TapDano Testing Report

1. Introduction

This **TapDano Testing Report** presents the methodologies, findings, and lessons learned from testing the TapDano project. TapDano consists of four open-source repositories, each with unique testing requirements:

1. **Firmware (JavaCard Applet)**
2. **SDK (used for communication with the Firmware)**
3. **Mobile/Web App (which uses the SDK)**
4. **Aiken Smart Contract (validates signatures generated by the Firmware)**

Although all four repositories are important, this report focuses mainly on the **JavaCard Firmware**, which stores private keys and cannot be easily updated once deployed to physical JavaCards. Additionally, the **Aiken Smart Contract** undergoes critical testing due to its security implications.

---

2. Project Repositories and Scope

2.1 Firmware (JavaCard Applet)

- **Importance**: Stores user private keys on a physical JavaCard.
- **Challenge**: Firmware updates are not straightforward without losing stored data.

2.2 SDK

- **Function**: Provides communication layer between the JavaCard firmware and other components (Mobile/Web App).
- **Testing Points**: Integration tests to ensure stable communication, API compliance, and error handling.

2.3 Mobile/Web App

- **Function**: Interfaces with the SDK to interact with the Firmware.
- **Testing Points**: Usability, user authentication flows, transaction handling, and UI/UX consistency.

2.4 Aiken Smart Contract

- **Function**: Validates signatures generated by the Firmware using `verify_ecdsa_secp256k1_signature`.
- **Testing Points**: Security audits, functional correctness, and performance under various load conditions.

---

3. Firmware Testing

3.1 Overview

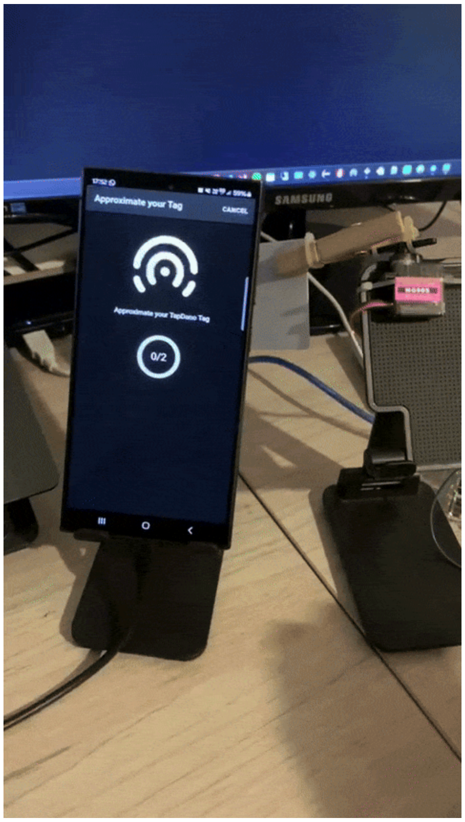
The **JavaCard Firmware** faced the most stringent testing requirements due to **memory-related bugs** identified early in development. These issues risked rendering the JavaCard unusable and causing permanent loss of private keys.

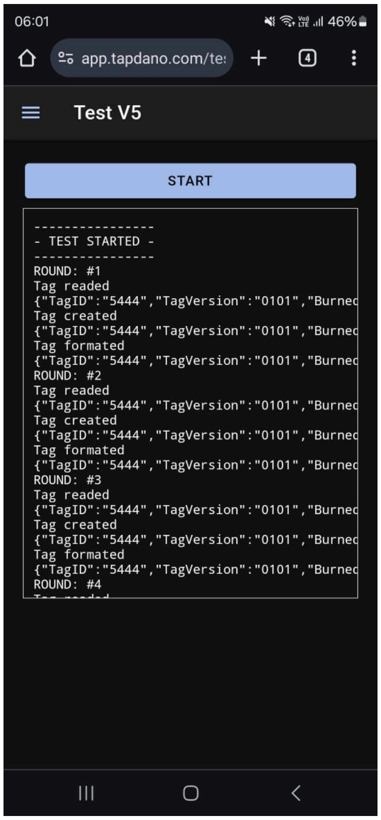
3.2 Memory Bug and Early Challenges

- **Description**: Severe memory allocation defects surfaced in physical testing, causing the chip to become unresponsive.
- **Diagnosis Complexity**:
  - The bug did **not** manifest in emulated environments or with standard USB card readers.
  - It appeared sporadically, often after dozens of successful tests on physical devices.
  - The unpredictable nature of the bug necessitated **advanced testing strategies**.

