Erklärungen zur Präsentation am 26.11.2024

Die 4 Fragwürdigen

November 2024

1 Einleitung

Überblick über das Ziel der Analyse. Wir haben uns die Masterarbeit angesehen blablabla, um die Schwächen und Stärken zu herauszufinden. Dabei stand folgende Frage im Mittelpunkt:

Die wichtigsten Analysebereiche:

- Codequalität und Struktur,
- Vollständigkeit und Reproduzierbarkeit der Dokumentation,
- Identifikation von Optimierungsmöglichkeiten.

—

2 Dokumentationsanalyse

Stärken

Die Dokumentation enthält ausführliche Beschreibungen der Algorithmen Kyber und SPHINCS+, inklusive:

- Testergebnisse (z. B. Signaturzeiten),
- Vergleich zwischen den Algorithmen.

Schwächen

1. Fehlende Details zur Architektur: Die Architektur von Client und Server wird zwar beschrieben, aber es fehlen wichtige Details. Zum Beispiel:

- Wie interagieren Module wie Verschlüsselung, Fehlerbehandlung und Protokollierung?
- Welche Datenformate oder Protokolle werden verwendet?
- Kann man die Architektur erweitern, damit man mehrere Clients unterstützzen kann?

2. Unzureichende Bibliotheksbeschreibung meiner Meinung nach: Tools wie OpenSSL werden genutzt, aber nicht richtig erklärt:

- Warum wurden diese gewählt?
- Welche Funktionen aus OpenSSL und libcurl kommen zum Einsatz?
- Auf Seite 48 der Masterarbeit stehen im Kapitel "Guidelines For Compiling and Running"paar Wörter zu den Bibliotheken. Aber lass das auch kritisieren, damit wir mehr Stoff haben zu erzählen

3. Fehlende Anleitungen: Es gibt keine Schritte, um die Tests zu reproduzieren.

3 Code-Review

Stärken

- Der Code ist modular und ermöglicht einfache Erweiterungen.
- NIST-empfohlene Algorithmen (Kyber, SPHINCS+) sind implementiert.

Schwächen

1. Fehlerbehandlung: Netzwerkfehler oder Verbindungsabbrüche werden nicht ausreichend behandelt. Dies führt zu Programmabbrüchen bei Problemen.

2. Verzögerungen: Die Verwendung von `usleep` für feste Wartezeiten ist ineffizient und könnte durch ereignisbasierte Mechanismen ersetzt werden.

3. Skalierbarkeit: Der Code ist nur für einen Client ausgelegt. Es fehlt eine Lösung für parallele Verbindungen.

—

4 Optimierungspotential

1. Verbesserte Fehlerbehandlung: Einführung einer Wiederholungsmechanik für Verbindungsprobleme.

2. Erweiterte Protokollierung: Speicherverbrauch, Latenzen und weitere Performance-Metriken loggen.

3. Architekturdiagramm: Darstellung der Client-Server-Kommunikation.

—

5 Verbesserungen mit Code-Snippets

Fehlerbehandlung und Wiederholungsmechanik

Problem: Keine Wiederholungsversuche bei Verbindungsproblemen.
Aktuell:

```
if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    fprintf(log_file, "Connection Failed (iteration %d)\n", i+1);
    close(sock);
    usleep(RETRY_DELAY);
    continue;
}
```

Verbessert:

```
for (int retries = 0; retries < MAX_RETRIES; retries++) {
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == 0) {
        fprintf(log_file, "Connection Successful "
            "(iteration %d, retry %d)\n", i+1, retries);
        break;
    }
    fprintf(log_file, "Retrying connection "
        "(iteration %d, attempt %d)\n", i+1, retries);
    usleep(RETRY_DELAY);
    if (retries == MAX_RETRIES - 1) {
        fprintf(log_file, "Connection Failed after "
            "%d attempts (iteration %d)\n", MAX_RETRIES, i+1);
        close(sock);
        return 1;
    }
}
```

—

6 Fazit

Dat janze bietet eine solide Grundlage. Es gibt aber Schwächen in der Fehlerbehandlung, Skalierbarkeit und Dokumentation. Nächste Schritte:

- Umsetzung unserer Optimierungen
- Testing
- Automatisierung der Experimente