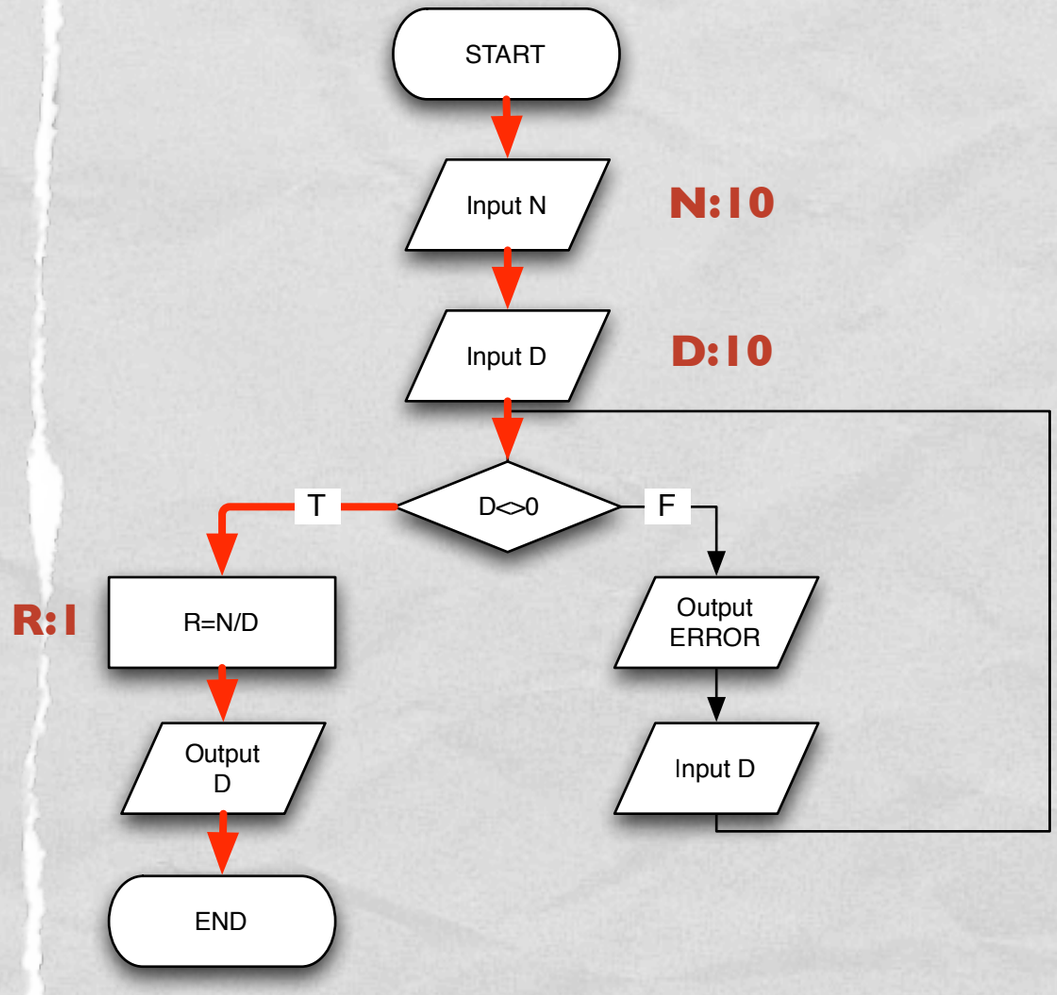
RE-TRUST 6th quaternary meeting
VILLACH (AUSTRIA)
12 March 2008

