

Prototyping an Embedded Off-Switch for AI Compute

James Petrie

1 Introduction

AI accelerators are essential for training frontier systems, but once deployed, preventing unauthorized use is difficult. Chips in facilities with limited oversight can be stolen, diverted, or repurposed. A rogue project with sufficient hardware could undermine safety agreements that others follow.

Software protections fail against adversaries with physical access. Hardware-level enforcement is more robust: circumventing it requires physically modifying chips without destroying them. Hardware off-switches could keep chips governable regardless of physical location, blocking unauthorized use of diverted hardware.

Prior work proposed embedding thousands of “security blocks” for authorization checks into each AI accelerator [4]. This project prototypes an ECDSA-based design for these security blocks in HardCaml and validates it in simulation.

Contributions:

1. Working RTL implementation of an embedded off-switch security block with ECDSA license verification
2. Architecture documentation with explicit security properties and design rationale

2 Related Work

Hardware-enabled governance mechanisms [1, 2] examined on-chip mechanisms and technical solutions for managing national security risks from AI hardware and supporting export control compliance.

Off-switches for AI compute [3, 4] proposed embedding cryptographic authorization checks in accelerators, either via firmware for faster deployment or distributed hardware blocks for stronger security. This project implements the hardware architecture from [4].

3 Methods

3.1 Approach

The security block is implemented in HardCaml, a hardware description language embedded in OCaml. This allows the design to be simulated and validated against test cases before synthesis to physical circuits.

The implementation is available at <https://github.com/JamesPetrie/off-switch> (key files: `security_block.ml`, `ecdsa.ml`, `arith.ml`, `point_add.ml`).

3.2 Architecture

The security block is a deadman’s switch: outputs are blocked by default and enabled only when valid authorization has been recently received. The chip generates a nonce, an external authority signs it, and the chip verifies the signature before granting time-limited permission to operate.

Figure 1 shows the security block architecture. The Int8 adder is a placeholder for actual chip operations such as matrix multiplies or data routing.

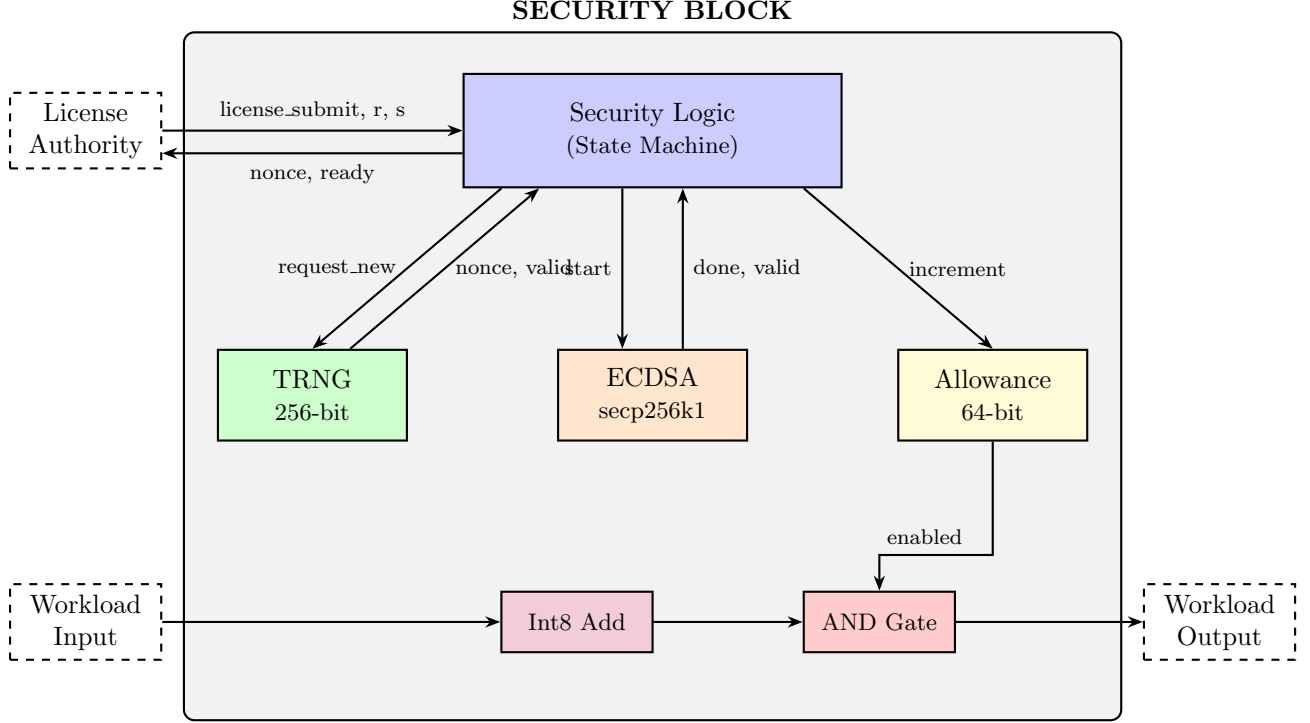


Figure 1: Security block architecture.

