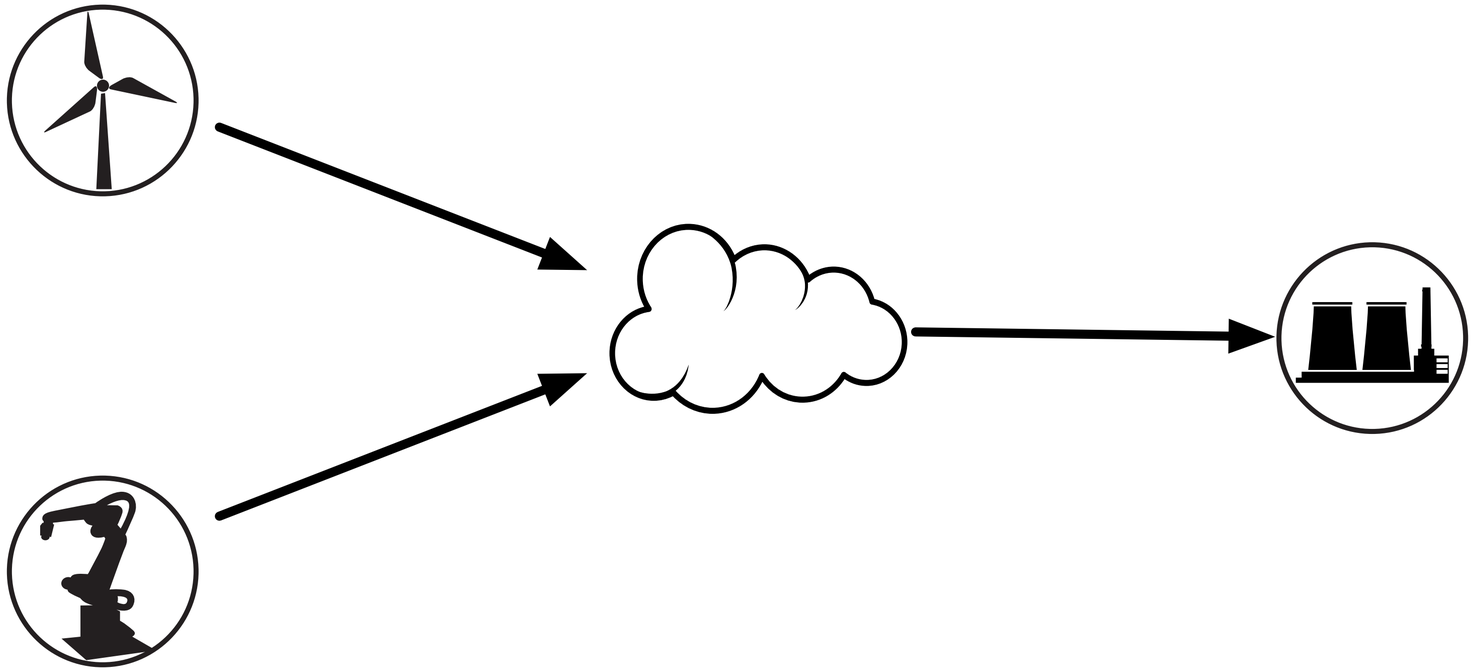
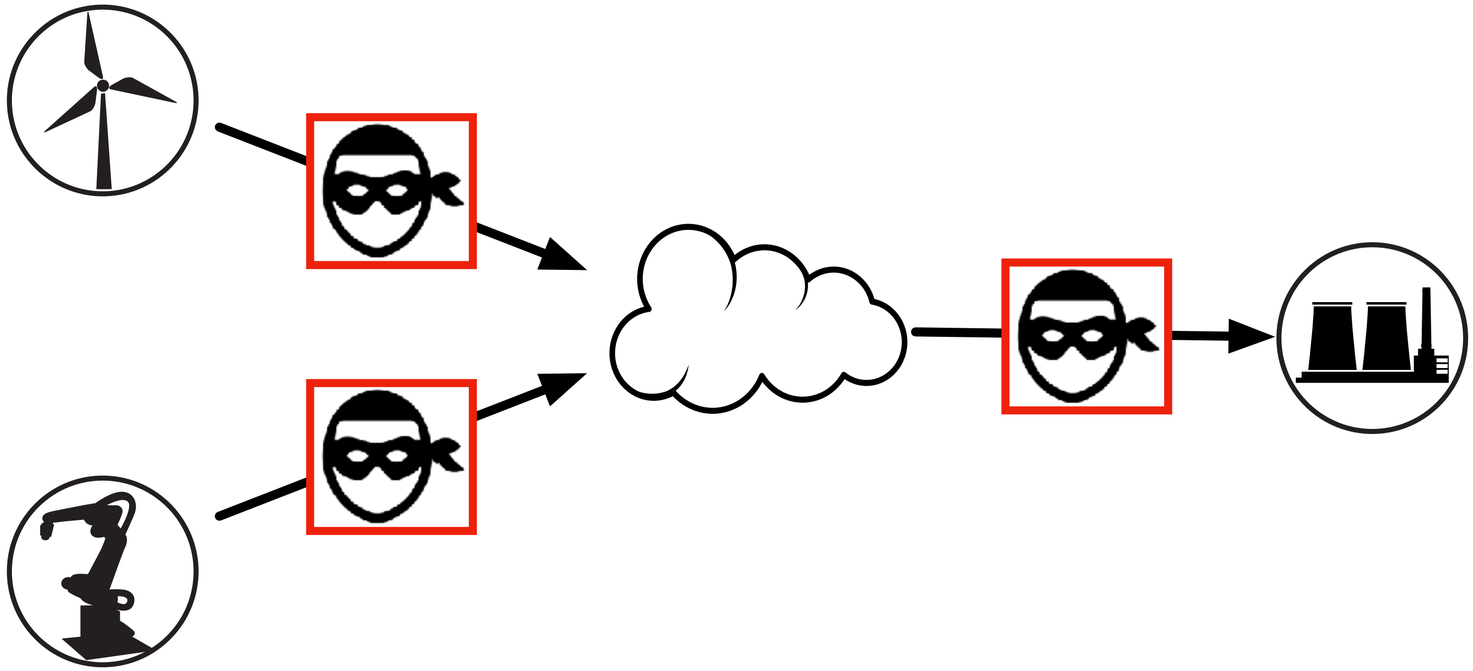
Teserakt

