Securing Functional Programs with Hardware Support

WG 2.8, Utrecht, 2024

KC Sivaramakrishnan

IIT MADRAS

Tarides

300,000,000

Voter Absenteeism in India

- 1/3 of eligible Indian voters do not cast their votes[1]
  - ~300 million in 2019
- **Reasons:** voter apathy, *internal migration*
- 450 million internal migrants in India[2]
  - Work and family reasons
- Constituency can be updated
  - But not — no emotional connection, transient residence
- ECI → IIT Madras, *"How to enable remote voting?"*

[1] "Discussion on improving voter participation of domestic migrants using remote voting", Election Commission of India

[2] "Internal Migration in India Grows, But Inter-State Movements Remain Low", Census of India 2011

Electronic Voting Machines (EVM) in India

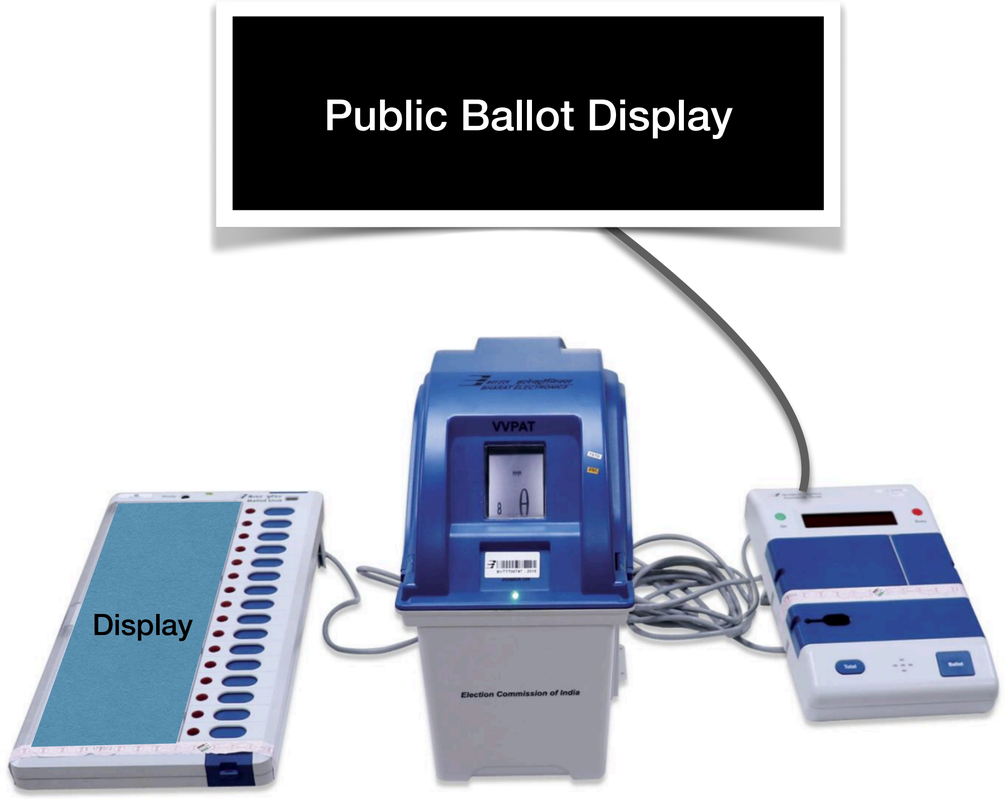
- EVMs used in India since 1990s
- EVM unit has
  - Physical candidate list + Physical buttons
  - Voter-verified paper audit trail (VVPAT)
    - Small fraction per constituency cross-checked
- Technology[1]
  - Processor
    - NXP MK61FX512VMD12 (ARM Cortex-M4) and others
  - Software built by ECIL and BEL
    - Closed source

Ballot Unit ***VVPAT*** ***Control Unit***

[1] Patrick Jones, "Delivering Democracy: The History and Deployment of Electronic Voting Machines in India and the United States", PhD thesis

Remote Voting Machine (RVM)

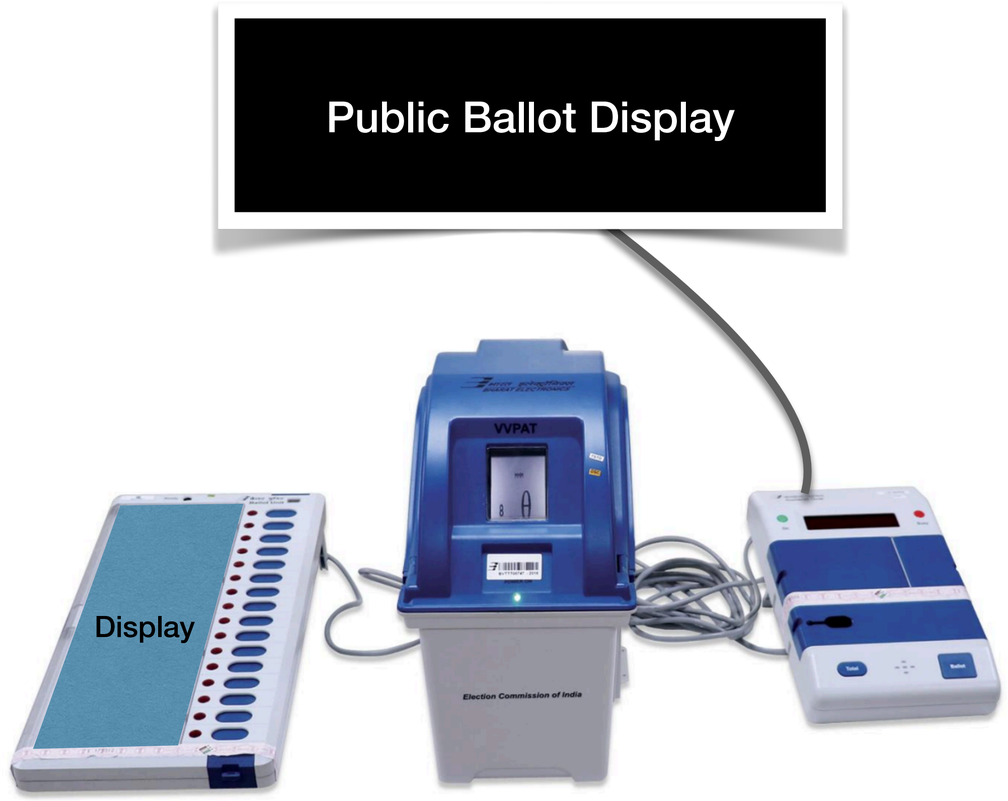
- Process[1]
  - Remote voters ***pre-register*** to vote from a remote constituency
  - Ballot list pre-loaded onto the machine
- Technology
  - No network connection
  - Ballots for different constituencies
  - Electronic display



