Post-Quantum DNSSEC Architecture: A Quantum-Resistant Framework for DNS Security

1 Introduction

The Domain Name System Security Extensions (DNSSEC) provide cryptographic authentication and integrity for DNS data, protecting against attacks like cache poisoning. However, the advent of quantum computing threatens traditional cryptographic algorithms (e.g., RSA, ECDSA) used in DNSSEC, as they can be broken by quantum algorithms like Shor's algorithm. To address this, we propose a **Post-Quantum DNSSEC Architecture** that replaces these algorithms with quantum-resistant alternatives:

- **Zone Signing Key (ZSK)**: Uses **Falcon-512**, a lattice-based signature scheme, to sign DNS resource record sets (RRsets).
- **Key Signing Key (KSK)**: Uses **Falcon-512** within a **Merkle Tree**, a hash-based structure, to authenticate the KSK public key and sign the DNSKEY RRset.

This report details the architecture, its components, the end-to-end workflow, a practical example, security benefits, and practical considerations for deployment. The goal is to ensure DNSSEC remains secure in a post-quantum world while maintaining compatibility with existing standards.

2 Architecture Overview

The proposed architecture integrates post-quantum cryptography into DNSSEC while preserving its core structure: a chain of trust from the root zone to authoritative zones. Below are the key components.

2.1 Zone Signing Key (ZSK) - Falcon-512

- **Purpose**: Signs all RRsets (e.g., A, MX, TXT records) in a DNS zone.
- **Algorithm**: Falcon-512, a lattice-based signature scheme selected by NIST for post-quantum standardization.
- **Characteristics**:
  - *Security*: Based on NTRU lattices and the Short Integer Solution (SIS) problem, providing quantum-resistant security.
  - *Signature Size*: 666 bytes, suitable for DNS packets.
  - *Performance*: Offers fast signing and verification.
- **Role in DNSSEC**:
  - The ZSK public key (897 bytes) is published in the DNSKEY RRset.
  - Each RRset is signed with Falcon-512, producing an `RRSIG` record.

2.2 Key Signing Key (KSK) - Falcon-512 with Merkle Tree

- **Purpose**: Signs the DNSKEY RRset, establishing the chain of trust.
- **Algorithm**: Falcon-512 signatures, with a Merkle Tree to authenticate multiple KSK public keys.
- **Characteristics**:
  - *Security*: Combines Falcon-512's lattice-based security with the collision resistance of SHA-256 in the Merkle Tree.
  - *Structure*: A binary tree where leaves are SHA-256 hashes of Falcon-512 KSK public keys, and the root (Merkle root) is the KSK public key published in the DNSKEY RRset.
  - *Statelessness*: No need to track signature state, as each KSK public key is used with Falcon-512 signatures.
- **Role in DNSSEC**:
  - The Merkle root is published in the DNSKEY RRset as the KSK public key.
  - The DNSKEY RRset is signed with a Falcon-512 KSK private key, producing an `RRSIG`.
  - The `RRSIG` is accompanied by a Merkle authentication path to verify the KSK public key against the Merkle root.

2.2.1 The Authenticated Path

- The *authenticated path* in the Merkle Tree consists of sibling hashes from the leaf (hash of a KSK public key) to the root.
- During validation, the resolver:
  - Hashes the KSK public key used in the `RRSIG`.
  - Iteratively combines this hash with the sibling hashes in the authenticated path to recompute the Merkle root.
  - Verifies that the recomputed root matches the KSK public key in the DNSKEY RRset.
- If valid, the resolver uses the KSK public key to verify the Falcon-512 signature on the DNSKEY RRset, ensuring the chain of trust.

2.3 DNSKEY Resource Record Set

- **Contents**:
  - ZSK public key (Falcon-512, 897 bytes).
  - KSK public key (Merkle root, 32 bytes).
- **Role**: Published in the zone, signed by the KSK, and used by resolvers to verify RRset signatures.