Components:

- **TRNG**: 256-bit nonce generation (counter in prototype; ring oscillator in production)
- **ECDSA**: Signature verification on secp256k1
- **Allowance**: 64-bit counter, decrements every cycle, incremented by valid licenses
- **Output gating**: Each output bit ANDed with **enabled** signal

Outputs pass through only when authorized. Each bit is ANDed with **enabled**, which is high only when **allowance** > 0. The rest of the design ensures **allowance** increments only on presentation of a valid signature over the current nonce.

The state machine orchestrates the authorization flow:

1. TRNG generates a nonce
2. Security block publishes nonce, asserts **nonce_ready**
3. External authority signs nonce, submits (r, s)
4. ECDSA verifies signature against hardcoded public key
5. If valid: allowance incremented, new nonce generated
6. If invalid: allowance unchanged, same nonce retained (retry permitted)

The ECDSA module (Figure 2) verifies signatures by computing $R = u_1 \cdot G + u_2 \cdot Q$ and checking

whether $R_x \equiv r \pmod{n}$.

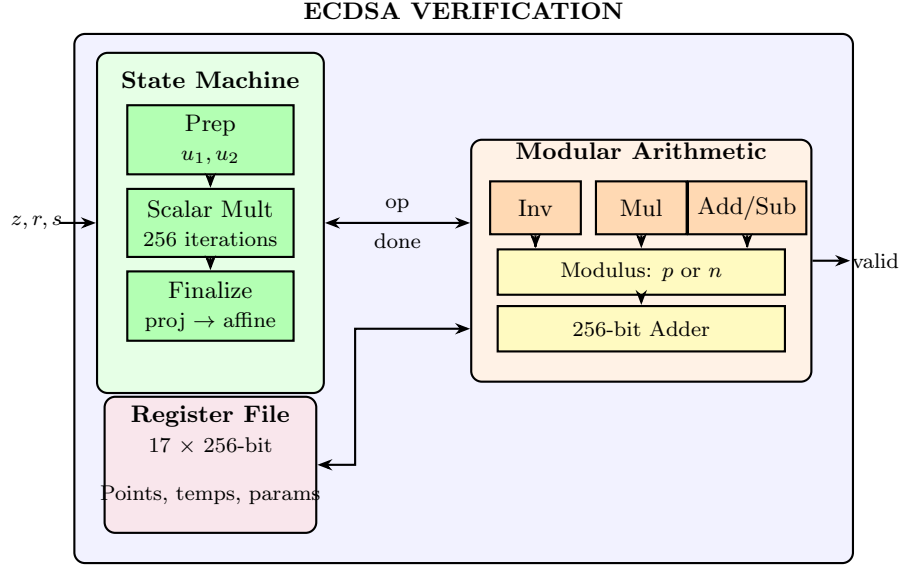


Figure 2: ECDSA verification module. The state machine orchestrates scalar multiplication using Shamir’s trick (processing both u_1 and u_2 in a single pass). Point operations use complete addition formulas in projective coordinates.

3.3 Design Rationale

Deadman’s switch: The original paper [4] argues that sending “off” signals to chips in unknown locations is unreliable. Chips may be air-gapped or shielded. Defaulting to off and requiring periodic re-authorization is more robust.

Redundancy through distribution: The full architecture calls for thousands of security blocks per chip. An attacker must bypass all of them. This prototype implements one block; duplication is straightforward.

Public key cryptography: Only the public key is stored on-chip. There is no secret to extract. The prototype hardcodes $Q = 2G$ for testing; production implementations would store chip-specific keys in Mask ROM.

ECDSA verification: The module uses Shamir’s trick to compute $R = u_1 \cdot G + u_2 \cdot Q$ in a single pass through 256 bit positions. Each bit position requires one point doubling and one conditional addition, executed as a fixed 40-operation sequence. Point operations use complete addition formulas [5] in projective coordinates, where the point at infinity is $Z = 0$. This eliminates data-dependent branches, simplifying verification.

Nonce rotation: Invalid signatures return to Publish with the same nonce, allowing retry. Valid signatures trigger immediate nonce regeneration.

Time-based depletion: Allowance decrements every cycle rather than per-operation, ensuring expiration even during idle periods.

4 Results

4.1 Artifacts

Code and extended architecture documentation are available at <https://github.com/JamesPetrie/off-switch>.