REMOTE ENTRUSTING BY REMOTE CONTROL FLOW MONITORING

Stefano Di Carlo
Politecnico di Torino

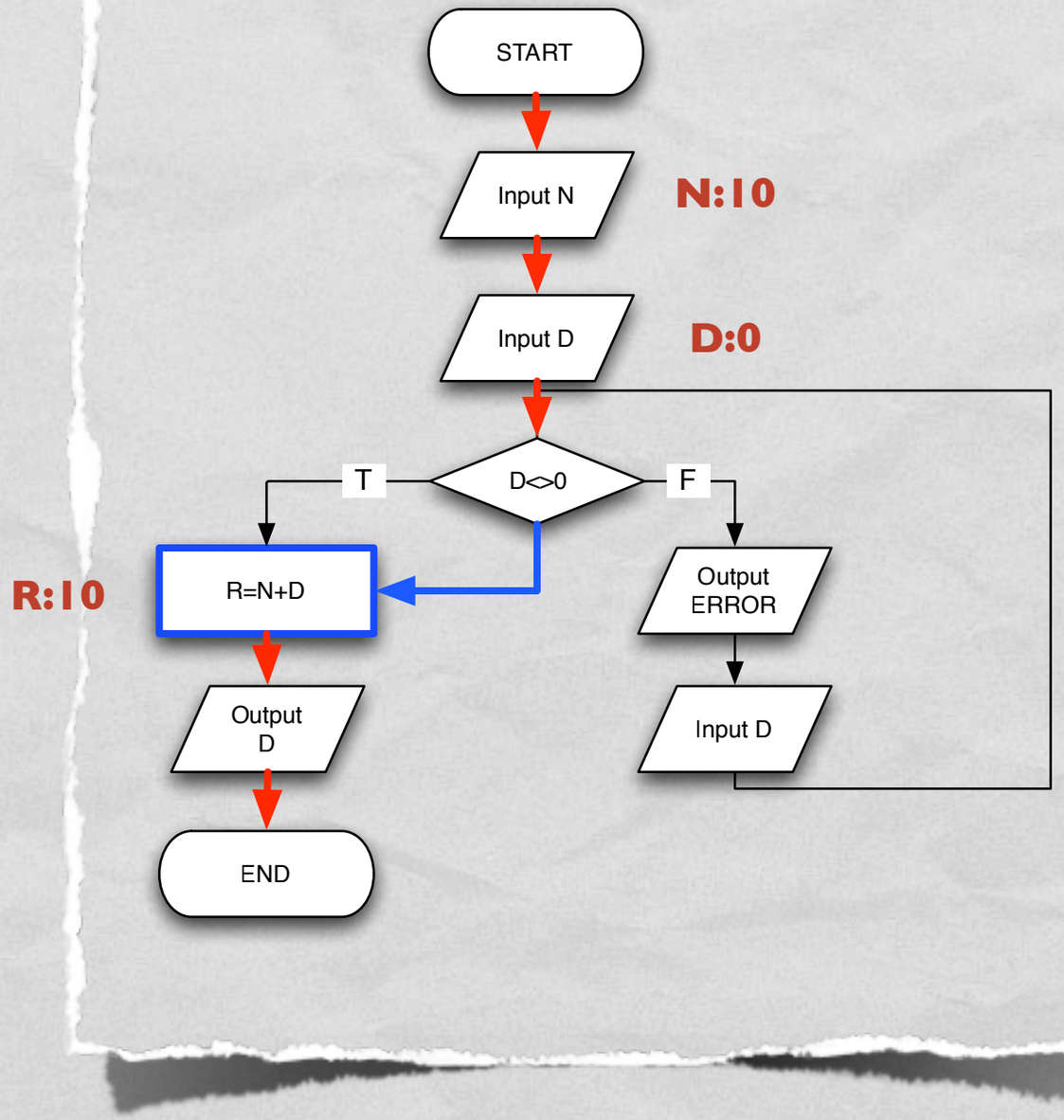
GOAL

- Using remote control flow monitoring to verify software code integrity.