2.4 Delegation Signer (DS) Record

- **Purpose**: Links the child zone to the parent zone in the chain of trust.
- **Content**: A SHA-256 hash of the KSK public key (Merkle root).
- **Role**: Published in the parent zone, signed by the parent's ZSK.

3 End-to-End Workflow

The workflow is divided into two phases: **Key Generation and Signing** (zone owner) and **Validation** (resolver). We use `example.com` as the example zone.

3.1 Key Generation and Signing (Zone Owner)

3.1.1 Step 1: Generate ZSK (Falcon-512)

- **Process**:
  - Generate a Falcon-512 key pair:
    - Private key: Short polynomials in an NTRU lattice.
    - Public key: 897 bytes, derived from the private key.
  - Publish the public key in the DNSKEY RRset: `example.com DNSKEY 256 3 16 <base64-encoded public key>`.

3.1.2 Step 2: Generate KSK (Falcon-512 with Merkle Tree)

- **Process**:
  - Generate multiple Falcon-512 KSK key pairs (e.g., 8 keys).
  - Compute leaf nodes by hashing each KSK public key with SHA-256.
  - Build the Merkle Tree:
    - Pairwise hash the leaf nodes (SHA-256) to form the next layer.
    - Continue until reaching the Merkle root (32 bytes).
  - Publish the Merkle root in the DNSKEY RRset: `example.com DNSKEY 257 3 16 <base64-encoded Merkle root>`.
  - Sign the DNSKEY RRset with a Falcon-512 KSK private key, producing an `RRSIG`.
  - Include the corresponding KSK public key and Merkle authentication path (sibling hashes) with the `RRSIG`.

3.1.3 Step 3: Sign RRsets with ZSK

- **Process**:
  - For each RRset (e.g., `example.com A 192.0.2.1`):

3.1.4 Step 4: Register DS Record

- **Process**:
  - Compute the DS record by hashing the Merkle root (SHA-256).
  - Submit to the parent zone (e.g., `.com`), which signs it with its ZSK.

3.2 Validation (Resolver)

3.2.1 Step 1: Validate DS Record

* Query the parent zone (`.com`) for the DS record of `example.com`.
* Verify the parent's `RRSIG` on the DS record using the parent's ZSK.

3.2.2 Step 2: Validate DNSKEY RRset

* Query `example.com` for its DNSKEY RRset.
* Verify the `RRSIG` on the DNSKEY RRset:
  - Hash the provided KSK public key (SHA-256).
  - Use the Merkle authentication path to recompute the Merkle root.
  - Compare with the KSK public key (Merkle root) in the DNSKEY RRset.
  - If valid, use the KSK public key to verify the Falcon-512 signature on the DNSKEY RRset.
* Verify the DS record by hashing the Merkle root and comparing with the DS hash.

3.2.3 Step 3: Validate RRsets

* Query for the RRset (e.g., `example.com A`).
* Retrieve the `RRSIG`.
* Use the ZSK public key (Falcon-512) to verify the signature:
  - Recompute the SHA-256 hash of the canonicalized RRset and metadata.
  - Verify the Falcon-512 signature using the ZSK public key.
* If valid, trust the RRset.

3.2.4 Step 4: Chain of Trust

* Recursively validate up the hierarchy (`.com` to root) until a trusted anchor (root KSK) is reached.

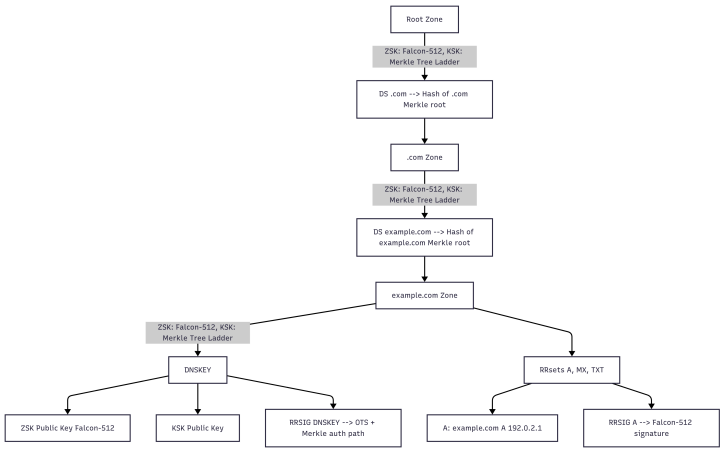

Figure 1: Post-Quantum DNSSEC workflow with Falcon-512 and Merkle Tree.