[1] “Discussion on improving voter participation of domestic migrants using remote voting”, Election Commission of India, 2022

Remote Voting Machine (RVM)

- Long road ahead
  - Need amendments in the Constitution to allow remote voting
  - Social challenges — Building trust
  - **Technological challenge** — *more complex than EVM*



[1] “Discussion on improving voter participation of domestic migrants using remote voting”, Election Commission of India, 2022

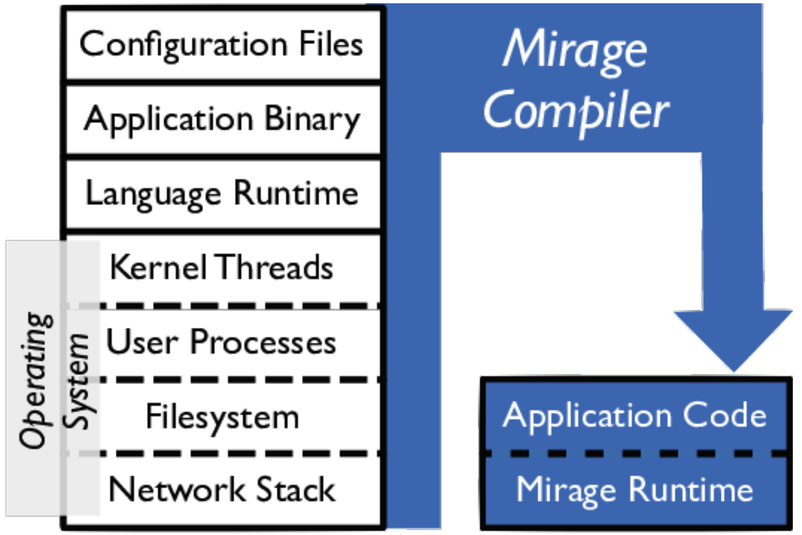
An idealised RVM stack

- Desiderata
  - Open-source — secrecy is not security
  - Hardware and software security features
  - Formally-verified software, where possible
  - Small trusted computing base (TCB)
- Out of scope[1]
  - E2E cryptographic verifiability: *cast-as-intended, recorded-as-cast, counted-as-recorded*
  - End-of-poll audit
- *Shakti RISC-V processor* running *bare-metal MirageOS unikernels*

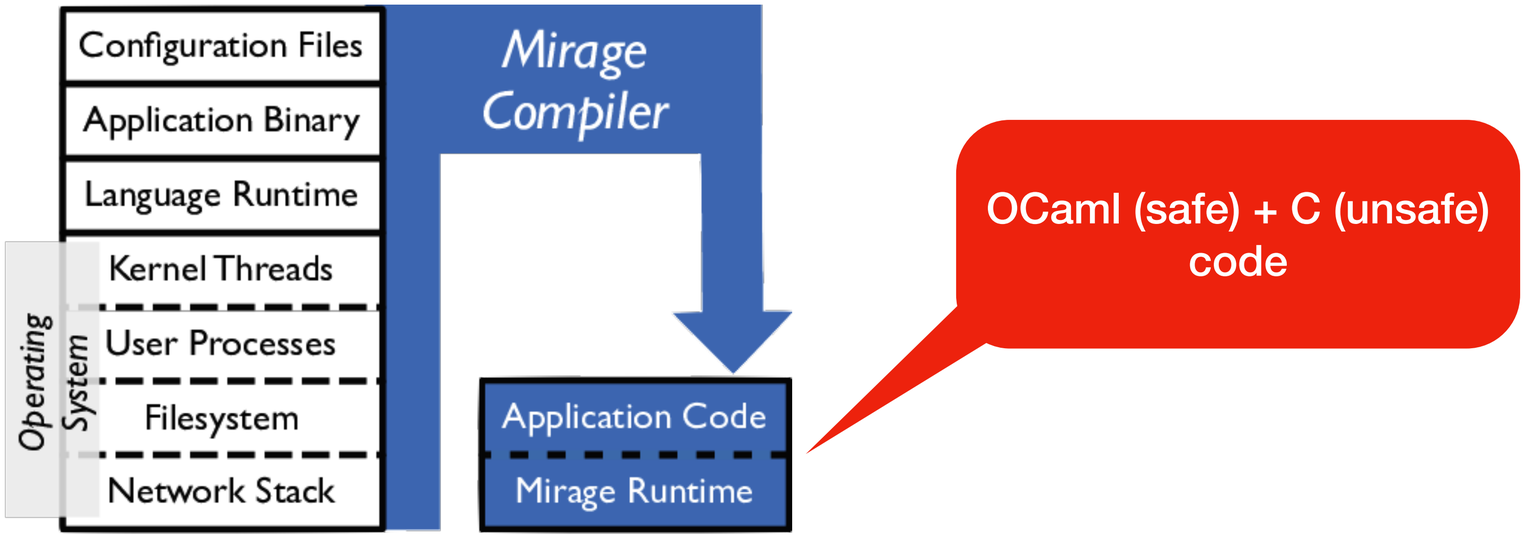
[1] "Citizens' Commission on Elections' Report on EVMs and VVPAT", IIT Delhi

