DIAT:

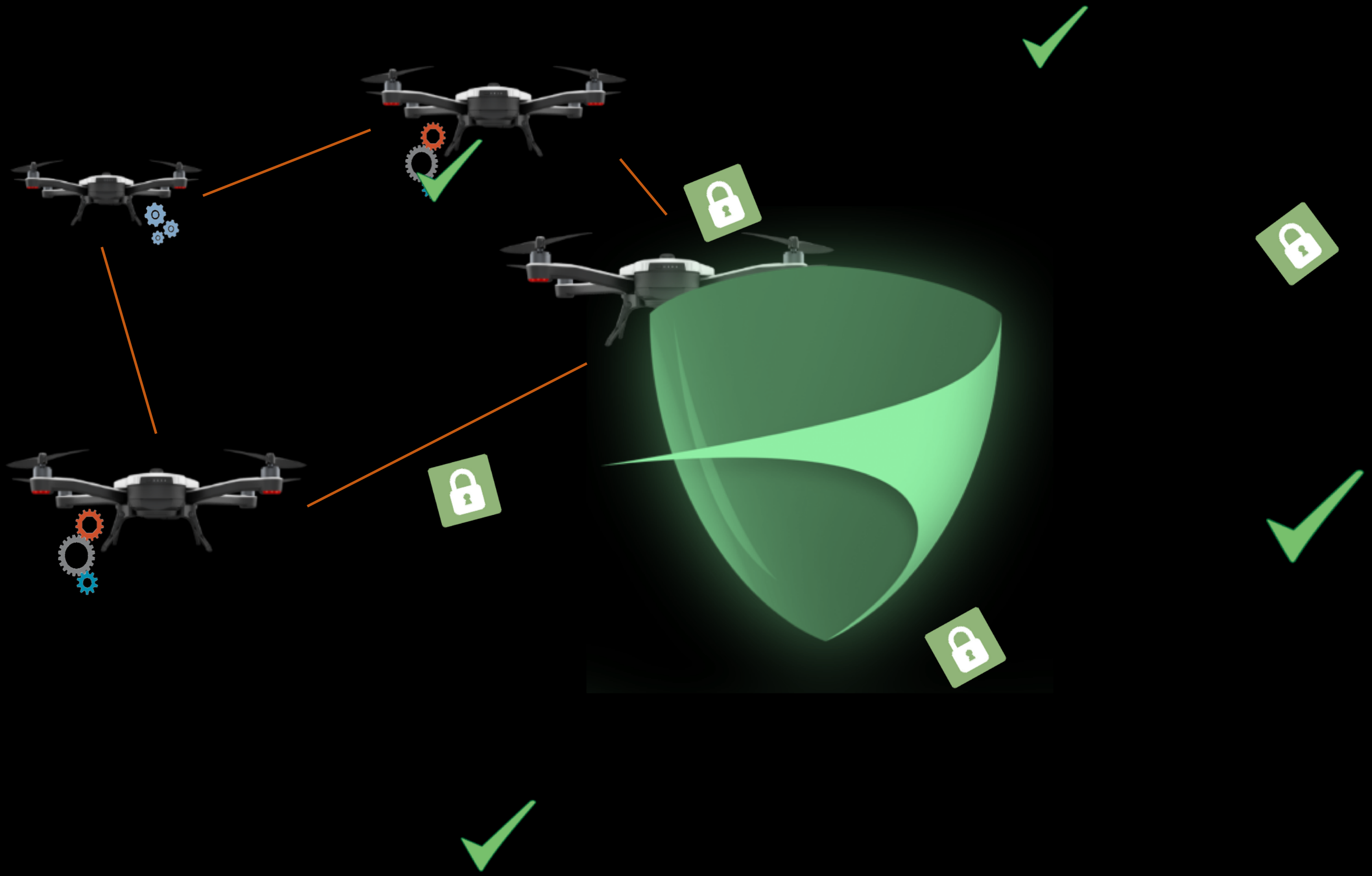
Data Integrity Attestation for Resilient Collaboration of Autonomous Systems

Tigist Abera, Raad Bahmani, Ferdinand Brasser,
Ahmad Ibrahim,
Ahmad-Reza Sadeghi, and Matthias Schunter

Technische Universität Darmstadt, Germany and
Intel Labs, Portland, OR, U.S.A.

