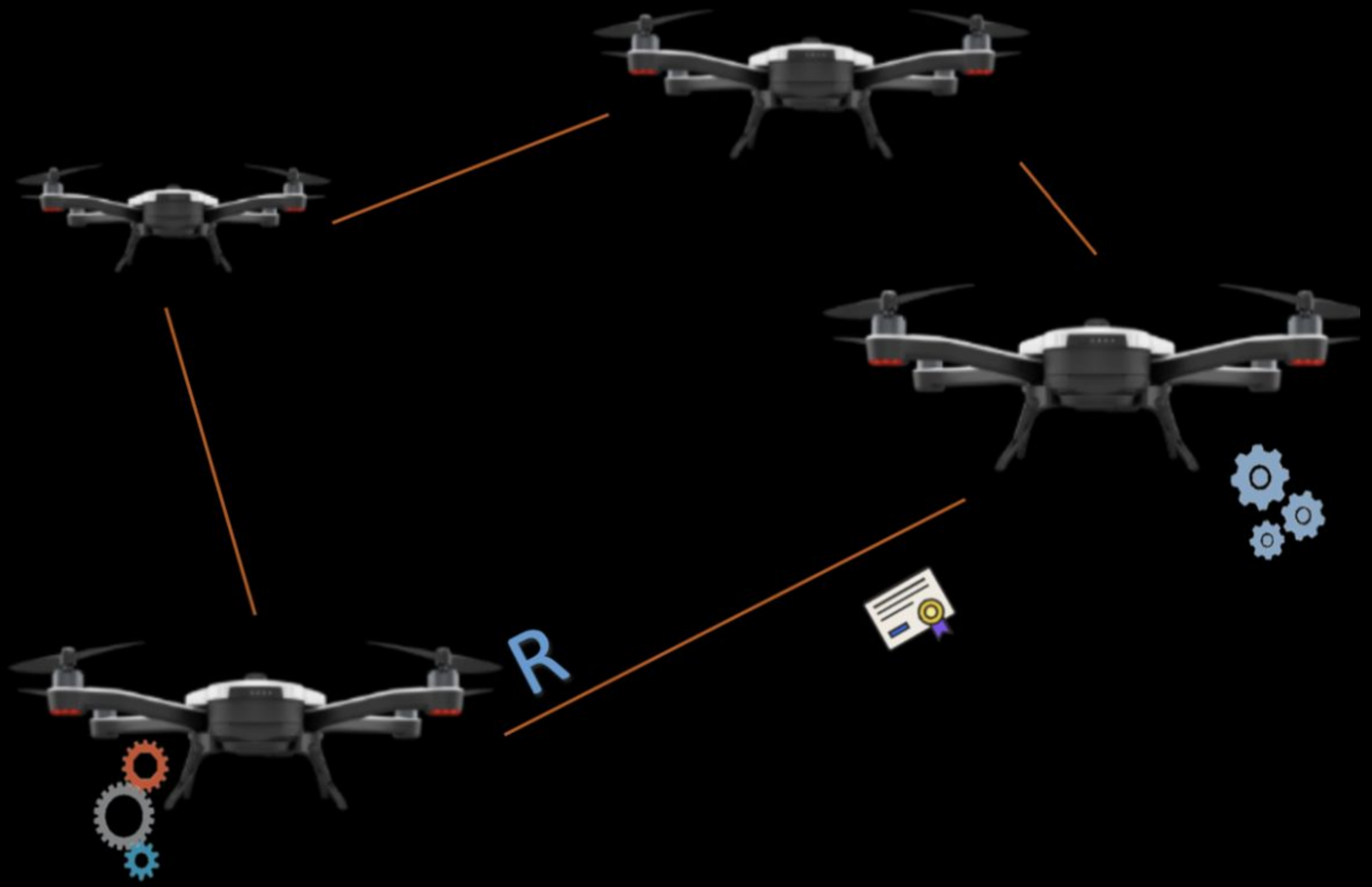
DIAT: Data Integrity Attestation for Resilient Collaboration of Autonomous Systems

Tigist Abera (Technische Universität Darmstadt), Raad Bahmani (Technische Universität Darmstadt), Ferdinand Brasser (Technische Universität Darmstadt), Ahmad Ibrahim (Technische Universität Darmstadt), Ahmad-Reza Sadeghi (Technische Universität Darmstadt), Matthias Schunter (Intel Labs)