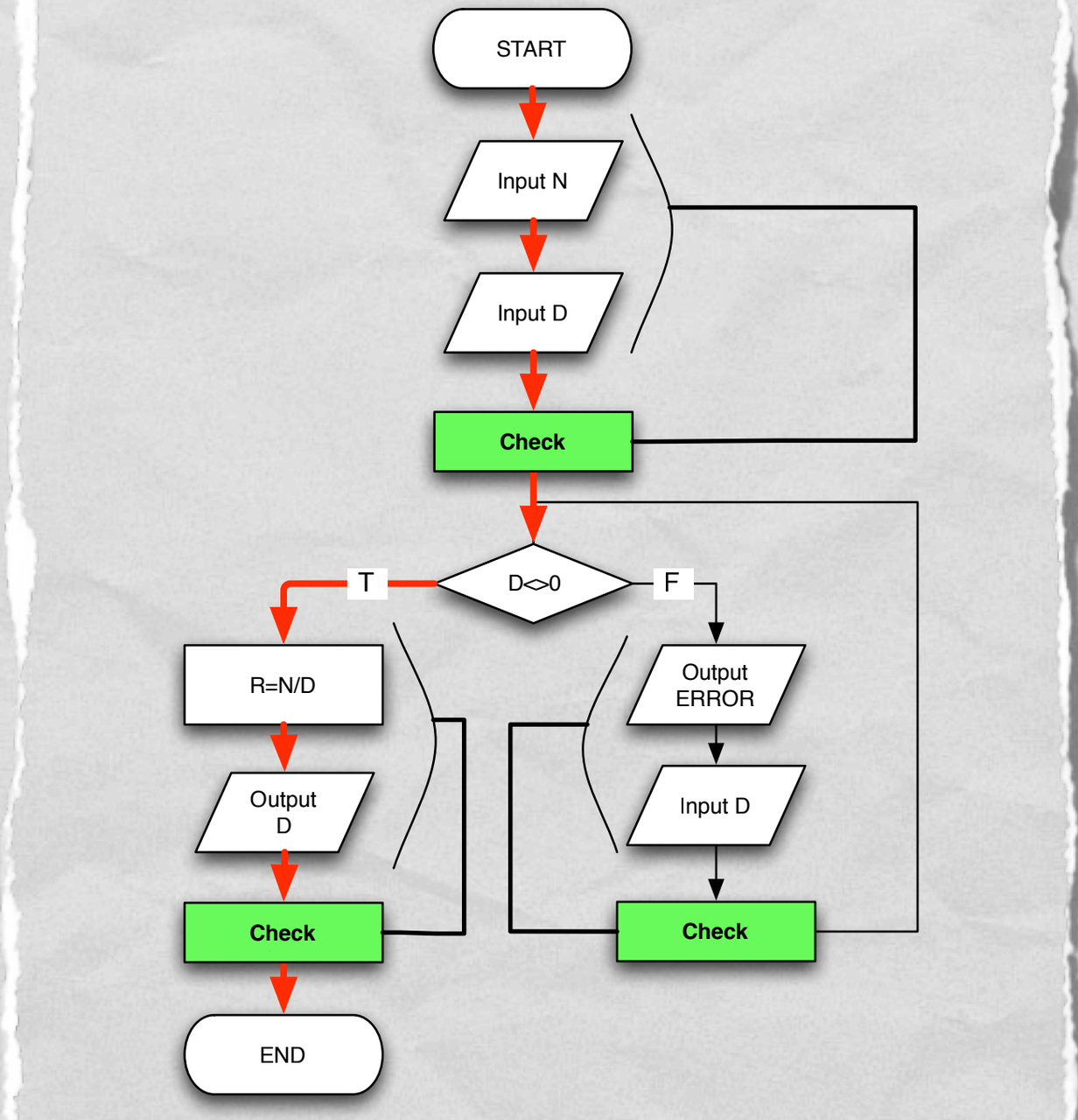
GOAL

- Target attacks:
  - Malicious modifications of instructions opcodes
- Malicious modifications of the program flow



CONTROL FLOW CHECKING

- Drawbacks:
  - Local checking only
  - Easy to bypass



REMOTE CONTROL FLOW CHECKING

- Split the program integrity verification among the untrusted and the trusted node:
  - Program execution performed on the untrusted node
  - Control flow validation performed on the trusted node

REMOTE CONTROL FLOW CHECKING

- Basic flow:
  - The target application collects information (traces) about executed instructions
  - Traces are transmitted from the untrusted node to the trusted node
  - The trusted node validates the control flow of the application
  - Any violation is detected as an attack

REMOTE CONTROL FLOW CHECKING

- Traces represented by checksums evaluated over the basic blocks of the application
- Drawbacks:
  - Cloning attack
- Solution:
  - Self modifying code

REMOTE CONTROL FLOW CHECKING

- Self modifying code:
  - The program contains a Modifying Instruction Table (MIT)

1	INC eax
2	MOV edx, 1982
3	ADD eax, 1
4	JMP next_instr

- An instruction is randomly selected to replace another instruction of the actual application code
  - Random generation algorithm and initial seed shared with the trusted server

REMOTE CONTROL FLOW CHECKING

- The trusted node is in charge of:
  - Monitoring the flow of instructions received from the untrusted node (correct sequence of basic blocks)
  - Validating the checksum of each basic block (correct instructions opcode)