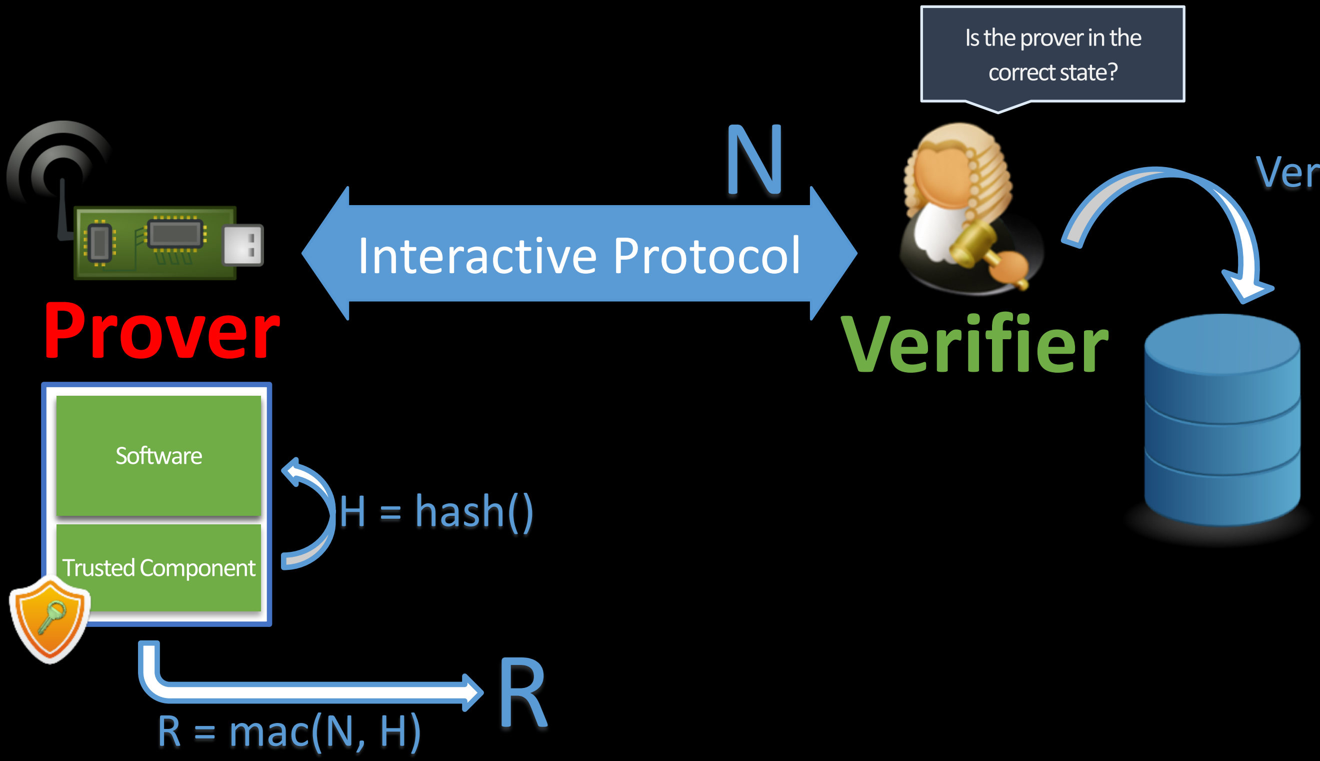
Motivation

Goal



Remote attestation checks trustworthiness of a remote (embedded) device

Remote Attestation



Is the prover in the correct state?