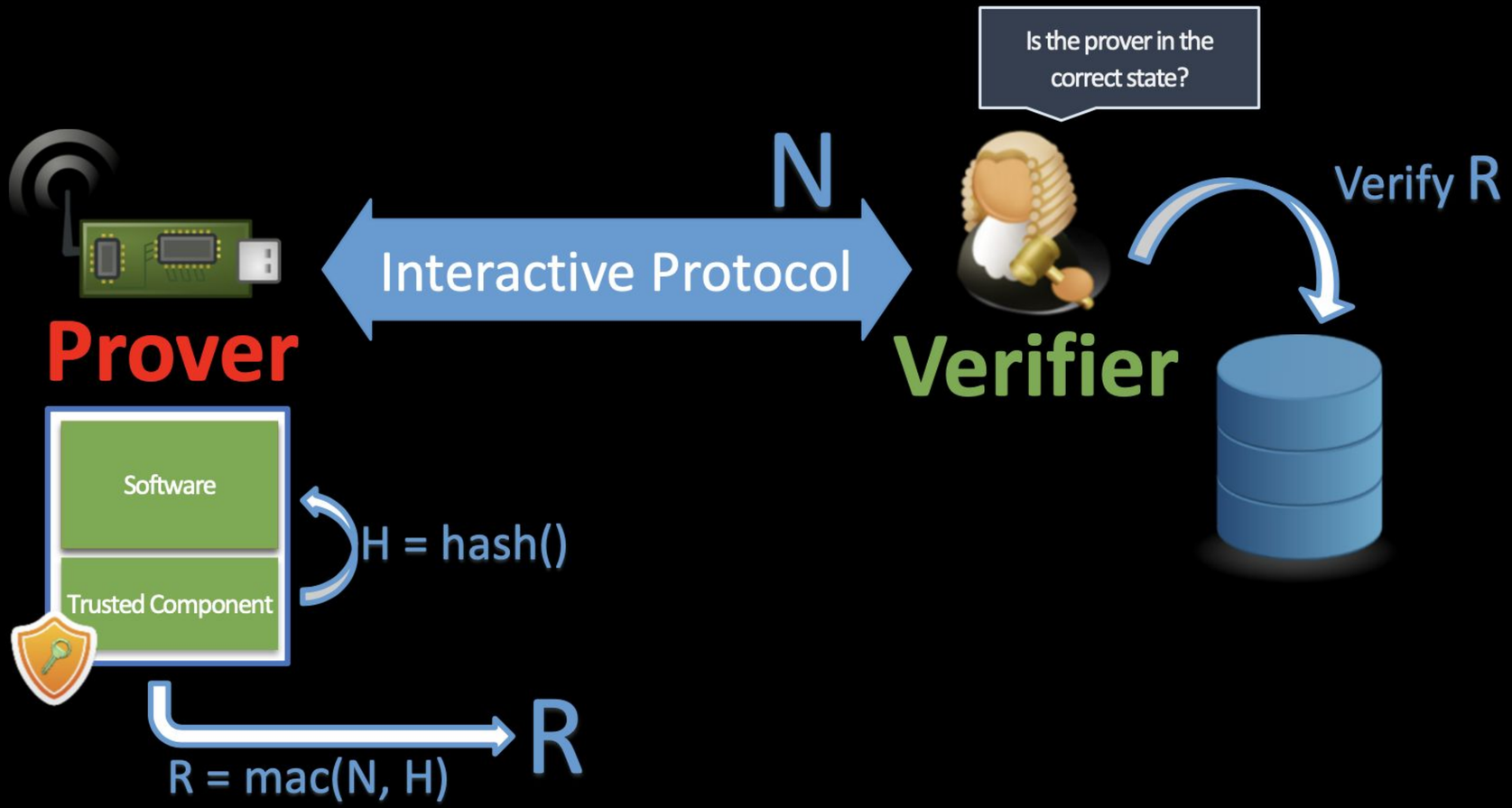
Presented by: Yuan Gao
11/08/2022

Acknowledgments: slide material derived from authors.