4 Security Analysis

* **Quantum Resistance**:
  - *Falcon-512*: Secure against quantum attacks (Shor's algorithm ineffective against lattice problems).
  - *Merkle Tree*: Relies on SHA-256, providing quantum resistance via collision resistance.
* **Classical Security**:
  - Falcon-512 offers 128-bit security against classical adversaries.
  - Merkle Tree ensures KSK public key authenticity with SHA-256's 256-bit security.
* **Attacks Mitigated**:
  - *Forgery*: Prevented by Falcon-512 signatures and Merkle Tree authentication.
  - *Key Compromise*: Mitigated by frequent ZSK rotation and secure KSK management.
* **Merkle Tree Integrity**:
  - Ensures authenticity of multiple KSK public keys without requiring state management.

5 Advantages

* **Quantum Resistance**: Falcon-512 and SHA-256-based Merkle Tree are NIST-aligned for post-quantum security.
* **Performance**:
  - Falcon-512: Compact signatures (666 bytes) and fast operations.
  - Merkle Tree: Efficient verification with small authentication paths (e.g., 3 hashes for 8 keys).
* **Compatibility**: Uses existing DNSSEC records (DNSKEY, DS, RRSIG).
* **Separation of Concerns**:
  - ZSK (Falcon-512): Optimized for frequent RRset signing.
  - KSK (Falcon-512 + Merkle Tree): Authenticates KSK public keys for long-term security.
* **Statelessness**: No need to track signature state, unlike one-time signature schemes.

6 Practical Considerations

* **Signature Sizes**:
  - Falcon-512: 666-byte signatures fit within DNS packets (with EDNS0).
  - Merkle Tree: Authentication paths (e.g., 96 bytes for 3 levels) are compact but require EDNS0 or DNS-over-TLS/DoH for larger RRsets.
* **Key Rotation**:
  - ZSK: Rotate monthly using standard DNSSEC rollover.
  - KSK: Rotate when KSK key pairs are depleted or compromised, updating the Merkle root and DS record.
* **Implementation**:

- Use `liboqs` for Falcon-512 and OpenSSL for SHA-256.
- Update DNS software (e.g., BIND, Unbound) to support algorithm ID 16 (Falcon-512) and Merkle Tree paths.

* **Transition Strategy**:
  - Deploy hybrid DNSKEY RRsets with ECDSA and Falcon-512.
  - Update trust anchors gradually as resolvers support new algorithms.
* **Storage**:
  - Store KSK private keys securely (e.g., HSM).
  - Cache Merkle Tree data for efficient signing and verification.

7 Summary Table

Component	Algorithm	Purpose	Signing Data	Validation Process
ZSK	Falcon-512	Signs RRsets	A, MX, TXT, etc.	Verify Falcon-512 sig using ZSK public key
KSK	Falcon-512 + Merkle Tree	Signs DNSKEY RRset	DNSKEY RRset	Verify Falcon-512 sig + Merkle path to root
DS	SHA-256	Authenticates KSK	Hash of Merkle root	Compare hash with KSK public key

Table 1: Summary of Post-Quantum DNSSEC Components

8 Conclusion

The proposed Post-Quantum DNSSEC Architecture ensures DNSSEC's security in a quantum computing era. By using Falcon-512 for both ZSK and KSK, and a Merkle Tree to authenticate KSK public keys, it balances performance, security, and compatibility. The workflow for `example.com` demonstrates its practicality, while the security analysis confirms its robustness against quantum and classical threats. Deployment requires DNS software updates and careful KSK management, but the architecture provides a scalable foundation for securing DNS in a post-quantum world.