N

Interactive Protocol

Prover

Verifier

Verify R

Software

Trusted Component

H = hash()

R = mac(N, H)

R

N	Random Nonce
R	Attestation Response

Key Limitation:

Static attestation schemes do not address runtime attacks

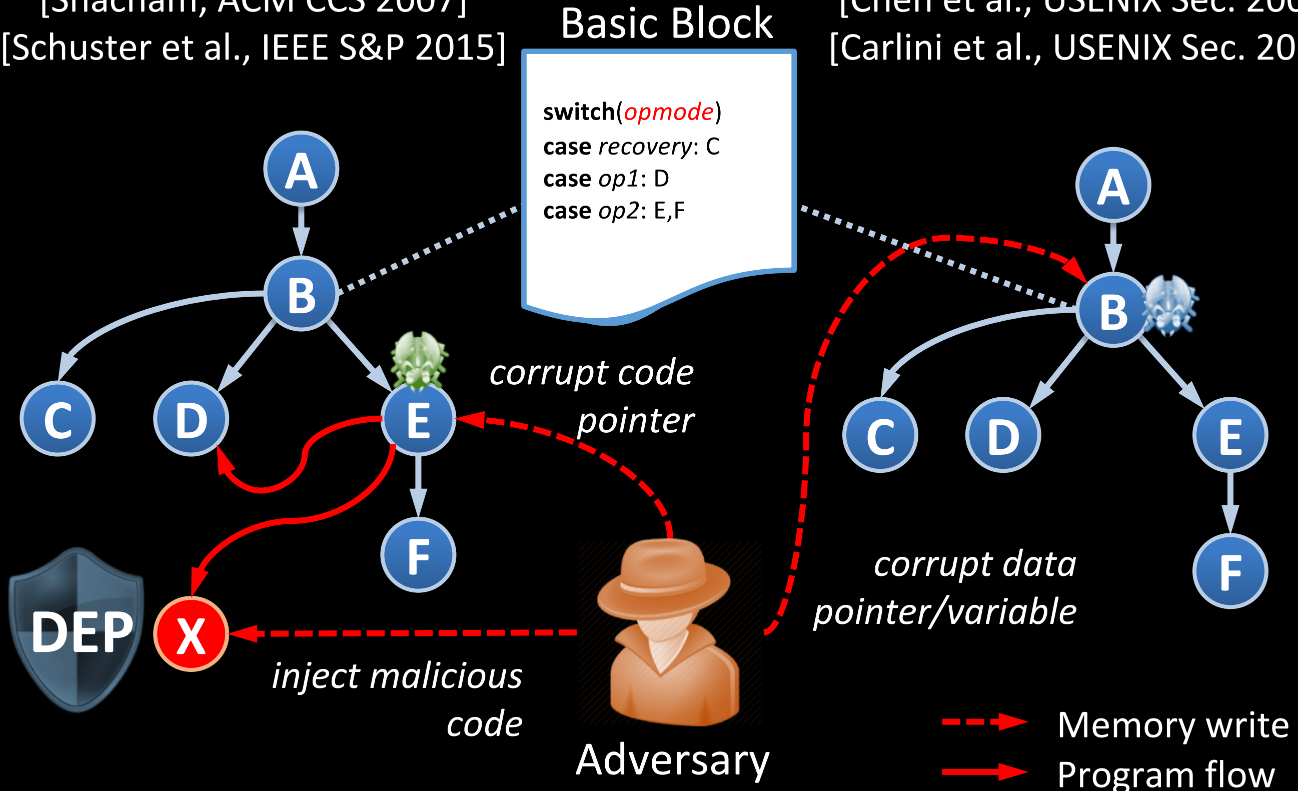
Problem Space of Runtime Attacks

Control-Flow Attack

[Shacham, ACM CCS 2007]
[Schuster et al., IEEE S&P 2015]

Non-Control-Data Attack

[Chen et al., USENIX Sec. 2005]
[Carlini et al., USENIX Sec. 2015]