Hardware and Software

- Shakti RISC-V processor family
  - https://gitlab.com/shaktiproject
  - developed at IIT Madras
  - Bluespec SystemVerilog
  - C-class: 5-stage, in-order; can boot Linux
  - 2 tape-outs: 22nm Intel, 180nm SCL
- MirageOS Unikernels
  - https://mirageos.org/
  - A library operating system in OCaml
  - Small TCB
  - Runs on Xen and KVM hypervisors, baremetal



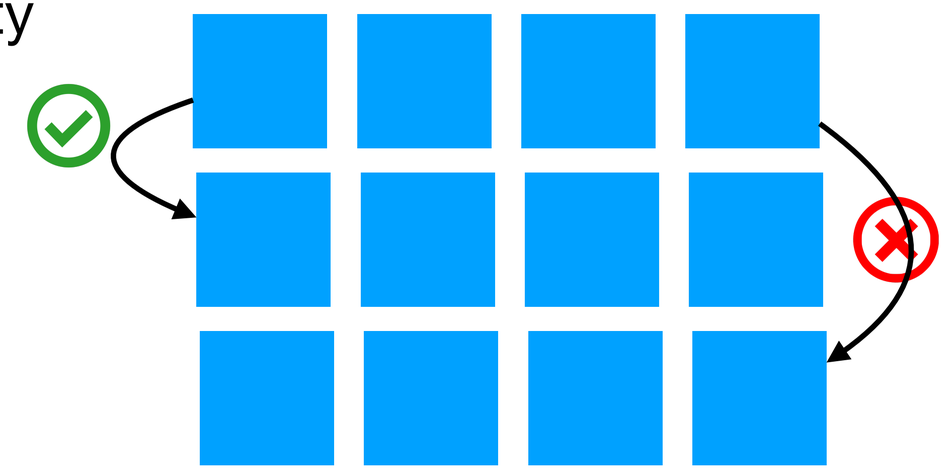
Challenge



- C code is unavoidable — OCaml runtime, mirage runtime, drivers, sqlite, crypto
  - C vulnerabilities can compromise OCaml's safety
- **Goal:** *Safe hardware/software stack for constrained environments*

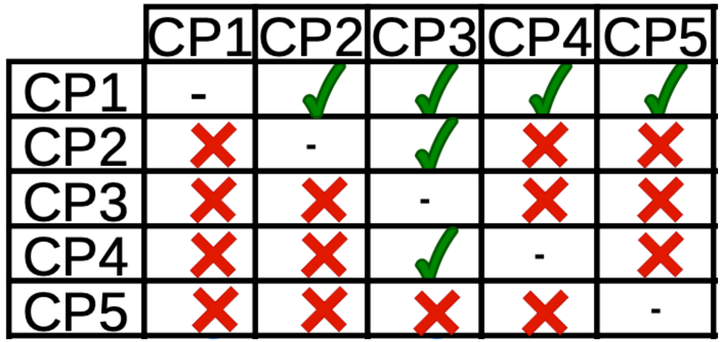
Compartments / SFI — overview

- Compartments offer *intra-process* isolation
  - Functions mapped to compartments
  - Restrict *control flow* and *data access* across security boundaries
- Control flow restricted by
  - Whitelisted PC ranges
  - Shadow stack to prevent ROP attacks
- Data access restricted by
  - VMM tricks (or) fat pointers (or) capabilities (à la CHERI)



FIDES — Secure compartments

- Security-hardened Shakti RISC-V processor + MirageOS unikernels
- Intra-process compartments
  - Vulnerabilities in C do not affect OCaml
- Compartment access matrix defined at *link time*
  - Run *unmodified* OCaml and C code
- Small extension to hardware and software
  - Two new instructions added to RISC-V ISA: **Val** and **Checkcap**
  - Modification to LLVM and OCaml compiler to emit these instructions



Access Matrix

Threat model

- Source code is untrusted
  - Inline assembly and use of Obj.magic trusted
- All code is compiled with FIDES C and OCaml compiler
  - Compiler instrumentation added by FIDES is correct
  - OCaml runtime is trusted
- Binary executable cannot be tampered with
- Hardware attacks rowhammer, fault attacks, side-channels are out of scope