Introduction


R

Remote Attestation



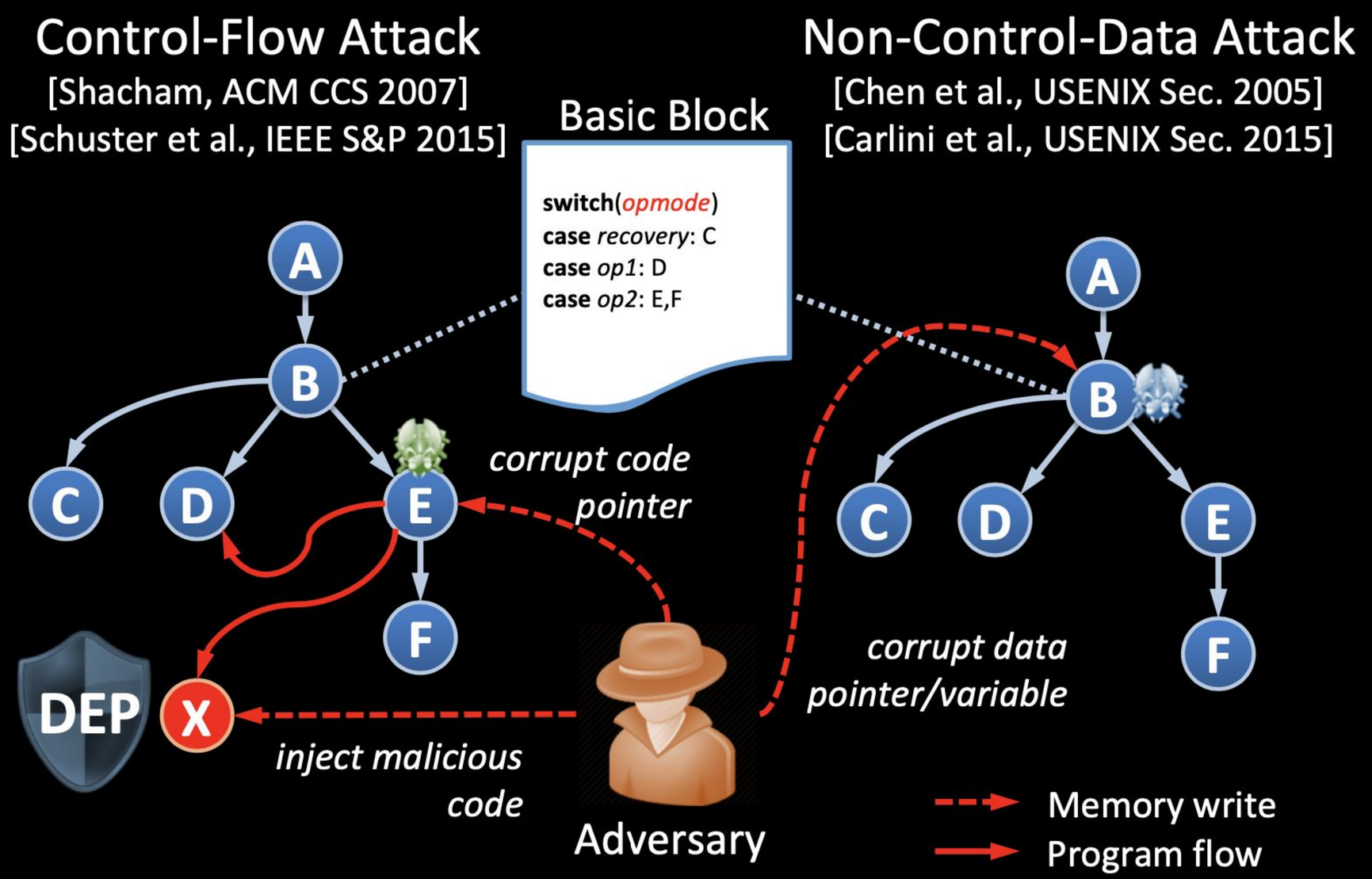
N Random Nonce

R Attestation Response

Key Limitation:

Static attestation schemes do not address runtime attacks

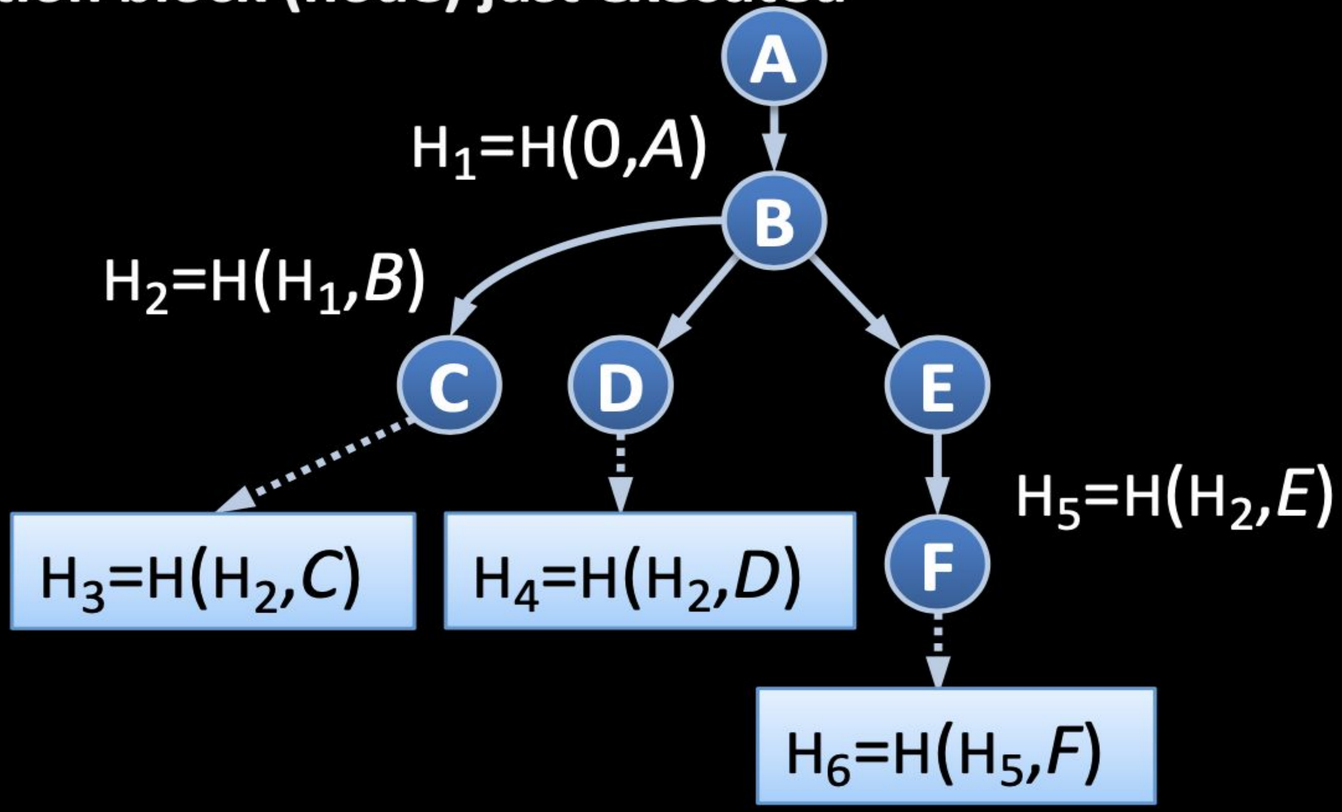
Problem Space of Runtime Attacks



Control-Flow Attestation

Cumulative Hash Value: $H_i = H(H_{i-1}, N)$

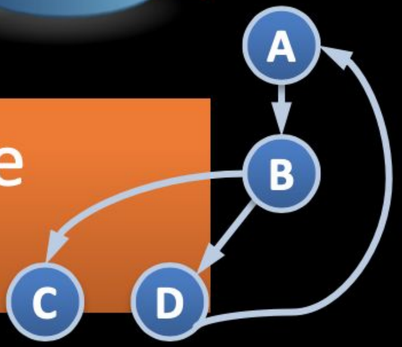
- H_{i-1} -- previous hash result
- N -- instruction block (node) just executed



Control-Flow Attestation

High overhead on the verifier

Program complexity leads to a large number of valid hashes

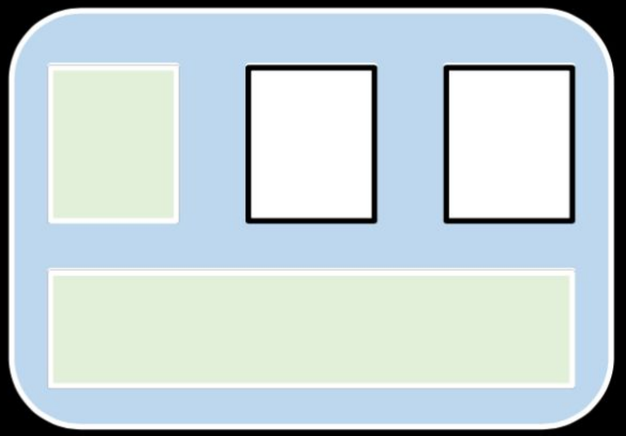


Only applicable to small programs

How to apply Control-flow attestation for autonomous systems ?

