K2 is a new architecture and verification approach for hardware security modules (HSMs).

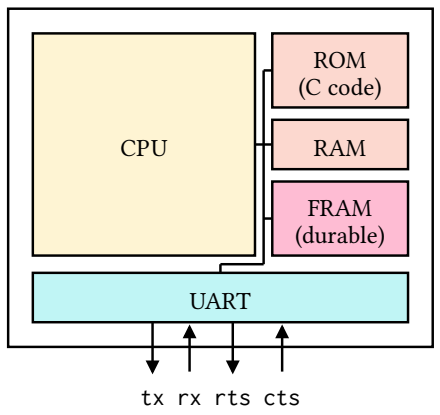
K2 uses rigid separation between I/O, storage, and computation over secret state to enable modular proofs while providing correctness/security guarantees across hardware and software.

anish.io/k2

The K2 Architecture for Trustworthy Hardware Security Modules

Anish Athalye[1], M. Frans Kaashoek[1], Nickolai Zeldovich[1], Joseph Tassarotti[2]
[1]MIT CSAIL, [2]NYU

An HSM implements a functional specification while aiming to be **free of hardware/software security bugs and timing side channels**.



~

```
# CA certificate signing HSM

var signing_key = null

def initialize(new_key):
  signing_key = new_key

def sign_certificate(cert):
  rsa_sign(signing_key, cert)
```

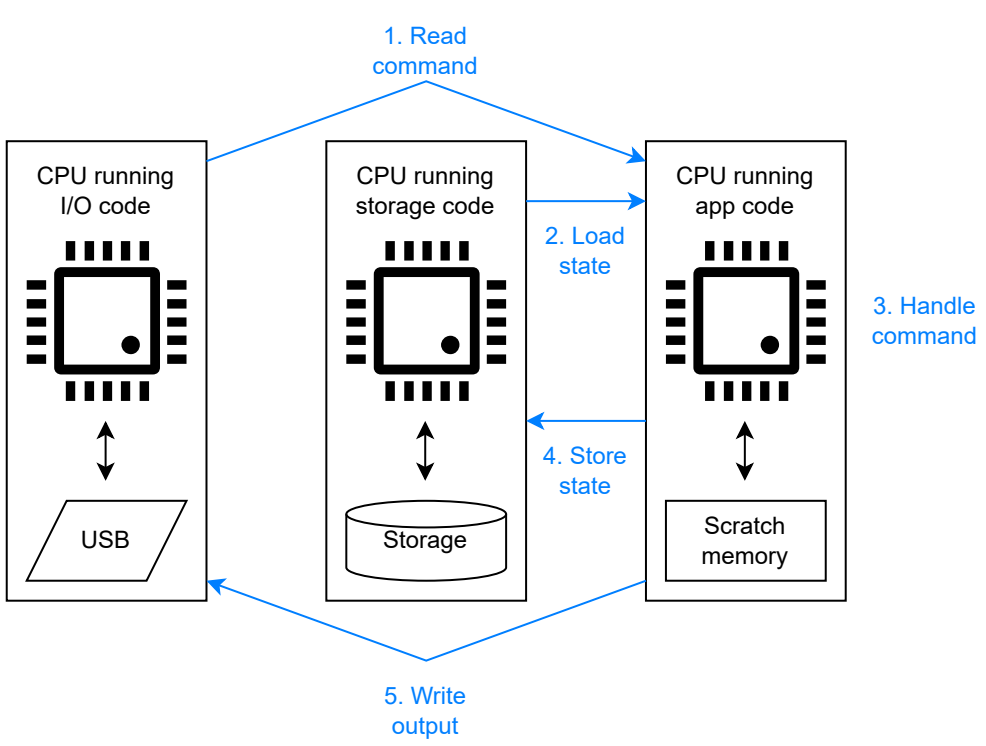
Implementations can **reuse existing crypto software verified for correctness** (we use HACL★).

We set up computation over secret state so that it runs end-to-end without interruption or intermediate observables. For verifying security and absence of timing side channels, we use a new tool called Chroniton that **proves that code runs in constant time at the hardware level**, using symbolic execution of the entire circuit at a cycle-accurate level.

```
void handle_command(
    char *state,
    char *command,
    char *new_state,
    char *response)
{
    ...
}
```

Chroniton => "for all inputs, runs in exactly 11,327,118 cycles on the OpenTitan SoC."

K2 is an architecture for HSMs that **separates** I/O, reading/writing persistent storage, and computing over secret state into logically-separate phases that run **as if running on isolated devices**.



K2 implements the logical design on a single SoC / single CPU using a **tiny kernel that runs phases in sequence**, enforcing isolation using the RISC-V PMP and clearing micro-architectural state between phases.