🛡️ Secure OTA Bootloader (STM32G0 + ESP32)

Siber Güvenlik ve Gömülü Sistemler yaklaşımıyla geliştirilmiş; **şifreli**, **hata toleranslı** ve **çift slot (A/B)** mimarisine sahip **güvenli uzaktan yazılım güncelleme (OTA)** sistemi.

1️⃣ Proje Tanıtımı (Introduction)

Bu proje, gömülü sistemlerde sahada karşılaşılan en kritik problemlerden biri olan **Uzaktan Güvenli Yazılım Güncelleme (Secure OTA)** ihtiyacına **endüstriyel seviyede** bir çözüm sunar.

Sistem, **DemeduKit (STM32G030)** ana denetleyicisi üzerinde çalışır ve **ESP32**'yi bir **Wi-Fi ağ köprüsü** olarak kullanarak sunucudan **şifreli firmware paketlerini** indirir. İndirilen yazılım; **bütünlük, sürüm ve kimlik doğrulama kontrollerinden** geçirildikten sonra **kesintisiz bank-swap (A/B)** mantığıyla aktif hale getirilir.

Bu yapı sayesinde:

- Güncelleme sırasında **güç kesintisi** olsa bile sistem çalışır halde kalır
- Yetkisiz veya bozulmuş firmware yüklenemez
- Eski ve zafiyetli sürümlere geri dönüş (rollback) engellenir

---

2️⃣ Geleneksel Bootloader ile Karşılaştırma

Özellik	Geleneksel Bootloader	Secure OTA Bootloader (Bu Proje)
Yazma Stratejisi	Aktif uygulamanın üzerine yazar	Dual-Slot (A/B) Bank-Swap
Veri Güvenliği	Plain-Text UART	AES-256 Şifreli Transfer
Bütünlük Kontrolü	Basit CRC	CRC32 + Header doğrulama
Sürüm Kontrolü	Yok	Anti-Rollback
Güç Kesintisi	Brick riski	Fault-Tolerant
Kullanıcı Bildirimi	Yok / LED	OLED durum & hata kodları

3️⃣ Uygulama ve Sistem Mimarisi

3.1 Donanım Bileşenleri

- **STM32G030C8T6** – Ana hedef MCU
- **ESP32** – Wi-Fi OTA köprüsü
- **0.96" I2C OLED (SSD1306)** – Durum ve hata bildirimi

3.2 Flash Hafıza Mimarisi (64 KB)

Bölüm	Başlangıç	Boyut	Açıklama
Bootloader	0x0800_0000	16 KB	Güvenli kök yazılım
Slot A (Active)	0x0800_4000	24 KB	Çalışan uygulama
Slot B (Download)	0x0800_A000	24 KB	Yeni firmware alanı

📌 **Not:** Slot A'nın ilk 64 byte'ı firmware header alanıdır. Uygulama başlangıcı ve VTOR → **0x0800_4040**

3.3 Güvenlik Mekanizmaları

- **AES-256 CBC** – Firmware gizliliği
- **CRC32** – Veri bütünlüğü
- **Magic Number & Header Kontrolü** – Format doğrulama
- **Anti-Rollback** – Eski sürümlerin engellenmesi

3.4 Güncelleme Akışı (Özet)

1. Firmware PC tarafında paketlenir ve şifrelenir
2. ESP32 HTTP üzerinden firmware'i indirir
3. STM32 Slot B'ye yazım yapar
4. CRC + sürüm doğrulaması gerçekleştirilir
5. Başarılıysa Slot A aktive edilir, Slot B geçersizlenir
6. Sistem yeni uygulamaya atlar

4️⃣ Yüksek Güvenlikli Endüstriyel Sistemlerle Karşılaştırma

Güvenlik Özelliği	Bu Proje	Bankacılık / Otomotiv Seviyesi
Kripto Algoritması	AES-256	AES-256 + ECC / RSA
Anahtar Saklama	Yazılım içi	Secure Element / HSM
Key Provisioning	Compile-time	Secure Factory Injection
Fiziksel Güvenlik	Yok	RDP + Tamper Detection
Güvenlik Seviyesi	**Level 2-3**	**Level 4-5**

5️⃣ Geliştirme ve Güvenlik Yol Haritası

- Anahtarların koddan çıkarılması
- Cihaz başına benzersiz anahtar enjeksiyonu
- Provisioning aracı (Python)
- RDP Level 1 / Debug kilitleme

- Secure Element entegrasyonu

☑ Sonuç

Bu proje; **güvenli OTA**, **düşük seviye donanım hâkimiyeti**, **fault-tolerant mimari** ve **endüstriyel bootloader prensiplerini** bir araya getiren, akademik ve profesyonel kullanıma uygun bir **Secure OTA referans tasarımıdır**.