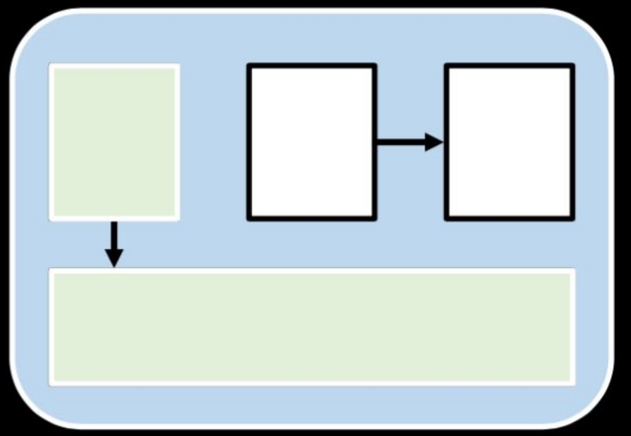
What? When? How?

High Level Idea



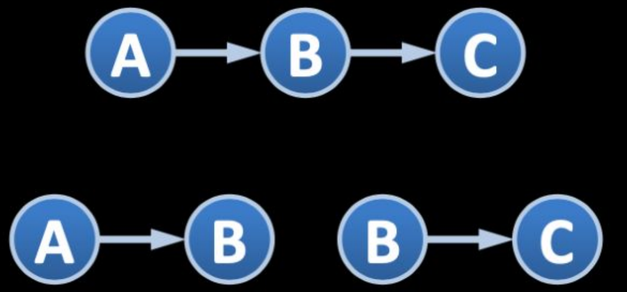
Modularization

Software is divided into smaller isolated modules



Data-flow attestation

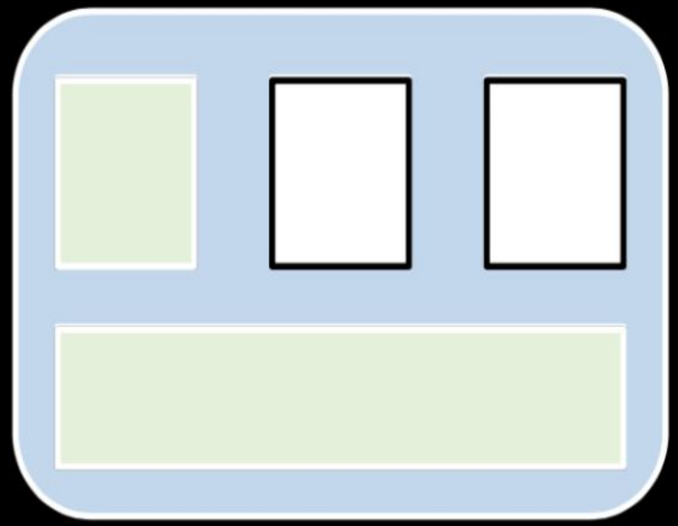
Attestation is executed when data is exchanged



