameters:** n=256, q=3329, k=3 (balance security and efficiency).
- **Core Algorithms:** Key generation, encryption, decryption.
- **Hash Functions:** SHA3-512, SHA3-256, SHAKE128, SHAKE256 (Keccak-based).

### 2.2 Keccak Limitations

- **State Size:** 1600-bit → High memory usage.
- **Cache Efficiency:** Suboptimal for small devices.
- **Memory Bandwidth:** Large operations slow down performance.

### 2.3 Ascon Advantages

- **Lightweight Design:** 320-bit state size (80% smaller than Keccak).
- **Standardization:** NIST-selected for lightweight cryptography.
- **Efficiency:** Better suited for IoT devices.

---

## 3. PROPOSED METHODOLOGY

### 3.1 Ascon Integration

- **Replacement Strategy:** Replace SHA3/SHAKE with Ascon primitives.
- **Function Mapping:**
  - SHA3-512 → Ascon-Hash256 (2x).
  - SHA3-256 → Ascon-Hash256.
  - SHAKE128 → Ascon-XOF128.
  - SHAKE256 → Ascon-CXOF128.

### 3.2 Ascon-Based Algorithms

- **Ascon-XOF128:** Matrix generation, randomization.
- **Ascon-CXOF128:** Noise sampling, domain separation.
- **Ascon-Hash256:** Key derivation, verification.

---

## 4. EXPERIMENTAL RESULTS

### 4.1 Performance Metrics

- **Latency:**
  - KeyGen: ↑19.4%
  - Encaps: ↑17.9%
  - Decaps: ↑13.1%
- **Resource Utilization:**
  - RAM: ↓12% (Keccak: 1,232B → Ascon: 1,096B).

- Flash: ↓36% (Keccak: 43,389B → Ascon: 31,911B).
- Cache Efficiency: Improved (reads ↓33%, writes ↓32% less than Keccak).

### 4.2 Trade-offs

- **Latency Increase:** Ascon requires more rounds due to the fix rate but compensates with better memory efficiency.
- **Overall:** Memory gains outweigh latency costs, making Ascon suitable for IoT.

---

## 5. CONCLUSIONS

- **Key Findings:**

  1. RAM usage reduced by 12%, Flash by 36%.
  2. IND-CCA2 security guarantees maintained.
  3. Latency increase offset by memory and cache efficiency gains.

- **Future Work:**

  1. Benchmark on IoT platforms (Cortex-M, RISC-V).
  2. Analyze energy consumption for IoT suitability.
  3. Matrix Generation can be optimized to improve latency