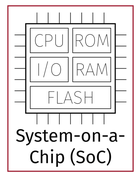
Modular Verification of Secure and Leakage-Free Systems: From Application Specification to Circuit-Level Implementation

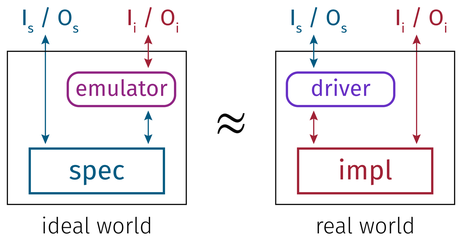
Anish Athalye[1], Henry Corrigan-Gibbs[1], M. Frans Kaashoek[1], Joseph Tassarotti[2], and Nickolai Zeldovich[1]
[1]MIT CSAIL [2]New York University

```
var prf_key, prf_counter, sig_key

def initialize(new_prf_key, new_sig_key):
    prf_key = new_prf_key
    prf_counter = 0
    sig_key = new_sig_key

def sign(message):
    if prf_counter == 2^64 - 1:
        return Error
    nonce = hmac_sha256(prf_key, prf_counter)
    prf_counter += 1
    return ecdsa_p256(message, sig_key, nonce)
```

$\approx_{\text{IPR}}$



Parfait relates an **application-level specification** to a **circuit-level implementation**.



Information-preserving refinement (IPR) **rules out correctness bugs, security bugs, and timing side-channel leakage**.

$$\frac{M_1 \approx_{IPR[d_{12}]} M_2 \qquad M_2 \approx_{IPR[d_{23}]} M_3}{M_1 \approx_{IPR[d_{12} \circ d_{23}]} M_3}$$

Parfait formalizes IPR and proves its **transitivity** to enable modular proofs.

HSM	Spec	Driver	Platform	Implementation Software	Implementation Hardware
ECDSA signer	40 LoC	100 LoC	Ibex	2,300 LoC	13,500 LoC
			PicoRV32	2,300 LoC	3,000 LoC
Password hasher	30 LoC	100 LoC	Ibex	1,000 LoC	13,500 LoC
			PicoRV32	1,000 LoC	3,000 LoC

We **built and verified 4 HSMs**, including a PKCS#11-compatible ECDSA signature HSM.

App	Proof	Dev time
ECDSA signer	500 LoC	-
Password hasher	200 LoC	Δ 2 hours

Platform	Proof size (LoC) Emulator	Hints	Mapping	Dev time	Verification ECDSA signer Time	Cycles/s	Password hasher Time	Cycles/s
Ibex	50	250	10	-	80 hrs	304	0.10 hrs	289
PicoRV32			10	Δ 2 hours	100 hrs	671	0.14 hrs	588

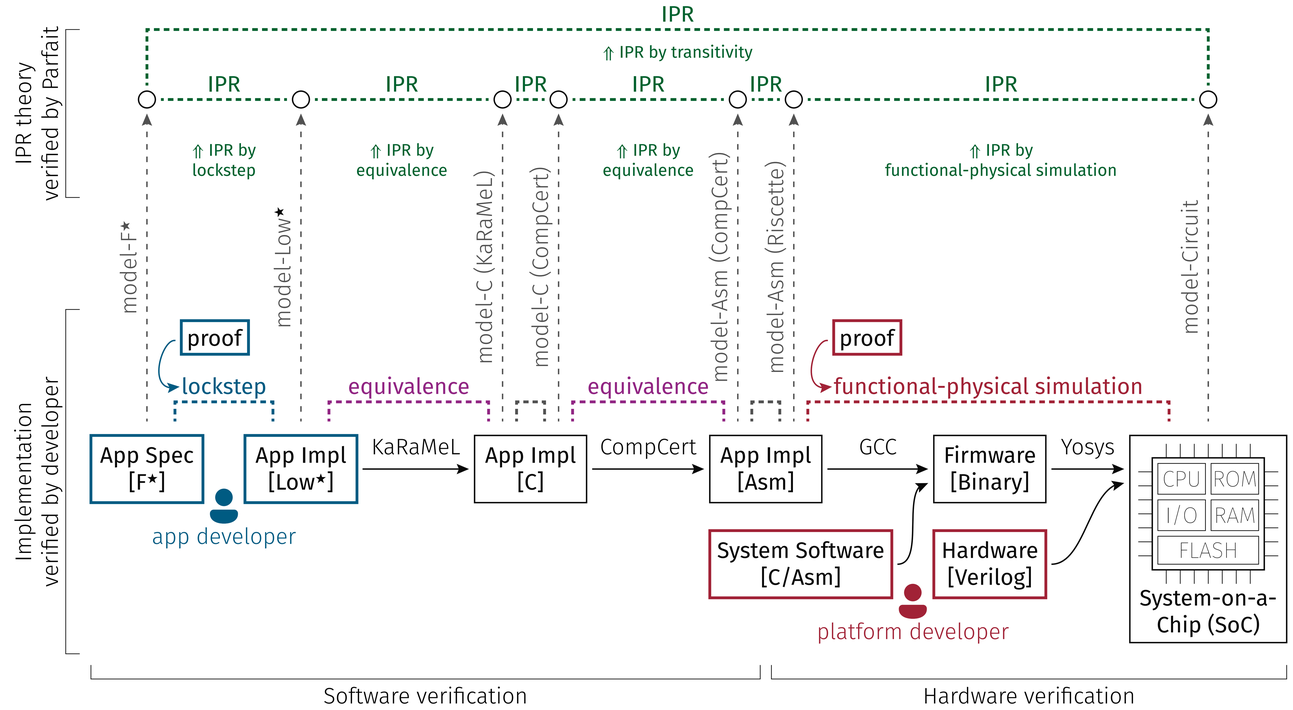
Parfait enables **verifying new applications and porting to new hardware platforms** with moderate effort.

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

Parfait's verification approach rules out software bugs, hardware bugs, and timing side channels in HSMs.

Parfait scales to sophisticated HSMs—we used Parfait to build and verify a PKCS#11-compatible ECDSA signature HSM.

anish.io/parfait



Parfait developer workflow and verification approach.