Exec path representation

Execution path is represented as a multiset of edges

Assumptions



Modularization

Software is divided into smaller isolated modules

Modular software

- can be decomposed into simple interacting modules

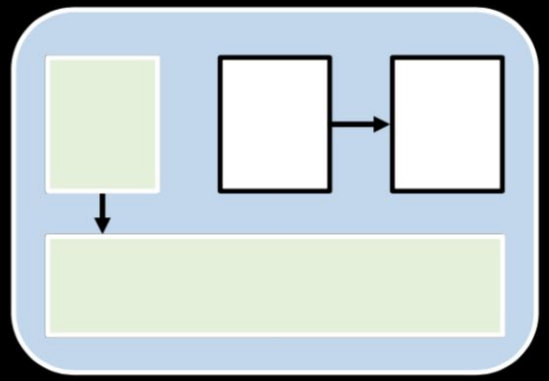
Data-flow monitoring

- Software modules interact through a well-defined communication channels

Isolation Architecture

- Software modules are securely isolated for each other

Data-flow monitoring

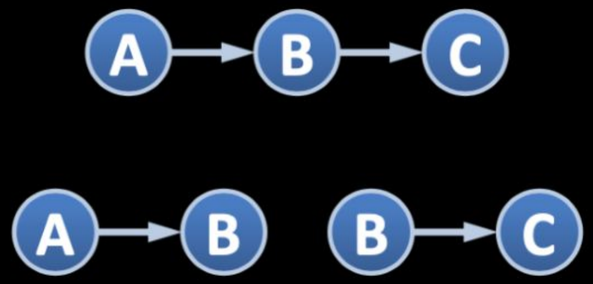


DFMonitor

Data-flow attestation

Attestation is executed
when data is exchanged

Control-flow monitoring

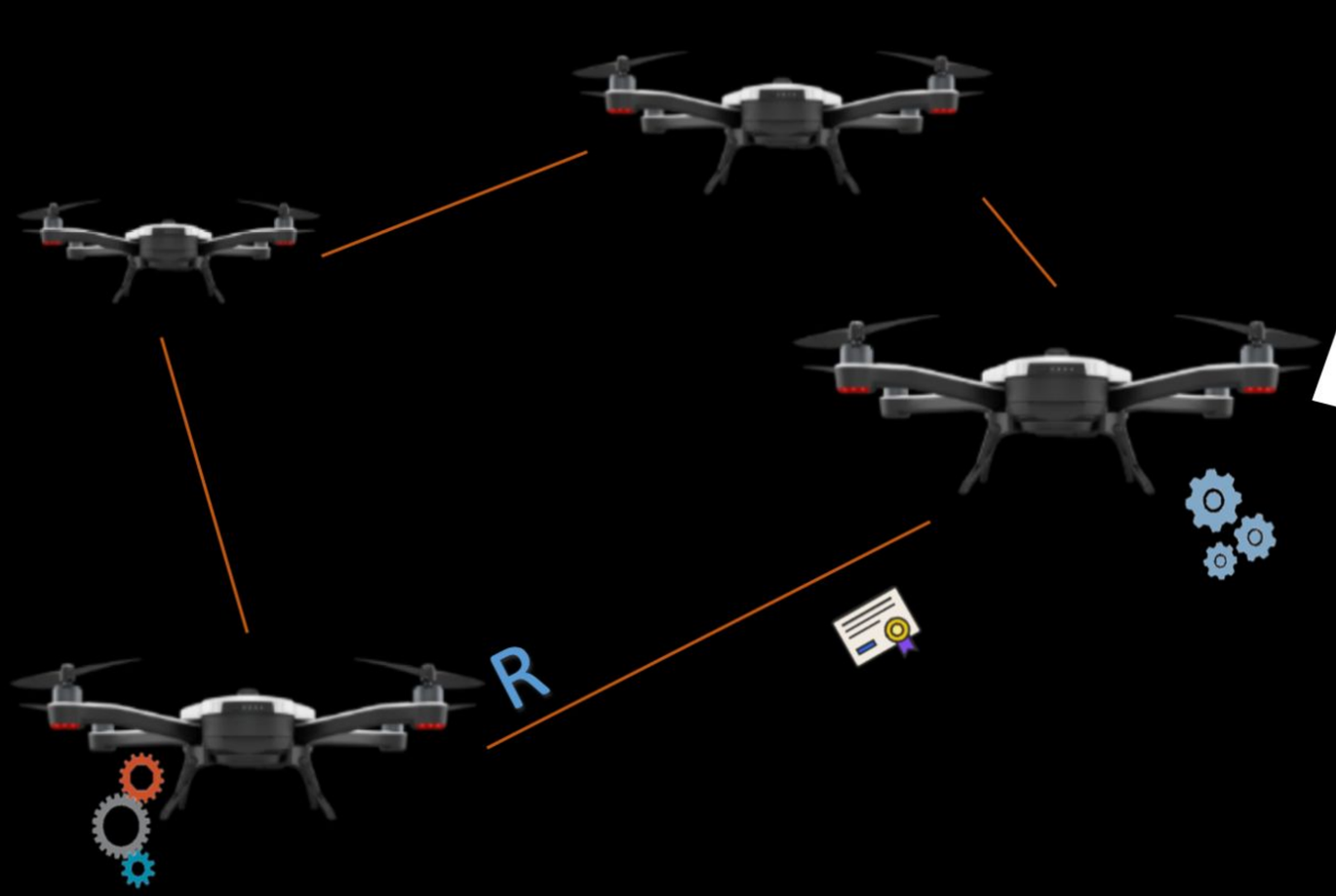


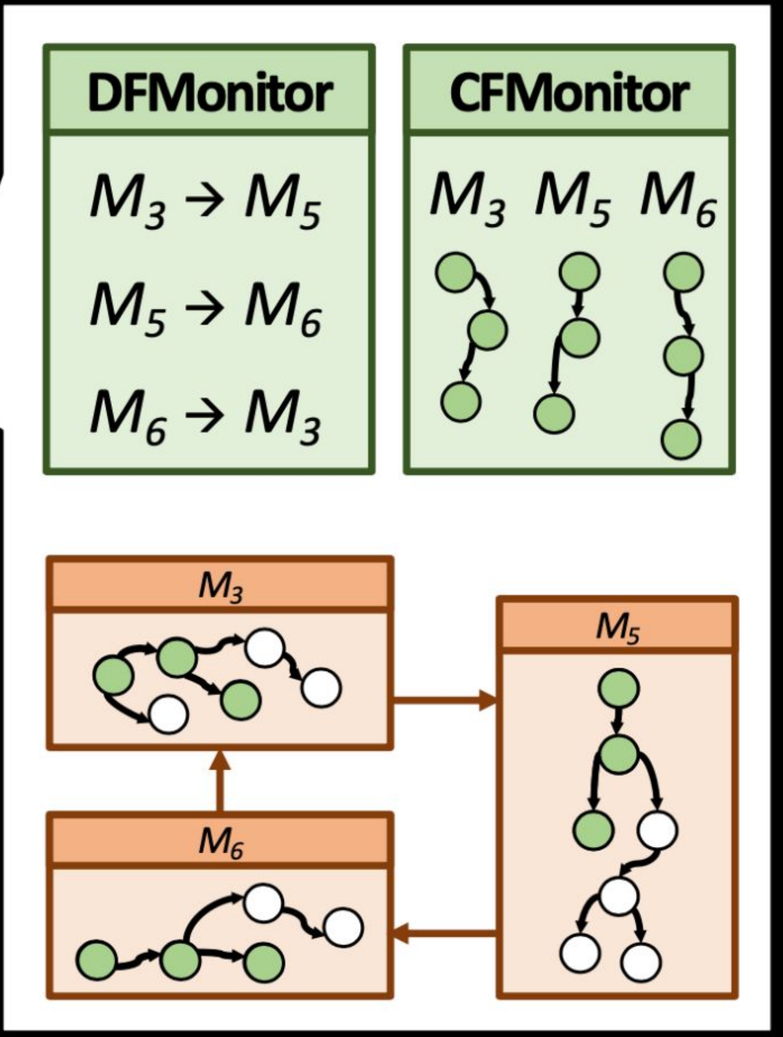
CFMonitor

Exec path representation