REMOTE CONTROL FLOW CHECKING

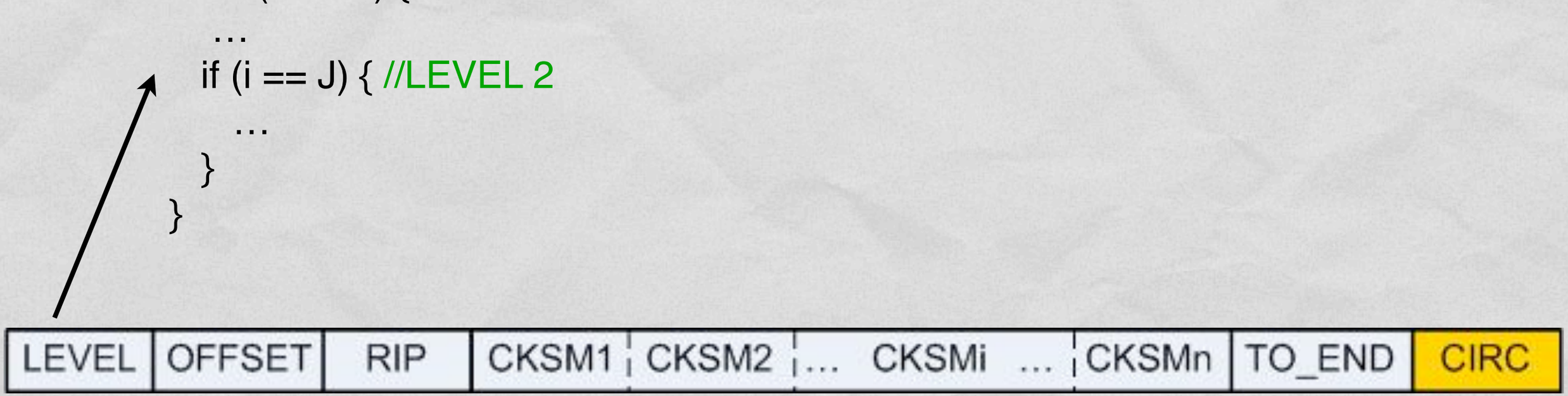
- Valid control flows represented through a Regular Expression String (RES) on the trusted node
- Each RES field represents a single basic block

RES FIELD

REMOTE CONTROL FLOW CHECKING

Basic block nesting level

```
while(i != 10) { //LEVEL 1
    …
    if (i == J) { //LEVEL 2
        …
    }
}
```



REMOTE CONTROL FLOW CHECKING

Basic block memory offset

```
while(i != 10) { //0x016
    …
    if (i == J) { //0x048
        …
    }
}
```

LEVEL	OFFSET	RIP	CKSM1	CKSM2	… CKSMi …	CKSMn	TO_END	CIRC

REMOTE CONTROL FLOW CHECKING

| → Option among two basic blocks

→ The *basic block* is mandatory.

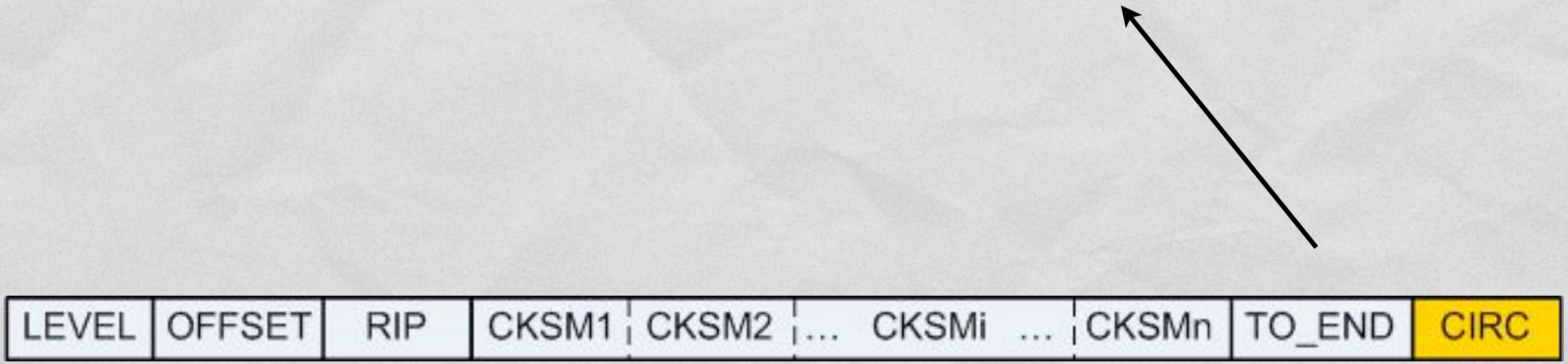
* → The *basic block* can be executed zero or more times