3.3 Testing Phases for Firmware

1. **Initial Unit and Integration Tests**
   - Conducted in emulated environments to validate basic functionality.
   - Covered cryptographic operations, data storage, and standard error handling.
2. **Physical Device Testing**
   - Revealed memory issues not seen in emulators.
   - Employed multiple JavaCard chips to isolate potential hardware variances.
3. **Automated Hardware Testing** (Milestone 4)
   - Introduced a **servo motor controlled via Arduino** for continuous insertion/removal cycles.
   - Test automation code and evidence: servo-controller-serial repository
   - **Objective**: Stress-test card insertion processes under various angles and frequencies to identify inconsistent memory usage.

- 



```
----------------
- TEST STARTED -
----------------
ROUND: #1
Tag readed
{"TagID":"5444","TagVersion":"0101","Burned
Tag created
{"TagID":"5444","TagVersion":"0101","Burned
Tag formated
{"TagID":"5444","TagVersion":"0101","Burned
ROUND: #2
Tag readed
{"TagID":"5444","TagVersion":"0101","Burned
Tag created
{"TagID":"5444","TagVersion":"0101","Burned
Tag formated
{"TagID":"5444","TagVersion":"0101","Burned
ROUND: #3
Tag readed
{"TagID":"5444","TagVersion":"0101","Burned
Tag created
{"TagID":"5444","TagVersion":"0101","Burned
Tag formated
{"TagID":"5444","TagVersion":"0101","Burned
ROUND: #4
```

4. **Extended WebNFC Testing**

    - Implemented post-Milestone 4 for greater convenience and repeatability.
    - Eliminated the need for physically moving the card closer or farther from the reader.
    - Enabled **thousands of interactions** on a single chip, confirming the stability of memory optimizations.

3.4 Resolution and Current Status

- **Memory Optimization**: Refined the JavaCard Applet code to handle memory allocations more efficiently.
- **Successful High-Volume Testing**: Hundreds (and later thousands) of consecutive tests performed without encountering prior memory failures.

3.5 Additional Testing Considerations

- **Security Evaluations**: Assess cryptographic key generation, storage, and deletion flows.
- **Regression Testing**: Regularly re-run critical test suites whenever firmware updates or minor patches are implemented.
- **Load and Stress Testing**: Continue high-frequency test loops to detect any regression in performance or stability.