Basic Block

```
switch(opmode)
case recovery: C
case op1: D
case op2: E,F
```

corrupt code pointer

inject malicious code

corrupt data pointer/variable

Adversary

Memory write

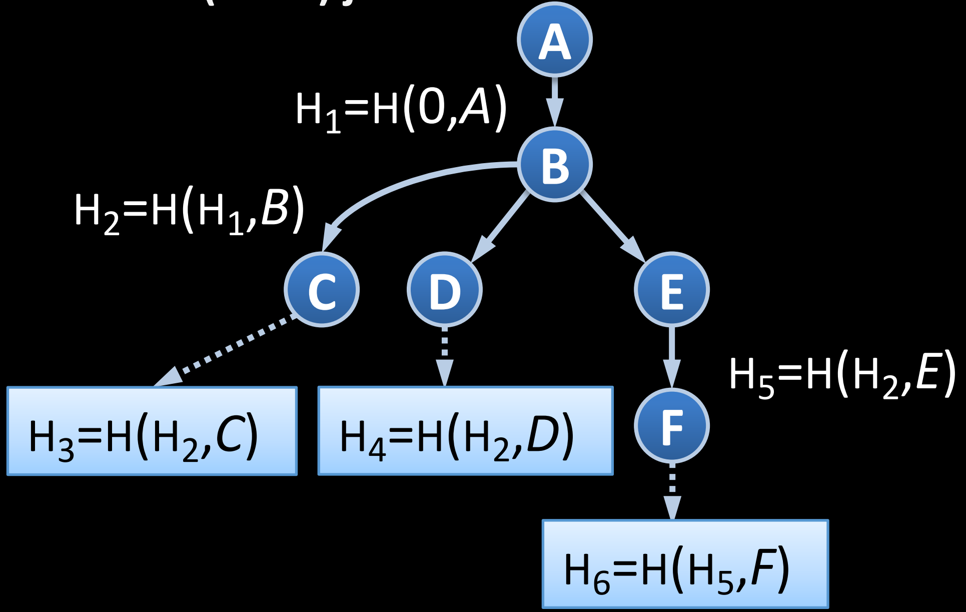
Program flow

Control-flow attestation aims at the detection 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

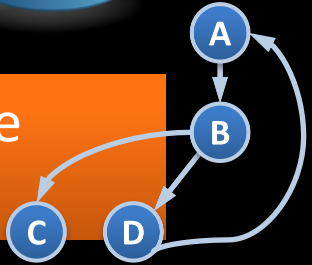


Problems

Control-Flow Attestation

High overhead on the verifier

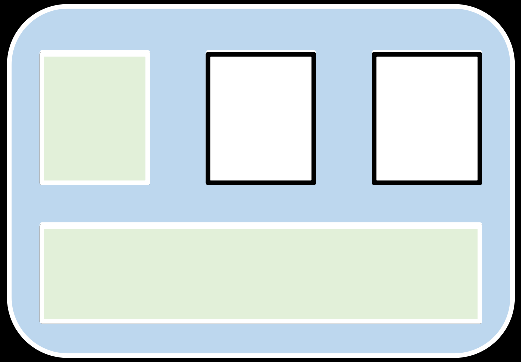
Program complexity leads to a large number of valid hashes



Only applicable to small programs

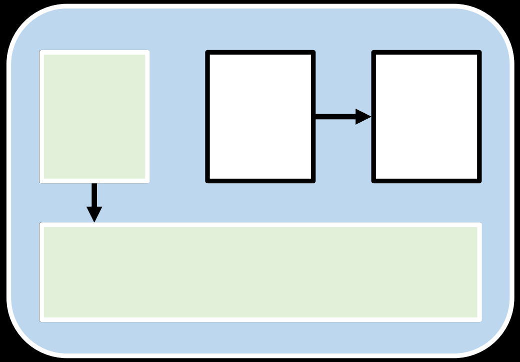
Control-flow attestation for autonomous systems