E4: MQTT encryption done right — 20181007



Problem

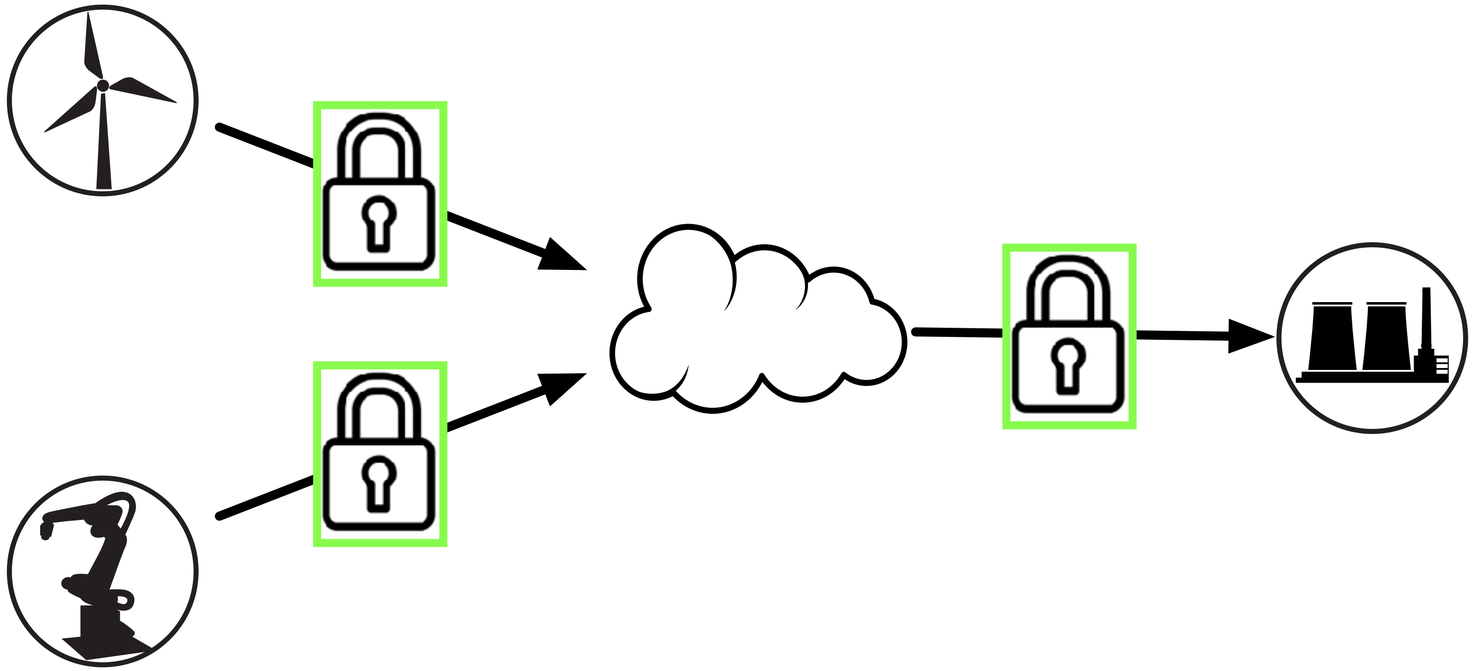
Data sent via third-party servers for reliability and performance