🛡 Secure OTA Bootloader (STM32G0 + ESP32)

A **secure, fault-tolerant, dual-slot (A/B) remote firmware update (OTA) system** developed with a **Mechatronics and Embedded Systems Engineering** approach.

📖 Project Overview

This project provides an **industrial-grade solution** to one of the most critical challenges in embedded systems deployed in the field: **Secure Remote Firmware Updates (Secure OTA)**.

The system runs on a **DemeduKit (STM32G030)** main controller and uses an **ESP32** as a **Wi-Fi network bridge** to download **encrypted firmware packages** from a server. After integrity and version checks, the update is activated using a **seamless bank-swap (A/B) mechanism**.

As a result:

- Power loss during update does **not brick the device**
- Unauthorized or corrupted firmware is rejected
- Rollback to vulnerable firmware versions is prevented

⚖ Traditional Bootloader vs Secure OTA Comparison

Feature	Traditional Bootloader	DEMEDUKIT Secure OTA
Update Strategy	Overwrites active firmware	Dual-Slot (A/B) Bank-Swap
Data Security	Plain-text transfer	AES-256 Encrypted
Integrity	Basic CRC	CRC32 + Secure Header

Feature Check	Traditional Bootloader	DEMEDUKIT Secure OTA
Version Control	None	Anti-Rollback
Power Failure Safety	High brick risk	Fault-Tolerant
Firmware Authentication	Simple magic byte	Signed Secure Header
User Feedback	Minimal	OLED status & error codes

System Architecture & Implementation

Dual-Slot Flash Architecture

- **Slot A – Active Application**
  Currently running, verified firmware.

- **Slot B – Update Slot**
  Temporary storage for encrypted OTA firmware.

If verification fails or power is lost, the bootloader **automatically falls back** to Slot A.

Low-Level Firmware Design

- No HAL usage
- STM32 **Low-Layer (LL)** drivers and direct **register-level access**
- Optimized for:
  - Minimal flash footprint
  - Deterministic behavior
  - Full hardware control

Security Mechanisms

Confidentiality

- **AES-256 CBC encryption**
- Encryption on PC/server side

- Decryption on STM32 during write or activation

Integrity & Authenticity

- **CRC32** for data integrity
- Secure firmware header with magic signature (`SECU`)

Anti-Rollback Protection

- Firmware version comparison
- Older versions are permanently rejected

Memory Map (STM32G030 – 64 KB Flash)

Region	Start Address	Size	Description
Bootloader	0x0800_0000	16 KB	Secure root firmware
Slot A (Active)	0x0800_4000	24 KB	Running application
Slot B (OTA)	0x0800_A000	24 KB	Encrypted update area

📌 **Note:** The first **64 bytes** of Slot A are reserved for the firmware header.
Application entry point & VTOR → **0x0800_4040**

OTA Update Workflow

1. **Firmware Packaging (PC)**
   - `app.bin` compiled
   - Secure header added (Version, Size, CRC, Magic)
   - AES-256 encryption → `secure_app.bin`
2. **Distribution (Server)**
   - Python HTTP server on Kali Linux
3. **Transfer (ESP32 → STM32)**
   - ESP32 downloads firmware
   - Chunked UART transfer to STM32
   - STM32 writes data into Slot B
4. **Verification**
   - Header validation
   - CRC32 integrity check
   - Anti-rollback version check
5. **Activation**
   - Firmware copied (or decrypted) into Slot A
   - Slot B magic invalidated to prevent loops

6. **Execution**
    - VTOR relocated
    - Control transferred to application

🏦 Comparison with High-Security Industrial Systems

Aspect	This Project	Banking / Automotive Grade
Key Storage	Embedded in firmware	HSM / Secure Element
Key Provisioning	Compile-time	Secure factory injection
Physical Protection	None	Mesh + zeroization
Security Level	Level 2	Level 4

🛣️ Security Upgrade Roadmap

- Unique per-device encryption keys
- Secure key injection protocol
- Provisioning tools and key database
- Debug port locking (RDP Level 1)

☑️ Final Outcome

This project demonstrates a **production-ready Secure OTA Bootloader** combining:

- Zero-Trust update philosophy
- Fault-tolerant boot architecture
- Low-level embedded system mastery
- Industrial embedded security principles

making it suitable for **real-world field-deployed embedded devices**.