High Level Idea



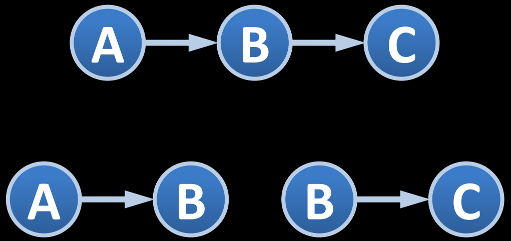
Modularization

Software is divided into smaller isolated modules



Data-flow attestation

Attestation is executed when data is exchanged



Exec path representation

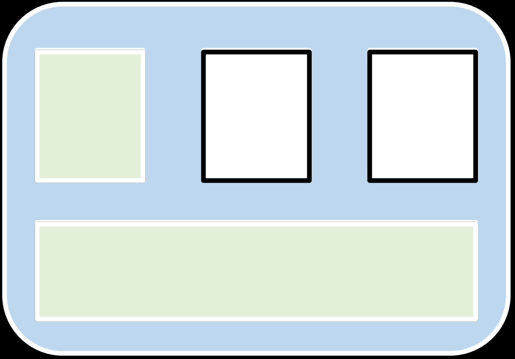
Execution path is represented as a multiset of edges

Assumptions

Modular software
can be decomposed into simple interacting modules

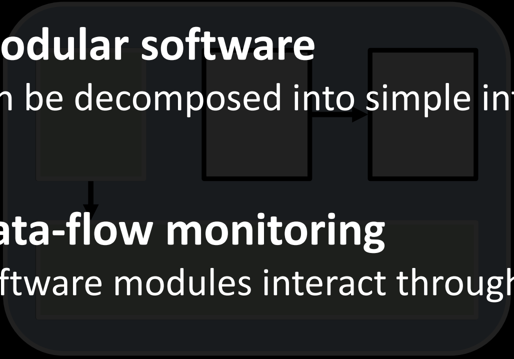
Data-flow monitoring
Software modules interact through a well-defined communication channels

Isolation Architecture
Software modules are securely isolated for each other



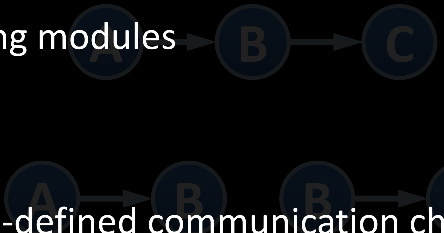
Modularization

Software is divided into smaller isolated modules



Data-flow attestation

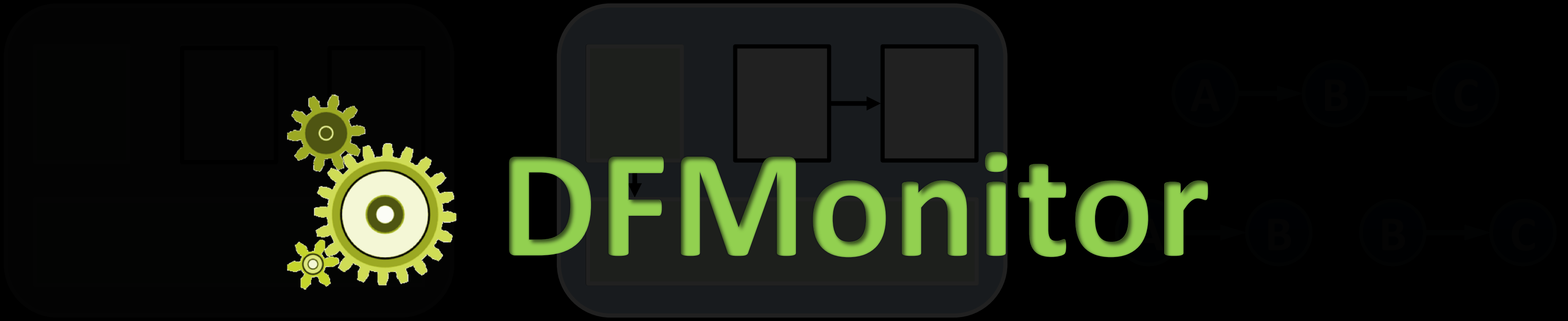
Attestation is executed when data is exchanged