Problem

Data sent via third-party servers for reliability and performance

Attackers can read and corrupt data in transit

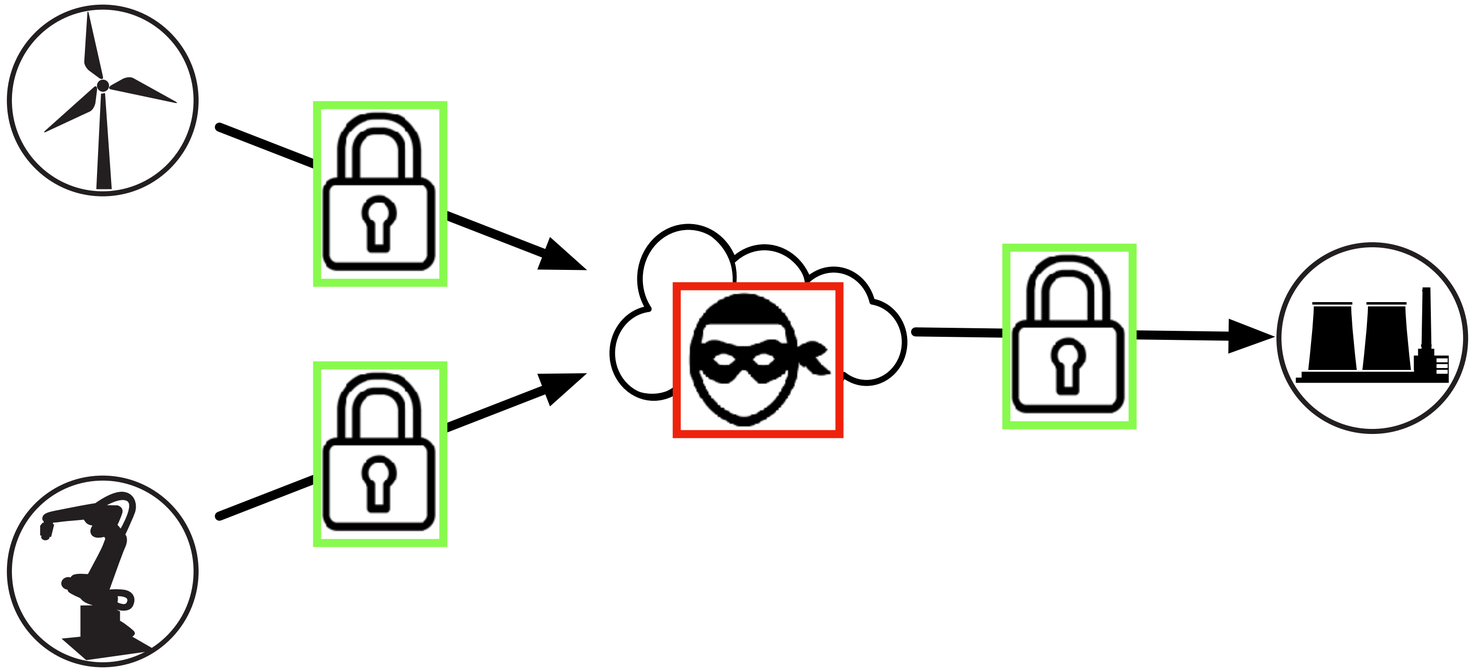


Problem

Data sent via third-party servers for reliability and performance

Attackers can read and corrupt data in transit

Client–server encryption is the default solution (TLS)



