FHE in edge computing

Fully Homomorphic Encryption (FHE) in Edge Computing

How FHE Works in Edge Computing

Core Concept: FHE allows computations to be performed directly on encrypted data without ever decrypting it. In edge computing contexts, this means:

1. **Data remains encrypted** on IoT devices and edge nodes
2. **Processing happens on ciphertext** - mathematical operations produce encrypted results
3. **Only the data owner** can decrypt the final results with their private key

Edge Computing Flow:

```
IoT Device → Encrypt Data → Send to Edge Server →
Compute on Encrypted Data → Return Encrypted Result →
Decrypt Locally
```

Key Use Cases

1. Healthcare IoT

- Medical sensors encrypt patient data locally
- Edge servers analyze encrypted health metrics
- Privacy preserved while enabling real-time diagnostics

2. Smart Cities

- Surveillance cameras encrypt video feeds
- Edge nodes process encrypted footage for threat detection
- Privacy-compliant monitoring

3. Industrial IoT

- Factory sensors encrypt operational data
- Edge analytics on encrypted production metrics
- Protects trade secrets while enabling optimization

4. Financial Services

- Mobile payment data stays encrypted during processing
- Fraud detection on encrypted transactions at edge
- Regulatory compliance maintained

5. Smart Home Devices

- Voice assistants process encrypted commands
- Privacy-preserving automation and analytics

Major Challenges

Computational Overhead

- FHE operations are **10,000-1,000,000x slower** than plaintext operations
- Ciphertext sizes are much larger (10-100x expansion)
- High memory requirements

Limited Operations

- Complex operations (like division, comparisons) are expensive
- Deep neural networks are challenging to implement

Power Constraints

- IoT devices have limited battery and processing power
- FHE computations drain resources quickly

Latency Issues

- Real-time applications struggle with FHE overhead
- Not suitable for ultra-low-latency requirements (<100ms)

Key Management

- Secure key distribution across edge networks
- Bootstrap key handling complexity

Mobile OS & IoT Support

Mobile Operating Systems

Android:

- No native FHE support
- Requires external libraries (Microsoft SEAL, HElib)
- Performance limited by mobile CPU constraints

iOS:

- Similar to Android - library-based implementation
- Apple's focus on on-device processing reduces FHE need
- Some Secure Enclave integration possibilities

IoT Computation Support

Current State:

- Most IoT devices **too resource-constrained** for full FHE
- Hybrid approaches more common:
  - Lightweight encryption on device
  - FHE processing at edge gateway
  - Result sent back to device

Hardware Acceleration:

- FPGA-based FHE accelerators emerging
- GPU acceleration helps but not widely deployed in IoT
- Specialized FHE chips in research phase

Compatible Platforms:

- Raspberry Pi 4/5 (minimal FHE possible)
- NVIDIA Jetson series (better FHE performance)
- Intel NUC and similar edge computers
- Traditional IoT sensors (ESP32, Arduino) - **not suitable** for FHE

Implementation Libraries & Tools

1. Microsoft SEAL

- **Language:** C++, with Python/C# wrappers
- **Best for:** Research and prototyping
- **Platform:** Cross-platform (Windows, Linux, macOS)
- **Link:** https://github.com/microsoft/SEAL

2. HElib (IBM)

- **Language:** C++
- **Best for:** Advanced FHE schemes
- **Platform:** Linux, macOS
- **Link:** https://github.com/homenc/HElib

3. TFHE (Fast FHE)

- **Language:** C++
- **Best for:** Boolean circuits, low-latency
- **Platform:** Cross-platform

- **Link:** https://github.com/tfhe/tfhe

4. Concrete (Zama)

- **Language:** Python, Rust
- **Best for:** Machine learning on encrypted data
- **Platform:** Cross-platform
- **Link:** https://github.com/zama-ai/concrete

5. PALISADE

- **Language:** C++
- **Best for:** Lattice-based cryptography
- **Platform:** Cross-platform
- **Link:** https://gitlab.com/palisade/palisade-release

Benchmark Applications & Testing

Ready-to-Test Applications

1. Encrypted Image Classification

- **Tool:** Concrete-ML by Zama
- **Test:** Run neural networks on encrypted images
- **Benchmark:** Accuracy vs. inference time

2. Private Set Intersection

- **Tool:** Microsoft SEAL examples
- **Test:** Find common elements between encrypted datasets
- **Benchmark:** Set size vs. computation time

3. Encrypted Database Queries

- **Tool:** CryptDB-style implementations

- **Test:** SQL queries on encrypted data
- **Benchmark:** Query complexity vs. response time

Benchmarking Frameworks

SHEEP (Secure Homomorphic Encryption Evaluation Platform)

- Compares different FHE libraries
- Standardized test circuits
- Performance metrics across schemes

HE-Benchmark Suite

- Tests common operations (addition, multiplication)
- Measures throughput and latency
- Available for SEAL, HElib, TFHE

Sample Benchmark Metrics to Track

1. **Encryption time** (ms/KB)
2. **Computation time** (operations/second)
3. **Decryption time** (ms)
4. **Memory usage** (MB for ciphertext)
5. **Accuracy degradation** (for ML applications)
6. **Power consumption** (watts, critical for IoT)

Getting Started with Testing

Quick Trial Setup (Using Microsoft SEAL)

```
# Install dependencies
git clone https://github.com/microsoft/SEAL.git
cd SEAL
cmake -S . -B build
cmake --build build
```

```
# Run examples
cd build/bin
./sealexamples
```

Python Quick Start (Using Concrete)

```
from concrete import fhe
import numpy as np

# Simple encrypted addition
@fhe.compiler({"x": "encrypted", "y": "encrypted"})
def add_encrypted(x, y):
    return x + y

# Compile and test
inputset = [(np.random.randint(0, 100),
             np.random.randint(0, 100))
            for _ in range(100)]
circuit = add_encrypted.compile(inputset)

# Benchmark
result = circuit.encrypt_run_decrypt(5, 10)
```

Practical Recommendations

For Production Edge Computing:

1. Use **hybrid encryption** - FHE only for sensitive operations
2. Implement **computation offloading** to more powerful edge nodes
3. Consider **homomorphic encryption alternatives** (like secure enclaves) for lower overhead
4. Start with **simple operations** before complex ML models

For IoT Scenarios:

1. Gateway-based FHE (not on-device)

2. Batching operations to amortize overhead
3. Hardware acceleration where possible
4. Regular vs. FHE mode switching based on sensitivity

FHE in edge computing is still emerging technology - powerful for privacy but with significant performance tradeoffs. The field is rapidly evolving with new optimizations and hardware support continuously being developed.