+ → The *basic block* can be executed one or more time

? → The basic block is optional

REMOTE CONTROL FLOW CHECKING

Identifies a terminal basic block

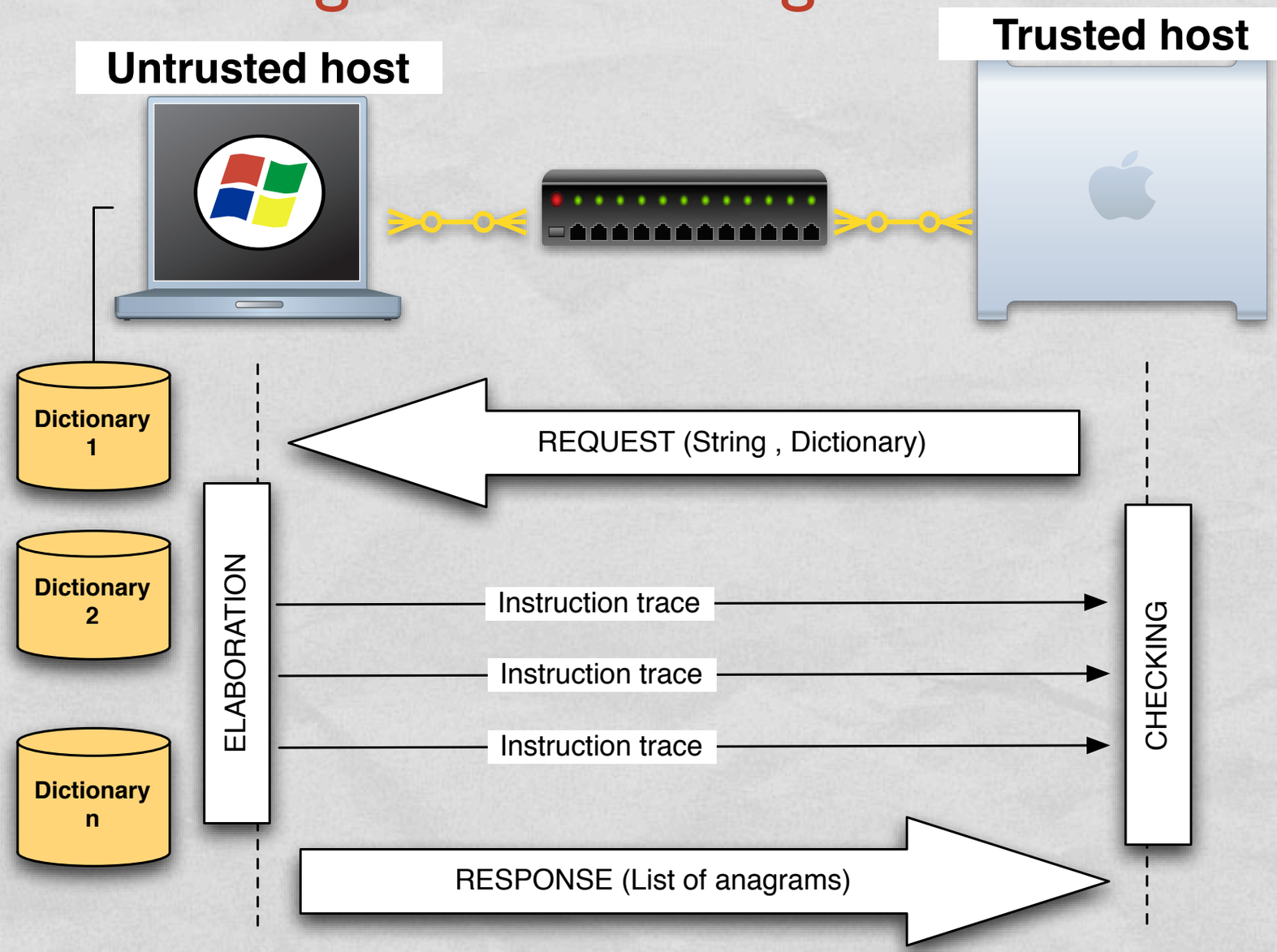


REMOTE CONTROL FLOW CHECKING

States if the missing notification of the basic block execution is accepted or not

EXPERIMENTAL RESULTS

- Remote anagrams searching



EXPERIMENTAL RESULTS

- Execution time:

Original Program	Control Flow Checking (SHA1)	Control Flow Checking (XOR)
0.015 sec	4.00 sec	0.125 sec

- Memory usage:

Original Program	Control Flow Checking (SHA1)
0.0748MB	2.30MB

CONCLUSIONS

- No conclusions yet .....