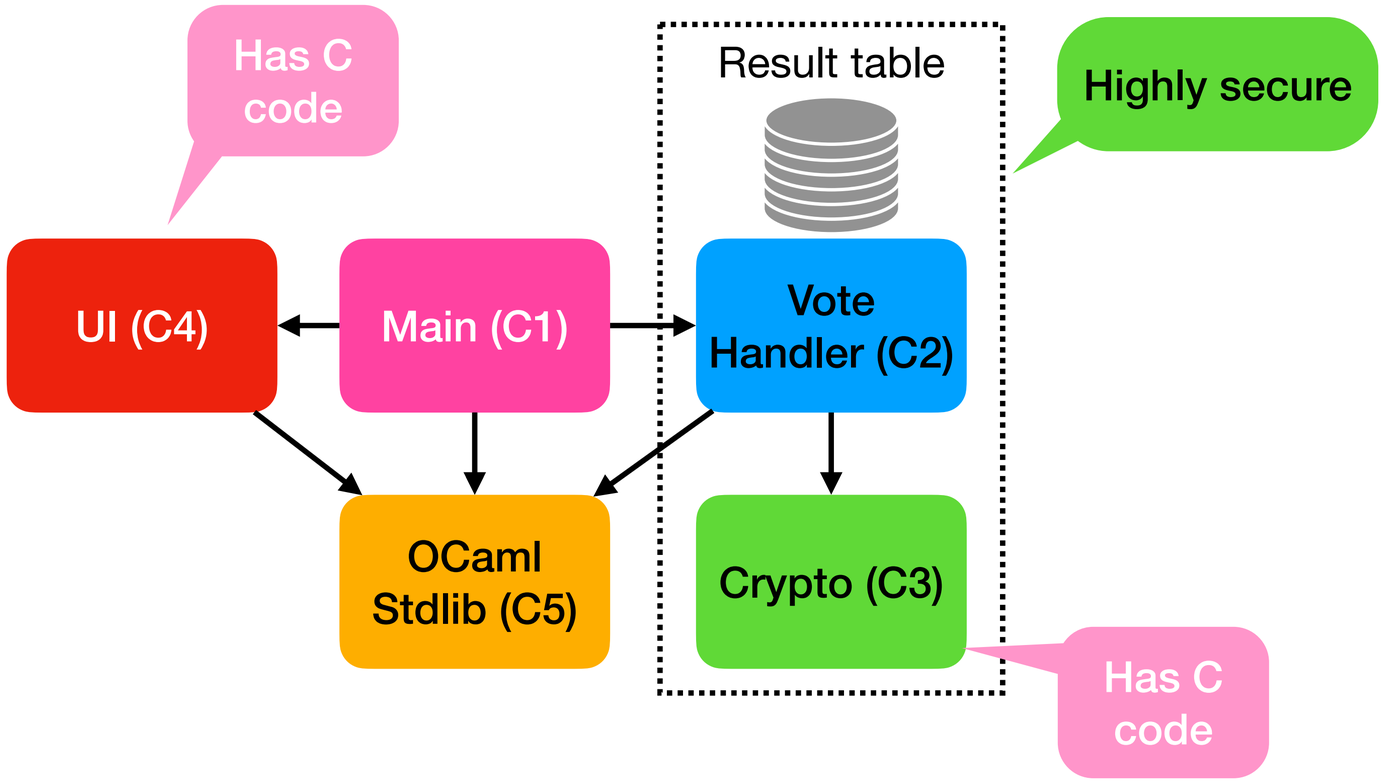
Guarantees

- **Control-flow integrity**
  - The control flow in every execution of the program respects the compartment access matrix
- **Memory safety**
  - No memory errors; all references point to valid memory
  - Pointers cannot be forged

FIDES — Challenges and opportunities

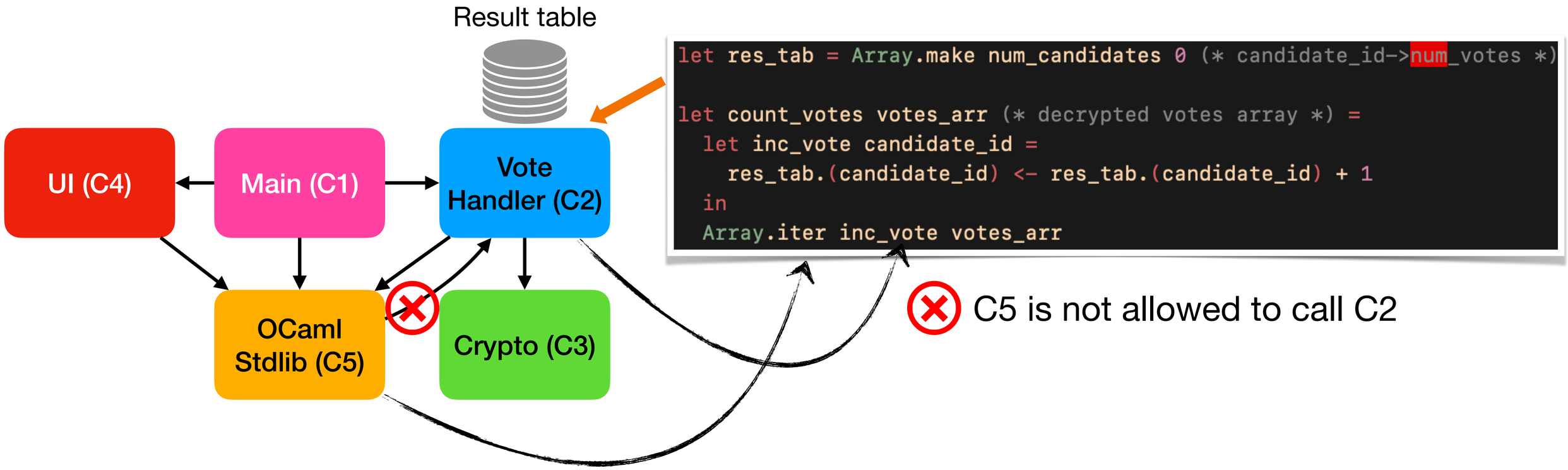
- OCaml offers memory safety
  - Hardware-accelerated fat pointers *only* for C code
    - Fine-grained data compartments
    - No fat pointers for OCaml code
  - Pay attention to FFI boundaries
- **FIDES code compartment must now handle FP features!**
  - Higher-order functions, tail calls, exceptions