4.2 Simulation Results

Simulation validates the following security properties:

- **Output gating:** When allowance reaches zero, outputs are forced to zero. When positive, computation proceeds normally.
- **Replay prevention:** After accepting a valid license, the block generates a new nonce. The same signature cannot be reused.
- **Fail-secure initialization:** Allowance starts at zero. Without a valid license, nothing works.
- **Invalid license handling:** Bad signatures do not affect allowance and do not change the nonce.
- **Cryptographic authorization:** Only signatures valid for the current nonce increment the allowance.

5 Limitations

Prototype status: This is a demonstration, not production-hardened code. Deployment would require thorough verification.

Circuit size: The original paper [4] targeted roughly 10,000 gates per block. This prototype is substantially larger due to redundant arithmetic circuits, excess registers, and the additional state required by complete addition formulas. Optimization would be needed to approach the original target.

Counter-based TRNG: The prototype uses an incrementing counter for nonce generation. This is acceptable for testing, not deployment. The interface supports substituting a ring oscillator.

Missing orchestration: A real chip needs logic to collect nonces from thousands of blocks and distribute licenses. This prototype implements one block with direct external interfaces.

Quantum vulnerability: ECDSA will not survive capable quantum computers. Production systems should include alternative block designs using post-quantum cryptography, as [4] suggests.

6 Future Work

Verification: The current test suite covers basic security properties. More thorough verification would include edge case testing, code review, and testing against malformed inputs.

Formal verification: The state machine has seven states and the security properties are enumerable. The use of complete addition formulas eliminates edge-case branches, and projective coordinates handle the point at infinity without special cases.

Circuit optimization: Sharing arithmetic circuits, reducing register count, and evaluating the size/verification tradeoff for complete formulas.

Alternative designs: The original paper proposes multiple independent block designs with different cryptographic assumptions, including post-quantum variants. Implementing alternatives would add defense in depth.

Orchestration: Building the on-chip infrastructure to manage thousands of blocks, possibly using existing microcontroller cores on AI accelerators.

7 Conclusion

This project built a working prototype of an embedded off-switch security block and documented its architecture. The implementation demonstrates the core authorization flow and security properties in simulation. While not yet production-ready, it provides a starting point for further development and security analysis.

Code

Repository: <https://github.com/JamesPetrie/off-switch>

The ECDSA module predates the hackathon. The security block integration, state machine, TRNG interface, and documentation were developed during the hackathon.

LLM Usage

Claude assisted with implementation, debugging, and drafting. Functionality was verified through simulation.

References

- [1] Aarne, O., Fist, T., & Withers, C. (2024). Secure, governable chips: Using on-chip mechanisms to manage national security risks from AI & advanced computing.
- [2] Kulp, G., et al. (2024). Hardware-enabled governance mechanisms: Developing technical solutions to exempt items otherwise classified under export control classification numbers 3a090 and 4a090.
- [3] Petrie, J. (2024). Near-term enforcement of AI chip export controls using a firmware-based design for offline licensing.
- [4] Petrie, J. (2025). Embedded Off-Switches for AI Compute. *arXiv preprint* arXiv:2509.07637.
- [5] Renes, J., Costello, C., & Batina, L. (2016). Complete addition formulas for prime order elliptic curves. *IACR Cryptology ePrint Archive*, 2015/1060.

A Limitations & Dual-Use Considerations

A.1 Technical Limitations

False security from insecure implementations: A flawed implementation could give false confidence that chips are controlled when they are not. If the ECDSA verification has subtle bugs, or if the TRNG produces predictable nonces, attackers might forge valid licenses. This risk motivates the emphasis on formal verification and the use of complete addition formulas that eliminate edge cases.

Accidental denial of service: Bugs or operational failures could disable chips that should remain operational. If the licensing authority’s systems fail, or if network connectivity is lost, legitimate users lose access to their hardware. The time-based allowance provides a buffer, but extended outages could still cause disruption. Production systems would need redundant licensing infrastructure and appropriate allowance periods.

A.2 Dual-Use Risks

Concentration of power: Hardware off-switches grant significant power to whoever controls the licensing keys. If a single entity controls authorization for a large fraction of global AI compute, that concentration could be misused for censorship, rent-seeking, or geopolitical coercion. Governance structures around key management would need careful design, potentially involving multiple independent parties or threshold signatures.

Encouraging parallel supply chains: If off-switches are mandated in some jurisdictions but not others, actors seeking ungovernable compute may invest in alternative chip fabrication. This could accelerate development of competing supply chains outside the reach of governance mechanisms, potentially worsening the problem the technology aims to solve.

Sabotage and weaponization: The same mechanism that allows legitimate authorities to disable chips could be exploited by attackers who compromise the licensing infrastructure. A compromised private key could disable chips globally. Conversely, state actors might use off-switches as a tool of economic warfare rather than safety governance.