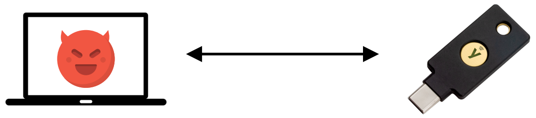
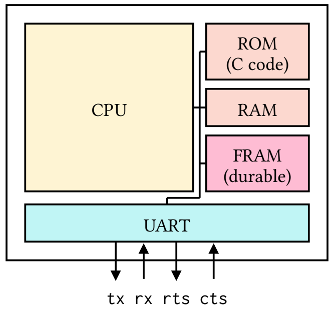
Verifying Hardware Security Modules with Information-Preserving Refinement

Anish Athalye, M. Frans Kaashoek, Nickolai Zeldovich
MIT CSAIL

Threat model: adversary compromises host machine, **gaining full control over the I/O interface** to the HSM.



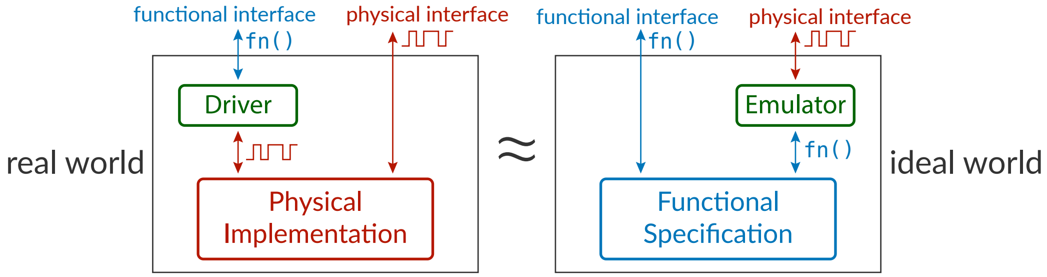
Our approach relates an HSM implementation's **wire-level behavior** to a functional specification's input-output behavior.



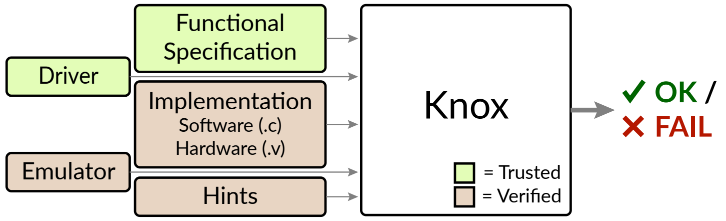
~

```
var bad_guesses = 0, secret = 0, pin = 0

def store(new_secret, new_pin):
  secret, pin = new_secret, new_pin
  bad_guesses = 0

def retrieve(guess):
  if bad_guesses >= 10:
    return 'No more guesses'
  if guess != pin:
    bad_guesses = bad_guesses + 1
    return 'Incorrect PIN'
  bad_guesses = 0
  return secret
```

Information-preserving refinement (IPR) says the **implementation's wire-level / timing behavior leaks no information**.



We built the *Knox* framework for **verifying HSMs with IPR**.



We **built and verified 3 simple HSMs**, and we showed that our approach catches hardware/software bugs and timing channels.

HSM	Spec		Driver	HW	SW	Proof
	core	total				
PIN-protected backup HSM	32	60	110	2670	190	470
Password-hashing HSM	5	150	90	3020	240	650
TOTP token	10	180	80	2950	360	830

Knox is a new framework for building hardware security modules (HSMs) with high assurance through formal verification.

Using a new security definition called ***information-preserving refinement***, Knox helps developers rule out hardware bugs, software bugs, and timing side channels in HSMs.

anish.io/knox