Compartments for RVM

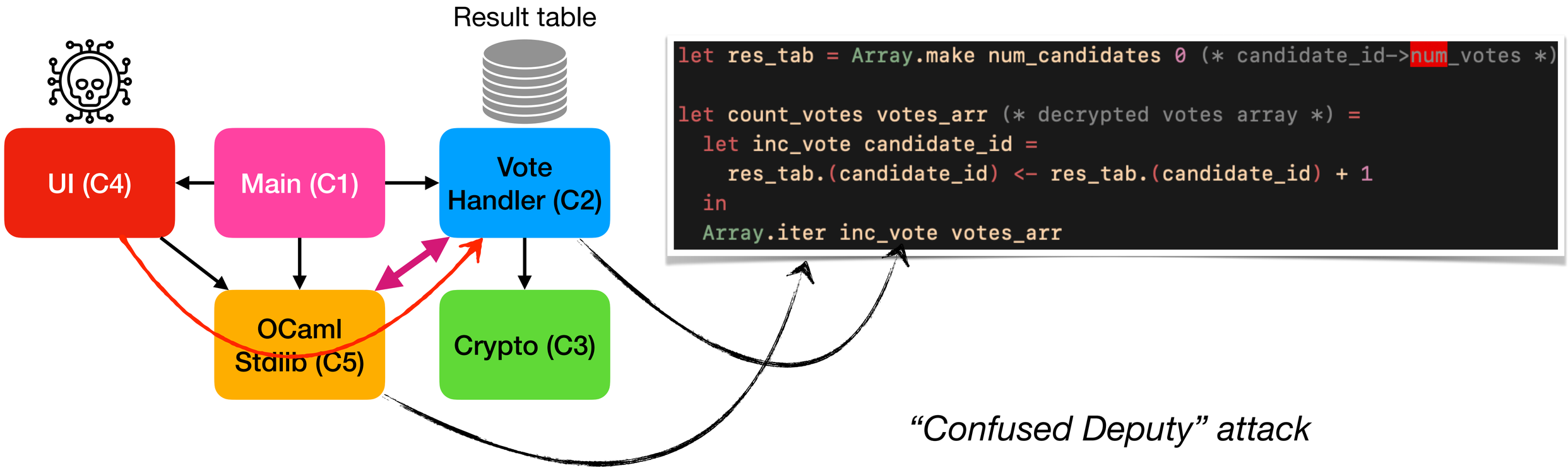


indicates call is allowed

Higher-order functions



Higher-order functions — Idea 1

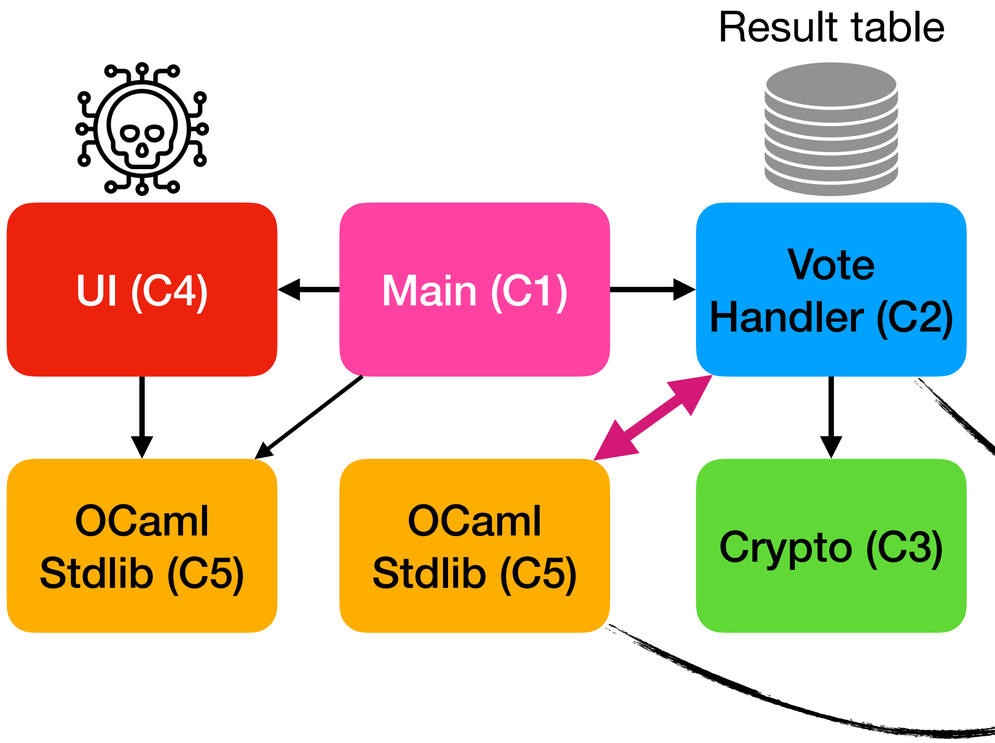



```
let res_tab = Array.make num_candidates 0 (* candidate_id->num_votes *)

let count_votes votes_arr (* decrypted votes array *) =
  let inc_vote candidate_id =
    res_tab.(candidate_id) <- res_tab.(candidate_id) + 1
  in
  Array.iter inc_vote votes_arr
```