4. Aiken Smart Contract Testing

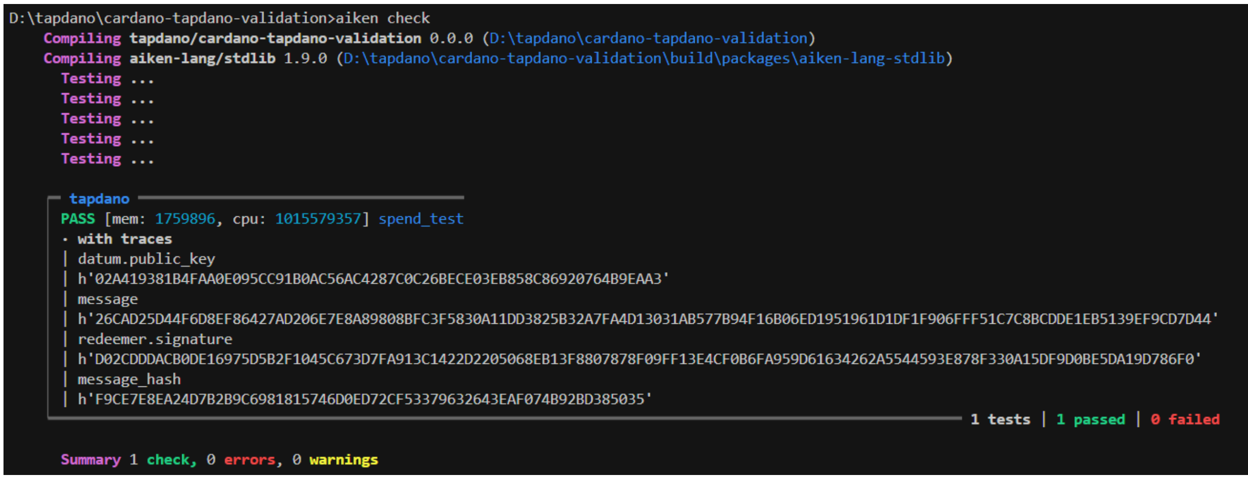
4.1 Importance

The **Aiken Smart Contract** is the second most critical component because it verifies the digital signatures generated by the Firmware. Reliability and security in the smart contract logic are paramount to prevent unauthorized transactions.

4.2 Simplification for Reliability

- **Purpose**: Limit the scope of the contract to essential signature validation.
- **Rationale**: Reducing code complexity minimizes potential attack vectors and bugs.

4.3 Testing Methodology

1. **Aiken Standard Testing**
    - Utilized the Aiken CLI (`aiken check`) to verify contract syntax, logic, and security.
    - Ensured compliance with Aiken best practices and patterns.
    - 

```
D:\tapdano\cardano-tapdano-validation>aiken check
    Compiling tapdano/cardano-tapdano-validation 0.0.0 (D:\tapdano\cardano-tapdano-validation)
    Compiling aiken-lang/stdlib 1.9.0 (D:\tapdano\cardano-tapdano-validation\build\packages\aiken-lang-stdlib)
      Testing ...
      Testing ...
      Testing ...
      Testing ...
      Testing ...

    ┍━ tapdano ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
    │ PASS [mem: 1759896, cpu: 1015579357] spend_test
    │ · with traces
    │ | datum.public_key
    │ | h'02A419381B4FAA0E095CC91B0AC56AC4287C0C26BECE03EB858C86920764B9EAA3'
    │ | message
    │ | h'26CAD25D44F6D8EF86427AD206E7E8A89808BFC3F5830A11DD3825B32A7FA4D13031AB577B94F16B06ED1951961D1DF1F906FFF51C7C8BCDDE1EB5139EF9CD7D44'
    │ | redeemer.signature
    │ | h'D02CDDDACB0DE16975D5B2F1045C673D7FA913C1422D2205068EB13F8807878F09FF13E4CF0B6FA959D61634262A5544593E878F330A15DF9D0BE5DA19D786F0'
    │ | message_hash
    │ | h'F9CE7E8EA24D7B2B9C6981815746D0ED72CF53379632643EAF074B92BD385035'
    ┕━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1 tests | 1 passed | 0 failed

      Summary 1 check, 0 errors, 0 warnings
```

2. **Functional Verification**
    - Validated `verify_ecdsa_secp256k1_signature` under varied scenarios (e.g., valid signatures, invalid signatures, edge cases).
3. **Extended Security Audits**
    - Potential introduction of static analysis tools for detecting vulnerabilities.
    - Peer reviews to confirm contract integrity.

5. Milestones and Evidence

5.1 Milestone 4

- **Memory Issue Discovery**: Prompted immediate shift from planned future testing (Milestone 6) to expedited testing.
- **Outcome**:
    - Deployed **automated hardware testing** to replicate field conditions.
    - Addressed and resolved the memory allocation issue.

5.2 Post-Milestone 4 Testing Expansion

- **WebNFC Testing**: Significantly scaled up test volumes without manual card repositioning.
- **Reliability Confirmed**: Demonstrated stable performance across thousands of read/write cycles.

6. Conclusion

The **TapDano** testing strategy evolved rapidly due to unexpected memory-related issues in the JavaCard Firmware. By advancing hardware testing and later employing **WebNFC** for high-volume testing, the team successfully resolved critical bugs and established a robust testing framework. Simultaneously, **Aiken Smart Contract** testing ensured the integrity of signature validation logic, further strengthening the project's security posture.

Going forward, **continued regression testing, security assessments, and stress testing** will help maintain TapDano's reliability as the project scales. By systematically addressing these challenges and adapting to new findings, the TapDano project is well-positioned for secure and stable operation.