Execution path is represented
as a multiset of edges

High Level Idea





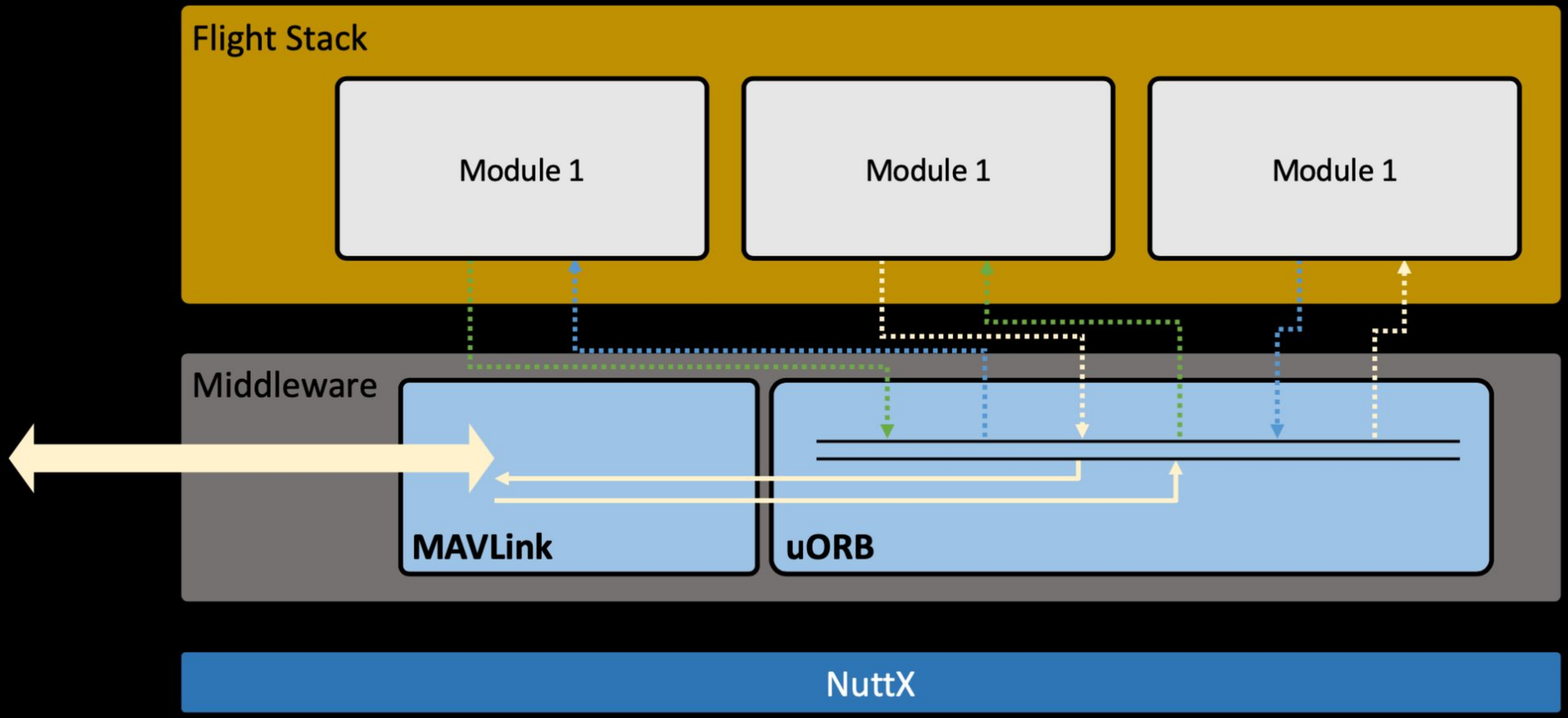
Implementation

Autonomous Drone

Pixhawk: open-hardware project autopilot hardware

PX4: open source flight control software for drones

PX4 autopilot Architecture



DFMonitor

Objective

Observes data flow between software modules and identify critical ones

Realization

Extending Middleware to enable data-flow monitoring functionalities

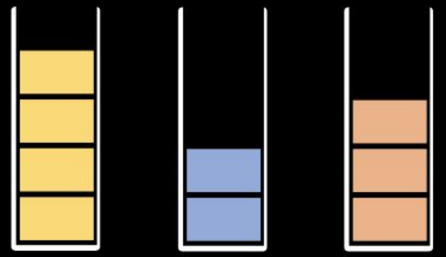
Functionalities:

- Extending MAVLink message format to include attestation requests/response
- Extending uORB to record message subscription and data generation
- Flushing uORB data buffers before when sensitive data is requested