“Confused Deputy” attack

→ indicates call is allowed

Higher-order functions — Idea 2

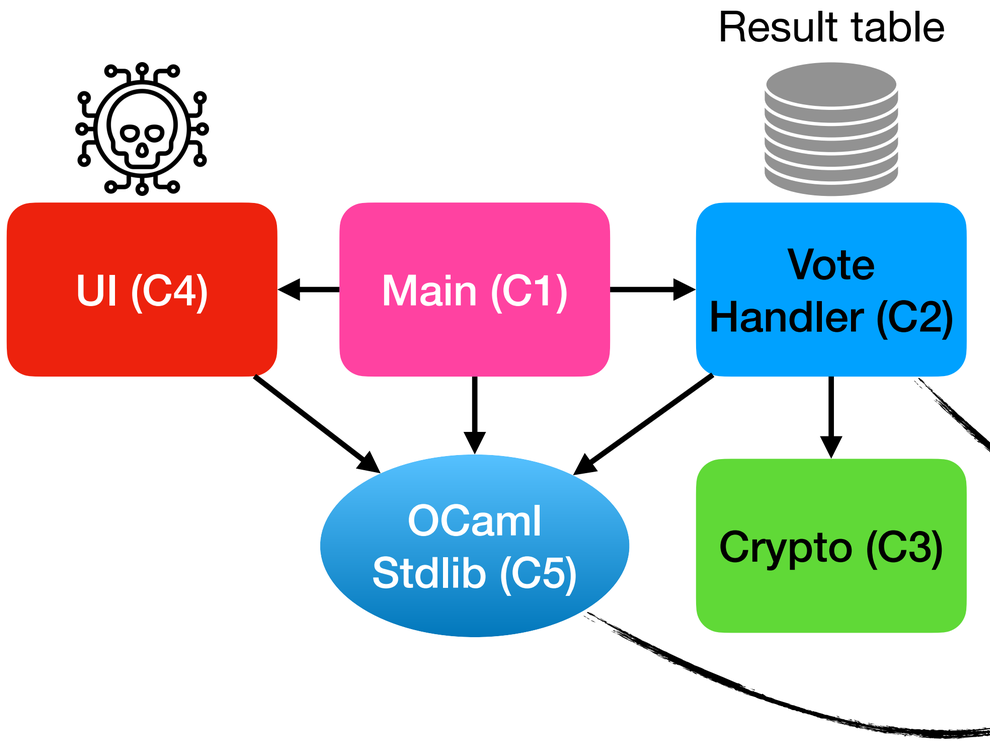



```
let res_tab = Array.make num_candidates 0 (* candidate_id->num_votes *)

let count_votes votes_arr (* decrypted votes array *) =
  let inc_vote candidate_id =
    res_tab.(candidate_id) <- res_tab.(candidate_id) + 1
  in
  Array.iter inc_vote votes_arr
```

- ⊗ Limited compartment resource
- ⊗ Shared state?

→ indicates call is allowed

Fluid compartments



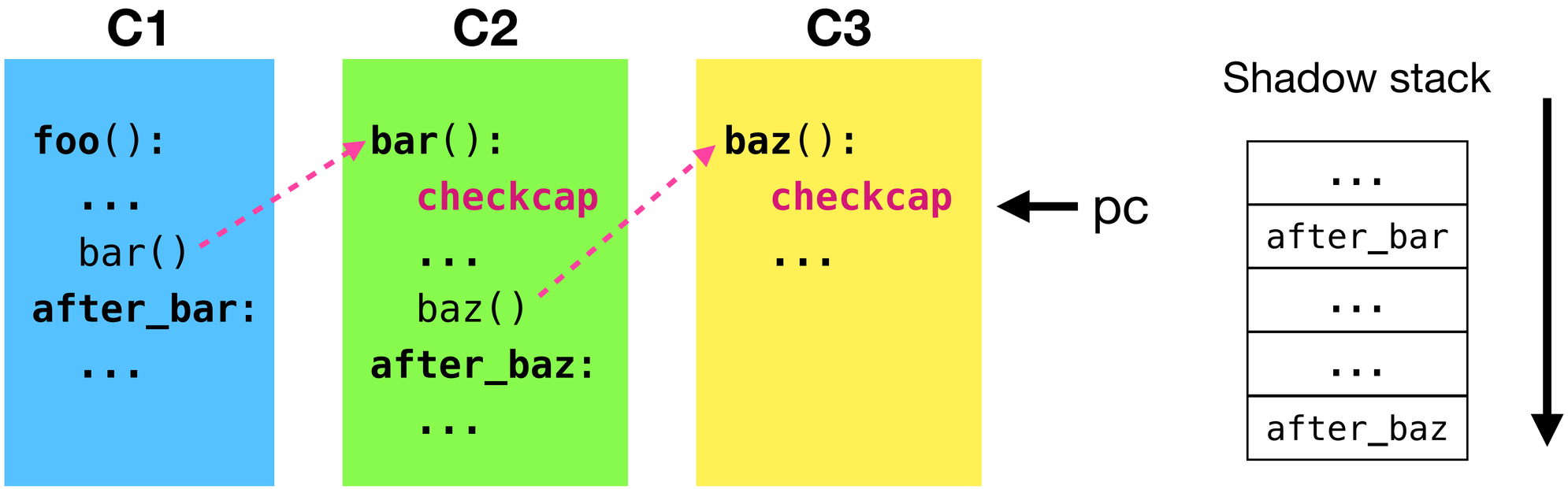

```
let res_tab = Array.make num_candidates 0 (* candidate_id->num_votes *)

let count_votes votes_arr (* decrypted votes array *) =
  let inc_vote candidate_id =
    res_tab.(candidate_id) <- res_tab.(candidate_id) + 1
  in
  Array.iter inc_vote votes_arr
```