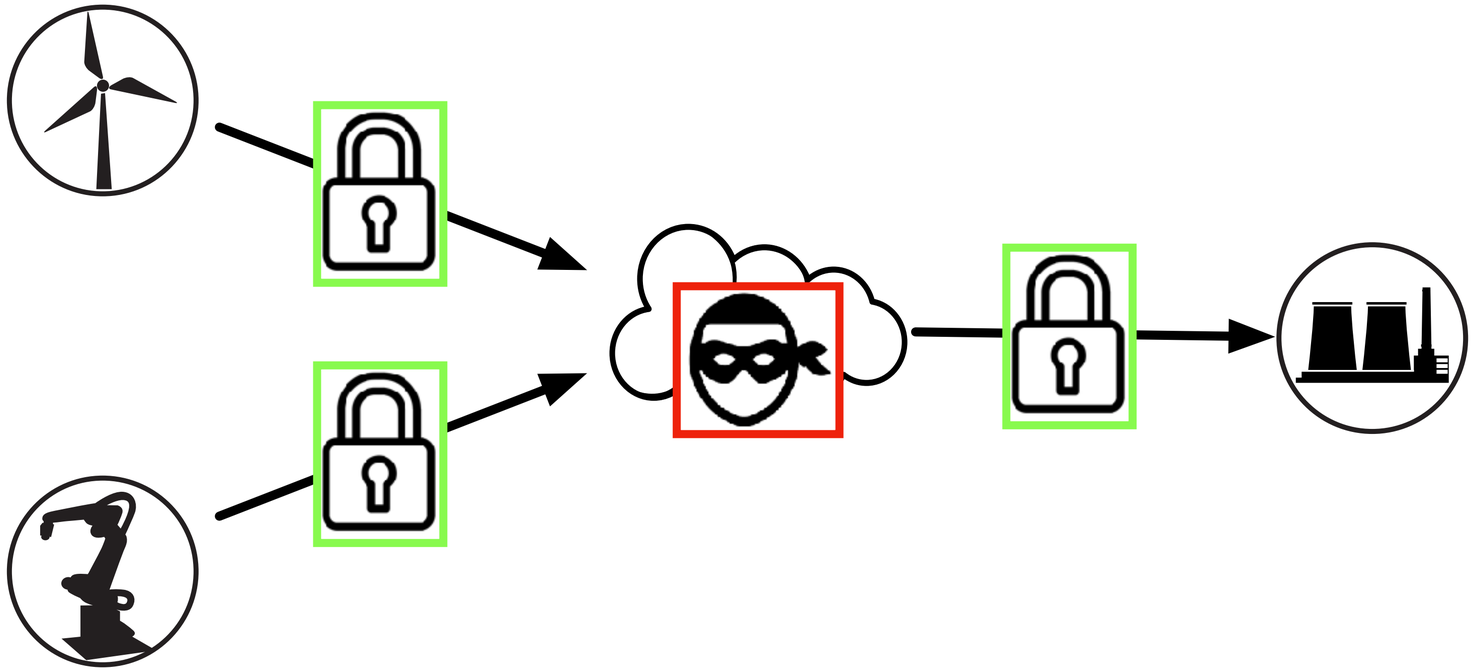
Problem

Data sent via third-party servers for reliability and performance

Attackers can read and corrupt data in transit

Client–server encryption is the default solution (TLS)

