National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cybersecurity

Varun Dilip Bhiwapurkar
Student ID: 23412216

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Varun Dilip Bhiwapurkar
Student ID:	23412216
Programme:	MSc in Cybersecurity
Year:	2025
Module:	MSc Research Project
Supervisor:	Michael Prior
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	674 Words
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Varun Dilip Bhiwapurkar
Date:	11th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Varun Dilip Bhiwapurkar
23412216

1 Introduction

In this Configuration Manual we will see the entire setup, installation, configuration, implementation and execution instructions for the Secure Over-The-Air (OTA) Firmware Update System which is developed as part of the MSc Research Project. The implementation consists of two major components:

(1) Node.js backend server which is responsible for handling the firmware uploads, generating signature manifests and storing the updated records in MongoDB and delivers OTA updates to vehicle client over HTTPS.

(2) React-based frontend client that acts as a vehicle ECU which fetches update metadata, verifies digital signatures, validates firmware integrity, and applies the update.

The manual contains the software requirements, configuration and installation instructions to perform procedures and test-case demonstration guidelines to simulate an actual working system accurately.

2 System Requirements

2.1 Hardware Requirements

- Operating System: Windows 11
- RAM: 8 GB RAM
- Minimum storage: 5 GB

2.2 Software Requirements

- Node.js v18+ (includes npm v9+)
- MongoDB Community Server v6.x
- OpenSSL (for generating the RSA key-pair)
- Web browser (Chrome)

2.3 Project File Structure

```
ota-project/
     backend/
       manufacturer_private.pem
       manufacturer_public.pem
       server.crt
       server.key
       server.js
       tamper.js
       spoof_signature.js
       update_check.js
       package.json
     frontend/
        ota-client/
            src/App.js
            package.json
            node_modules/
```

- Port 8443 (for HTTPS backend server) and Port 3000(for React server) must be accessible.

3 Backend Configuration

3.1 Backend Package Versions

These are the packages with their versions that were used in the backend

```
"dependencies":
  cors: 2.8.5,
  dotenv: 17.2.3,
  express: ^5.1.0,
  mongoose: 8.2.0,
  multer: 2.0.2

"dev Dependencies":
  nodemon:3.1.11
```

3.2 Installing dependencies

Open a windows powershell terminal in the backend directory:

```
cd backend
npm install
```

3.3 SSL and RSA Key Setup

Ensure these files are present in the backend directory:

- manufacturer_private.pem
- manufacturer_public.pem
- server.crt
- server.key

3.4 Starting MongoDB

```
Start-Service MongoDB
```

3.5 Running the Backend Server

```
cd backend
node server.js
```

The Expected output would look like:

```
[mongo] connected
[ota-server] running https://localhost:8443
```

4 Frontend Configuration

4.1 Frontend Package Versions

```
"dependencies":
  @testing-library/dom: 10.4.1,
  @testing-library/jest-dom: 6.9.1,
  @testing-library/react": 16.3.0,
  @testing-library/user-event": 13.5.0,
  base64-js: 1.5.1,
  react: 19.0.0,
  react-dom: 19.0.0,
  react-scripts: 5.0.1,
  web-vitals: 2.1.4
```

4.2 Installing Frontend Dependencies

```
cd frontend/ota-client
npm install
npm start
```

The React client will automatically open the browser page at:

```
http://localhost:3000
```

5 Database Configuration

The system stores:

- Firmware binary
- Manifest metadata
- RSA digital signature
- Exact signed manifest string (manifest_json)

To look at the updates:

```
mongo
use ota_demo
db.updates.find().pretty()
```

6 Demonstrating the OTA Update Simulation Workflow

6.1 Step 1: Upload Firmware (Manufacturer)

Creating firmware:

```
"Demo firmware v4" | Out-File -Encoding ascii firmware_v4.bin
```

Uploading Firmware:

```
curl.exe -k -F "firmware=@firmware_v4.bin" -F "version=1.0"
https://localhost:8443/api/upload
```

The expected output would include:

- Manifest metadata
- Base64 digital signature
- Update ID

6.2 Step 2: Vehicle Client Request

Clicking on the 'Check for Update' button, the React client:

1. Fetches manifest and signature
2. Imports manufacturer public key
3. Verifies RSA-PSS signature
4. Downloads firmware
5. Computes SHA-256 hash
6. Applies update if integrity matches

7 Test Case Execution

7.1 Normal Update

Expected output:

- Signature verification: **OK**
- Hash integrity check: **Matches**
- Firmware downloaded and applied

7.2 Tampering Test

Here we will modify the firmware using tamper.js:

```
node tamper.js <id>
```

Expected Vehicle Client Output:

```
Signature OK
Hash mismatch!
Update rejected
```

7.3 Spoofing Test

Replace the authentic signature with a fake signature:

```
node spoof_signature.js <id>
```

Expected Client Result:

```
Signature verification FAILED
Update rejected
```

8 Troubleshooting

Content-Length: 0 or Empty Firmware

It is caused because of firmware conversion issue inthe server route,to fix this issue use robust buffer conversion inside the firmware endpoint.

Signature Verification Failed

Below are the most common reasons for this issue:

- Incorrect public key
- Manifest string mismatch
- Wrong RSA-PSS saltLength (correct salt length = 32).

CORS Errors

Ensure that the backend file includes:

```
app.use(cors());
```

9 Conclusion

This Configuration Manual provides full step by step installation, configuration and testing of the workflow to secure autonomous OTA update system. By following these steps any user can deploy the backend and frontend applicationsand perform cryptographic verification also demonstrate robust security protections including tampering detection and signature validation.