A fluid compartment inherits the policies of the caller

indicates call is allowed

Shadow stack

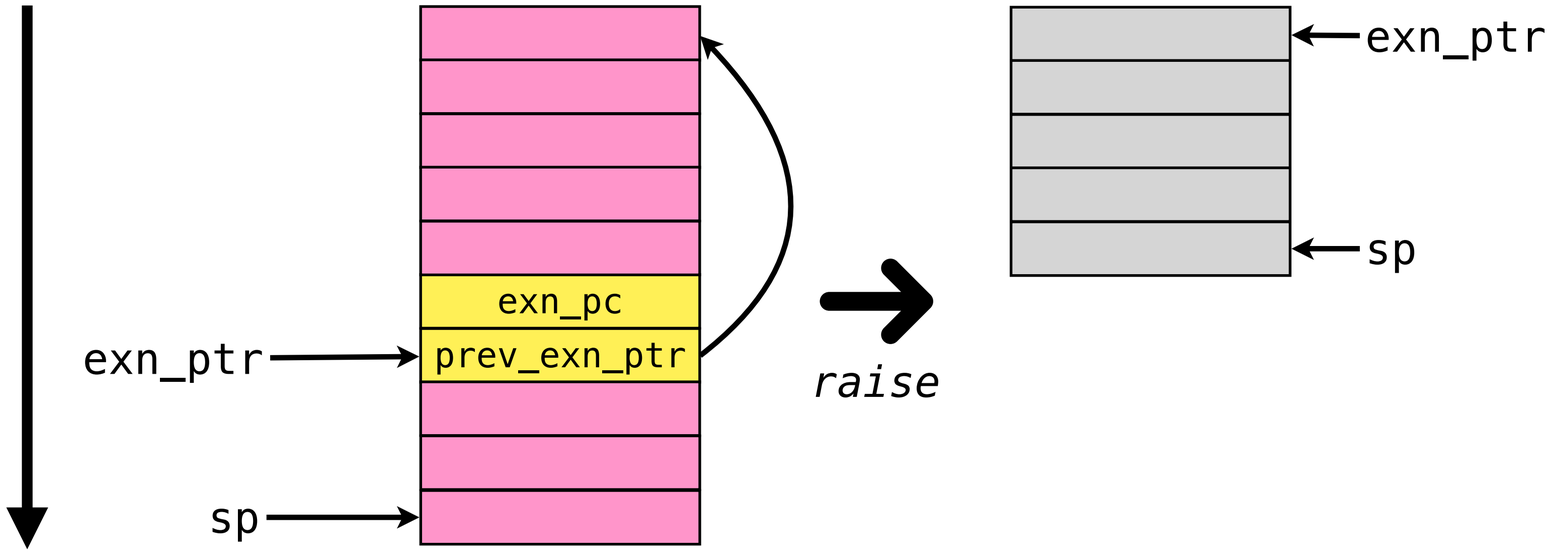
- Stores the return addresses for inter-compartment calls
- ***Inaccessible from user-code***
  - Maintained and validated by hardware



Non-call-return control flow

- Typical compartment schemes handle only call-return sequence
- OCaml has several non-call-return control-flow operations
  - *Tail calls, exceptions, effect handlers!*
  - Need to manage the shadow stack carefully

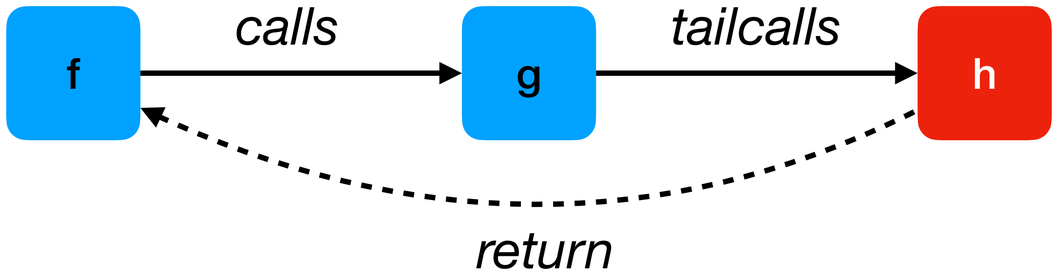
Exceptions and shadow stacks



- Exceptions may be thrown across compartments
  - Need to unwind shadow stack appropriately
  - **Challenge:** Detect when intra-compartment exceptions are raised
- **Solution:** Security monitor (SM) updates last `exn_pc` to a special routine

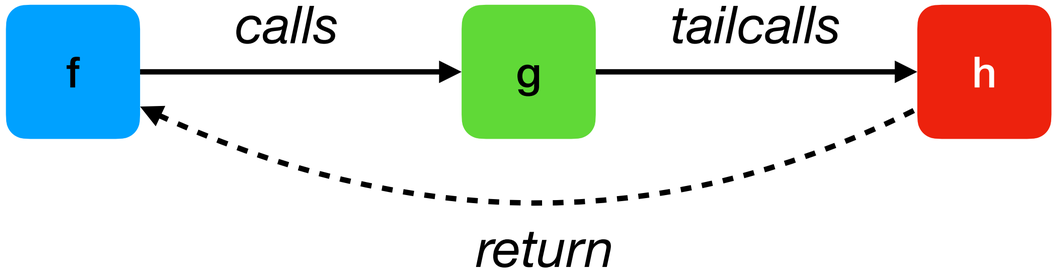
Tail calls and shadow stacks

Intra call then inter tailcall



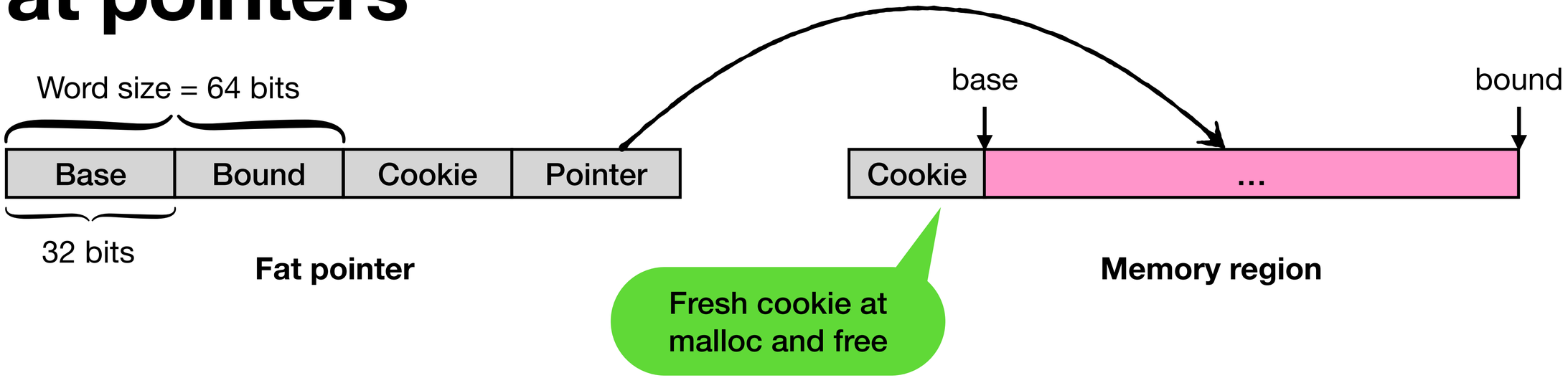
*At **g** to **h** call, return address at **f** pushed onto the shadow stack*