But is useless against malicious or compromised servers

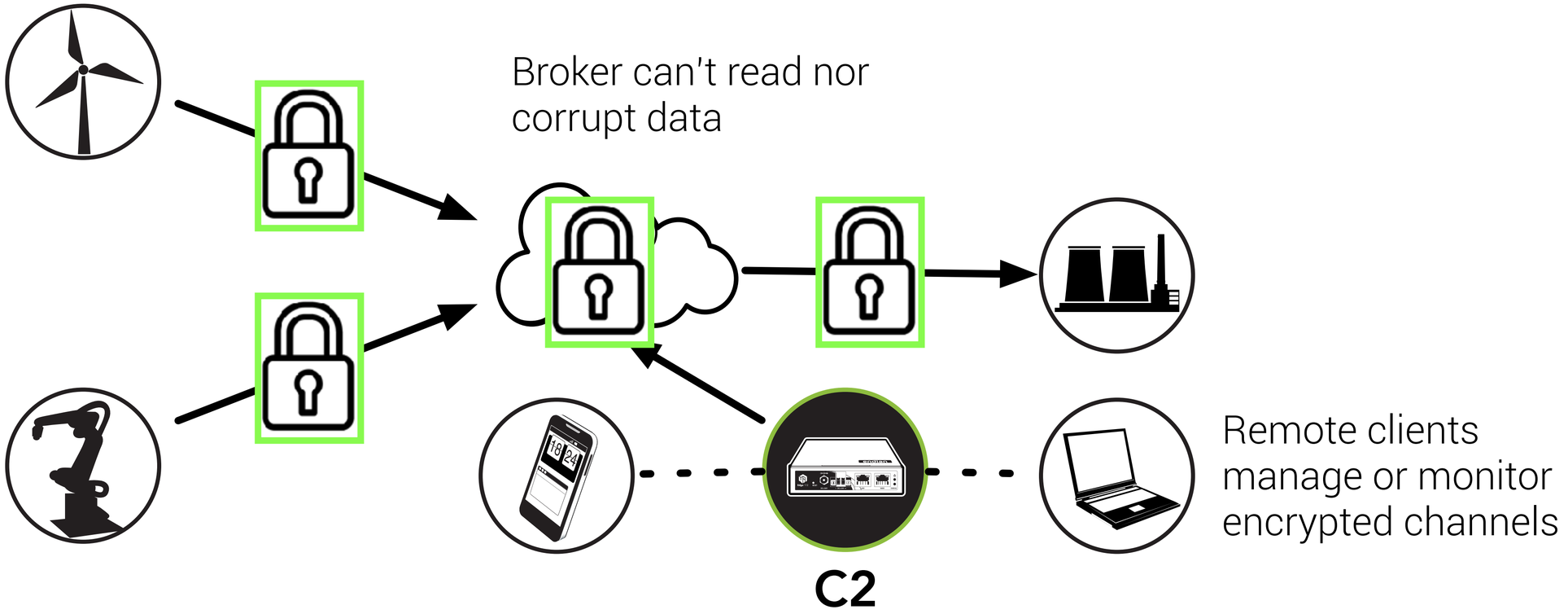


Pain points for users

No commercial solution exists

Current workaround (one global key) is cumbersome and insecure

Standard crypto won't always fit in constrained platforms (like AVR)



Our solution

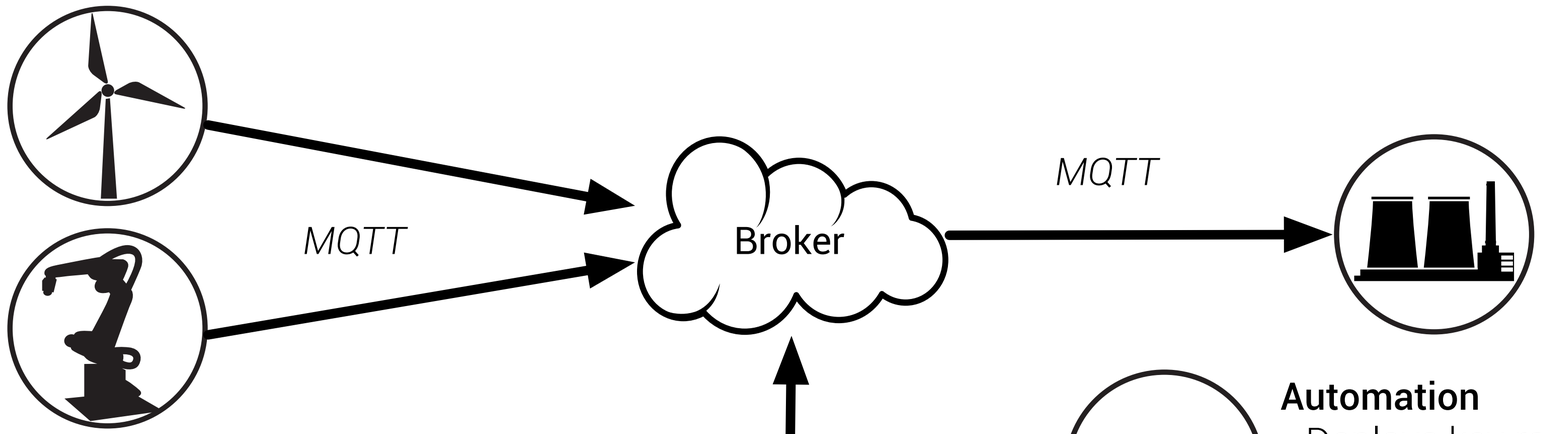
Strong encryption: confidentiality, authenticity, replay protection