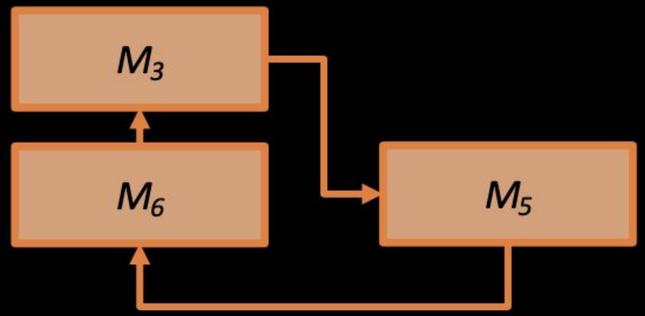
DFMonitor

Extending MAVLink message format

Flushing uORB data buffers



Observing data flow between modules



CFMonitor

Objective

Observes execution of critical modules and records their control flow

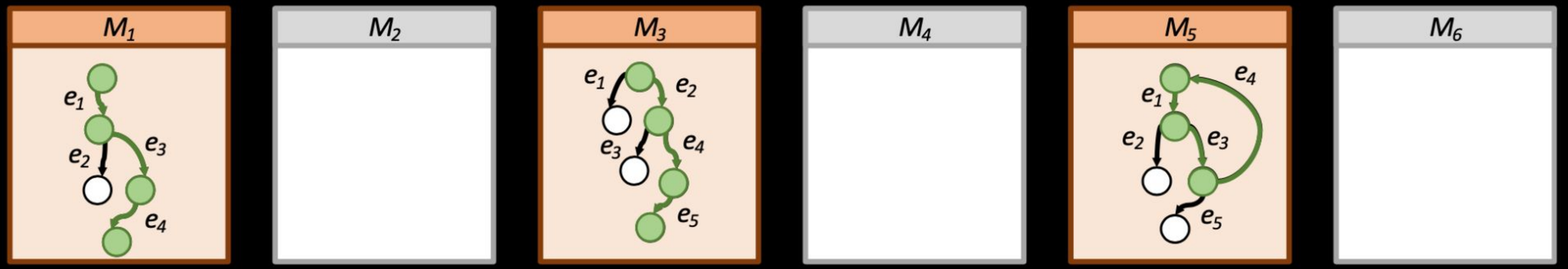
Realization

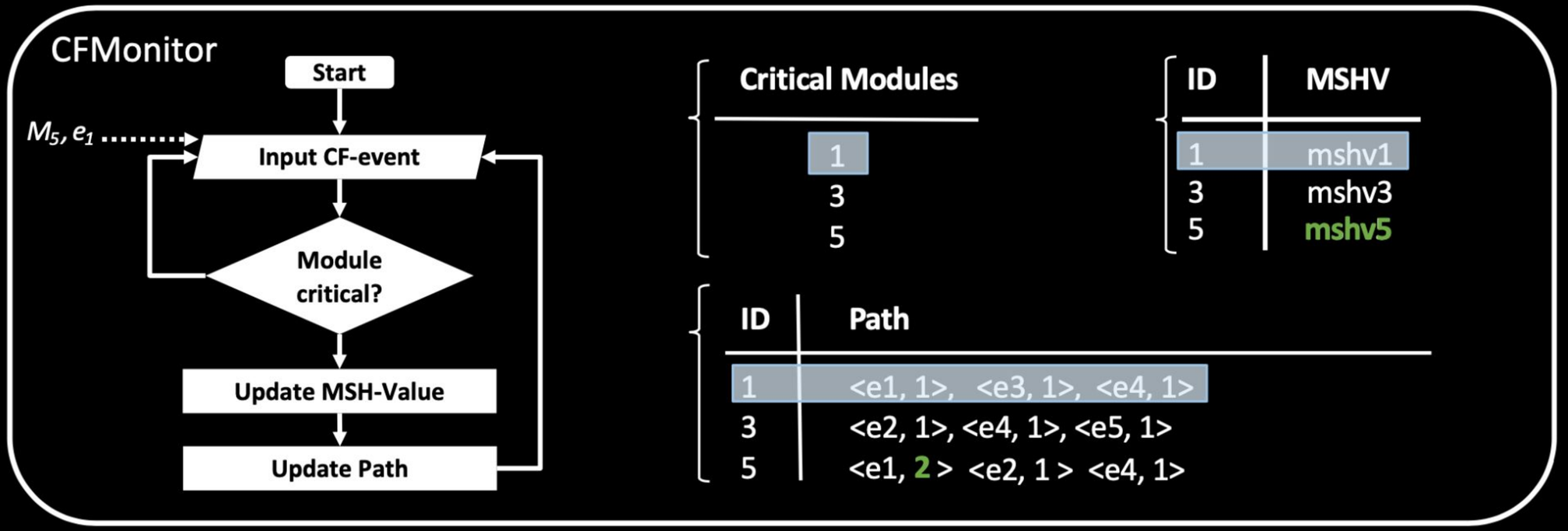
Instrumenting software modules with instructions that allow recording its control flow

Functionalities:

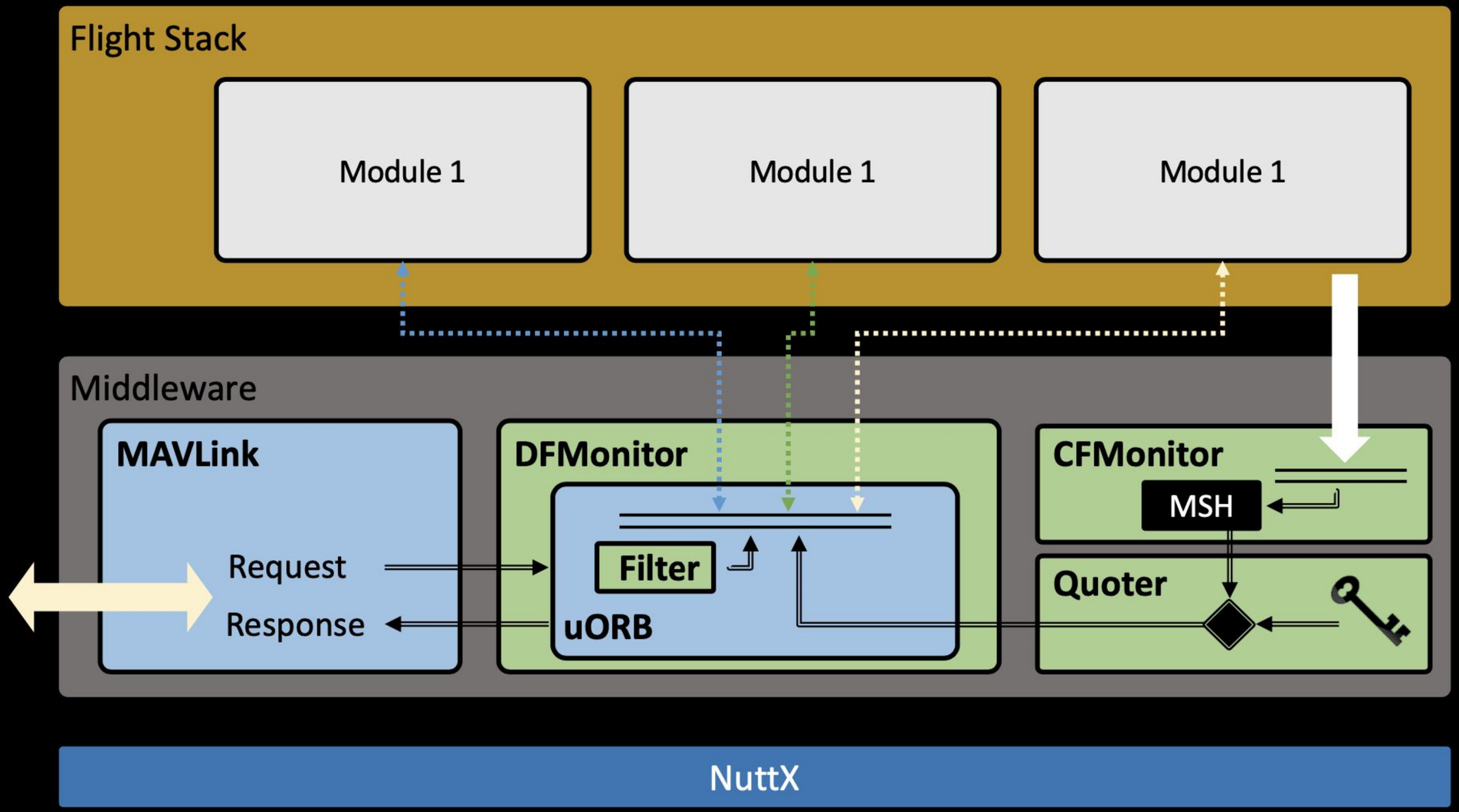
- Logic for recording the control flow events of critical modules
- Instrumentation instruction which call the logic at every control-flow event

CFMonitor


M_1
e_1
e_2
e_3
e_4
M_2
M_3
e_1
e_2
e_3
e_4
e_5
M_4
M_5
e_1
e_2
e_3
e_4
e_5
M_6


CFMonitor
Start
M_5, e_1
Input CF-event
Module critical?
Update MSH-Value
Update Path
Critical Modules
1
3
5
ID
MSHV
1
mshv1
3
mshv3
5
mshv5
ID
Path
1
<e1, 1>, <e3, 1>, <e4, 1>
3
<e2, 1>, <e4, 1>, <e5, 1>
5
<e1, 2 > <e2, 1 > <e4, 1>

Integration Into PX4



Evaluation

Module	CFG	Exe Path	Attestation (ms)	Verification(ms)
GPS	2922	22249	835	849
Gyroscope	912	20004	748	760
e-Compass	1468	18907	716	718
IMU Sensor	1905	158671	6341	6357
Pressure Sensor	1051	1150	46	46
FMU	1828	38132	1510	1511
PX4IO	3661	12723	484	489
LED Driver	532	32	1	1
STM32 ADC	251	21274	805	808
Commander	7852	9418	354	365
Load Monitor	135	8	0.3	0.4
Sensors	2032	40410	1618	1623
Systemlib	2555	662142	26341	26365
Total	27014	1005120	39799, 3	39892, 4

GPS coordinates involves 1 of 13 executing modules

Modularity entails an improvement of 95% on runtime

Different Data Types

Data		*cmd_state*	*battery_status*	*sensor_acel*	*sensor_gyro*
Count	Critical Modules	12	12	2	2
	Executed Modules	12	13	7	8
	Percentage	100%	92%	28%	25%
$\sum$ *of* CFGs	Critical Modules	197823	46860778	194	250
	Executed Modules	197823	46862156	1590	1328
	Percentage	100%	99%	12%	18%
$\sum$ *of* Executed Paths	Critical Modules	26572	26572	3373	2817
	Executed Modules	26572	27104	13622	13873
	Percentage	100%	98%	24%	20%

Security Considerations

DFMonitor:

- **All critical modules** will be detected and attested

CFMonitor:

- **Adding edges** not in CFG will be detected
- **Adding edges** in CFG to execution path requires security policy
- **Reordering edges** in the execution path cannot be detected

Conclusion

- Static attestation cannot detect runtime attacks
- Control-flow attestation (CFA) is too complex
- DIAT allows CFA in the autonomous settings. **However, this requires**
  - Modular software design with clear communication
  - Strong isolation between software modules

References

- Paper
- Presentation

Q&A

Discussions

- How do you handle asynchronous events like interrupt that change the control flow like traps, break point? Improve their system?
- They mentioned they are working on a hardware solution for the previous question. If you are working on this, what idea you have to solve this problem?