Inter call then inter tailcall



*At **g** to **h** call, nothing pushed onto the shadow stack*

Fat pointers



- Fat pointers into the stack have frame scope
  - Each frame has a cookie *freshened* at call and return
- **val** instruction validates fat pointer before access
- OCaml does not use fat pointers
  - At FFI, use OCaml object header info to create fat pointer
  - Use a special cookie that skips temporal validation

Hardware changes

```
# for each instruction
hw_check:
  if (not (pc in pc_range(cur_comp) or
           pc in pc_range(fluid_comp))):
      assert (*pc == checkcap)
      jmp sm_entry

sm_exn:
  checkcap

sm_ret:
  checkcap
```

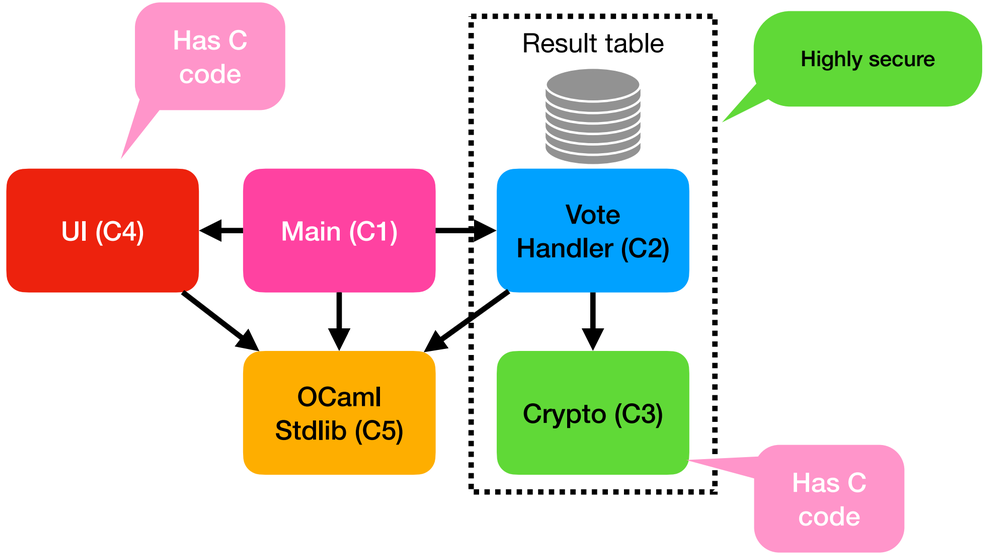
Hardware changes

```
14 sm_entry:
15   caller = ra # caller's return address
16   callee = csr_comepc # callee's entry address
17   if (callee == sm_ret):
18     # inter-compartment return
19     jmp sm_exit
20   if (callee == sm_exn):
21     # handling inter-compartment exception
22     jmp sm_exn_unwind_shadow_stack
23   if (caller == sm_ret):
24     # inter-compartment tail call preceded by
25     # inter-compartment call
26     jmp l_tail_call
27   if (in_C(callee)):
28     save(sm_stack,C_callee_saved_registers)
29   else:
30     # used during GC for finding roots on the stack across
31     # compartments
32     push(sm_gc_stack,caller)
```

```
33   # Update the PC of the latest exception handler so
34   # that inter-compartment exceptions can be detected
35   exn_handler_pc = get_latest_exn_handler_pc()
36   if (exn_handler_pc != sm_exn):
37     set_lastest_exn_handler_pc(sm_exn)
38     push(sm_stack, exn_handler_pc)
39   push(sm_stack, caller)
40   push(sm_stack, sp)
41   push(sm_stack, cur_comp)
42   target_comp = compartment_id(callee)
43   l_tail_call:
44     if (not (access_allowed(cur_comp, target_comp))):
45         access_error()
46   zero_out(temp_and_callee_saved_registers)
47   cur_comp = target_comp # switch to the target compartment
48   ra = sm_ret
49   jmp callee
```

Evaluation

- Compiler changes
  - ~300 lines for OCaml, ~2300 lines for LLVM
- Protyped on Xilinx Artix-7 AC701 FPGA
  - 38.2K LUTs (+6.1% over base)
  - 17.4K registers (+6.0% over base)
- Performance on voting application
  - 4% increase in code size
  - 23% increase in instruction cycle count



Limitations

- **Features:** Effect handlers, parallelism
- OCaml runtime is trusted
  - **WIP:** Verified garbage collector for OCaml
- Data compartments are too weak
  - Objects shared across compartments remain accessible forever
  - Revocation through ownership and borrowing à la Rust
    - modal types in OCaml
- Hardware is exotic
  - Arm MTE for fat pointers in C?

References

- Panel on Indian Electronic Voting Machines (EVMs), EVT/WOTE ‘10
- Patrick Jones, “Delivering Democracy: The History and Deployment of Electronic Voting Machines in India and the United States”, PhD thesis
- “Internal Migration in India Grows, but Inter-State Movements Remain Low”, Census of India 2011
- “Citizens’ Commission on Elections’ Report on EVMs and VVPAT”, IIT Delhi
- “Discussion on improving voter participation of domestic migrants using remote voting”, Election Commission of India
- “FAQs on Security Features of The ECI-EVMs”, ECI