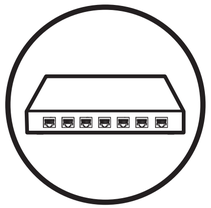
Simple key management: web UI, CLI, automation of key policies

Enables advanced functionalities such as dynamic group messaging

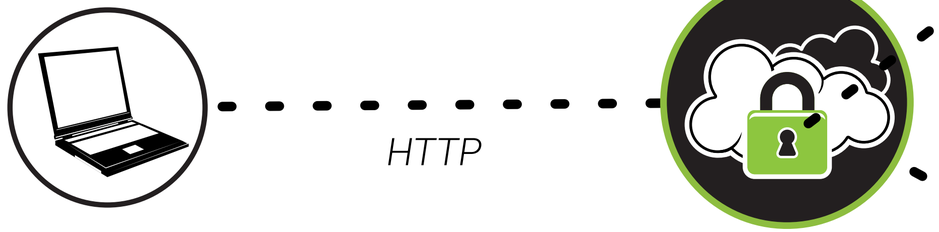
TLS recommended, but E4 does not depend on its security

Typical deployment





Automation
- Deploys key rotation policies
- Using shell scripts or our Go SDK



C2

Engineer workstation
- Command-line utility + interactive shell
- Remotely manage keys and permissions
- Support for batch commands and scripts

Monitoring workstation
- Web UI desktop or mobile platforms
- Dashboard with live data and analytics
- Full key management functionalities

Client software

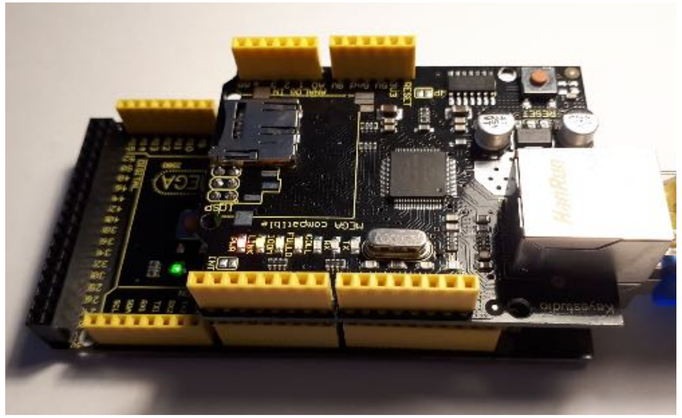
Low-memory, multiple integration options

Encrypts and decrypts MQTT payloads, with only 24 bytes overhead

Processes key management commands received from C2

2 main options, fully interoperable:

- **Go** native library for Linux/macOS/Windows
- **C** lightweight library, running on 8-bit AVR



Also did PoCs in **Java** and **Rust**

C2 server

Production-grade back-end, high reliability

Sends commands to clients, stores clients identities and topic keys

Written in Go, two interfaces:

- **HTTP**/REST: used by web UI and automation scripts
- **gRPC**: used by CLI tool and interactive shell

Security features:

- TLS-protected HTTP and gRPC links
- Database encryption, support for HSM and software vaults

Teserakt

Want to try our solutions or see a demo? Any questions?
Contact us at contact@teserakt.io