Exec path representation

Execution path is represented as a multiset of edges

Data-Flow Monitoring



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

Control-Flow Monitoring

CFMonitor
A
B
C
A
B
B
C

Modularization

Software is divided into smaller isolated modules

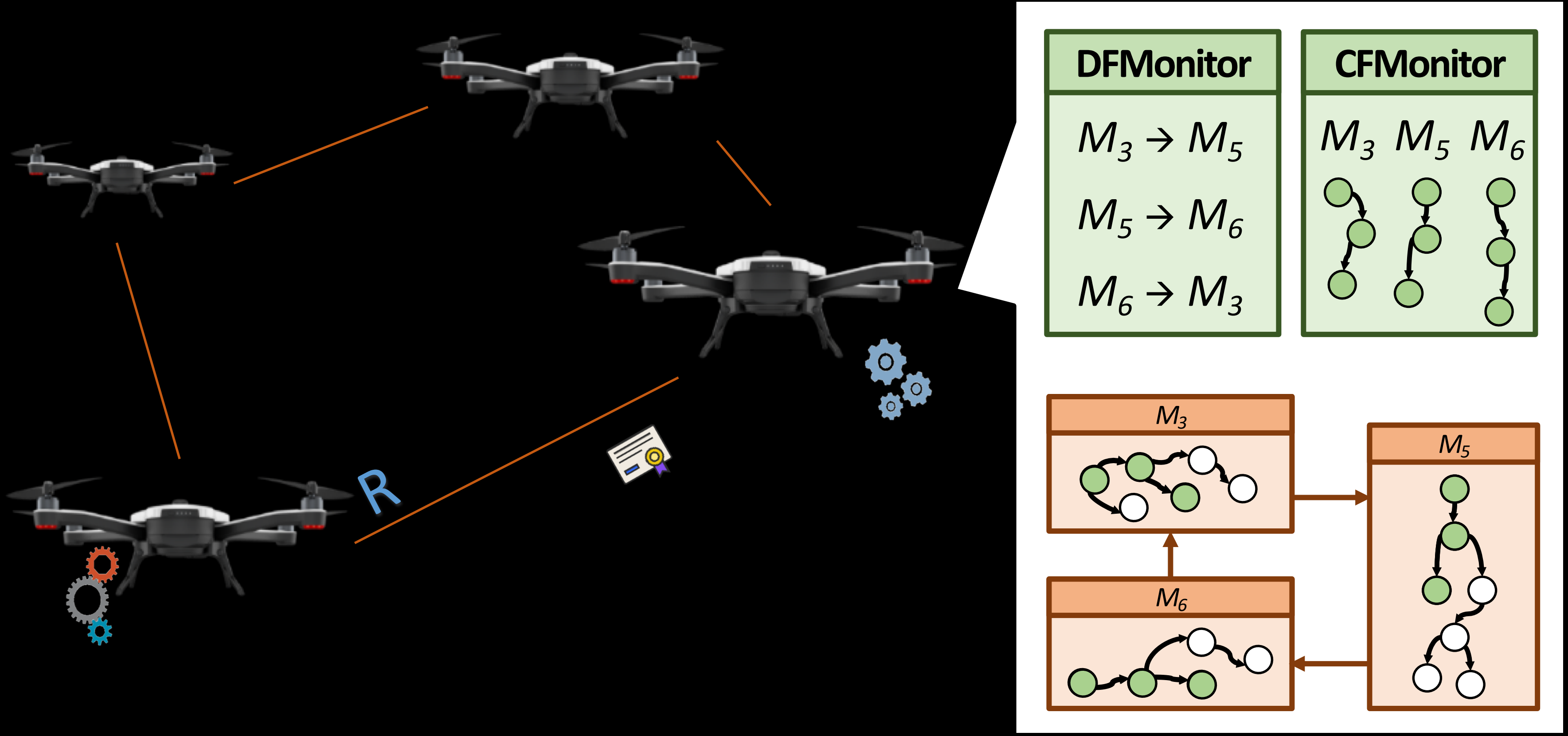
Data-flow attestation

Attestation is executed when data is exchanged

Exec path representation

Execution path is represented as a multiset of edges

High Level Idea



DFMonitor

$M_3 \rightarrow M_5$

$M_5 \rightarrow M_6$

$M_6 \rightarrow M_3$

CFMonitor

M_3 M_5 M_6

M_3

M_5

M_6

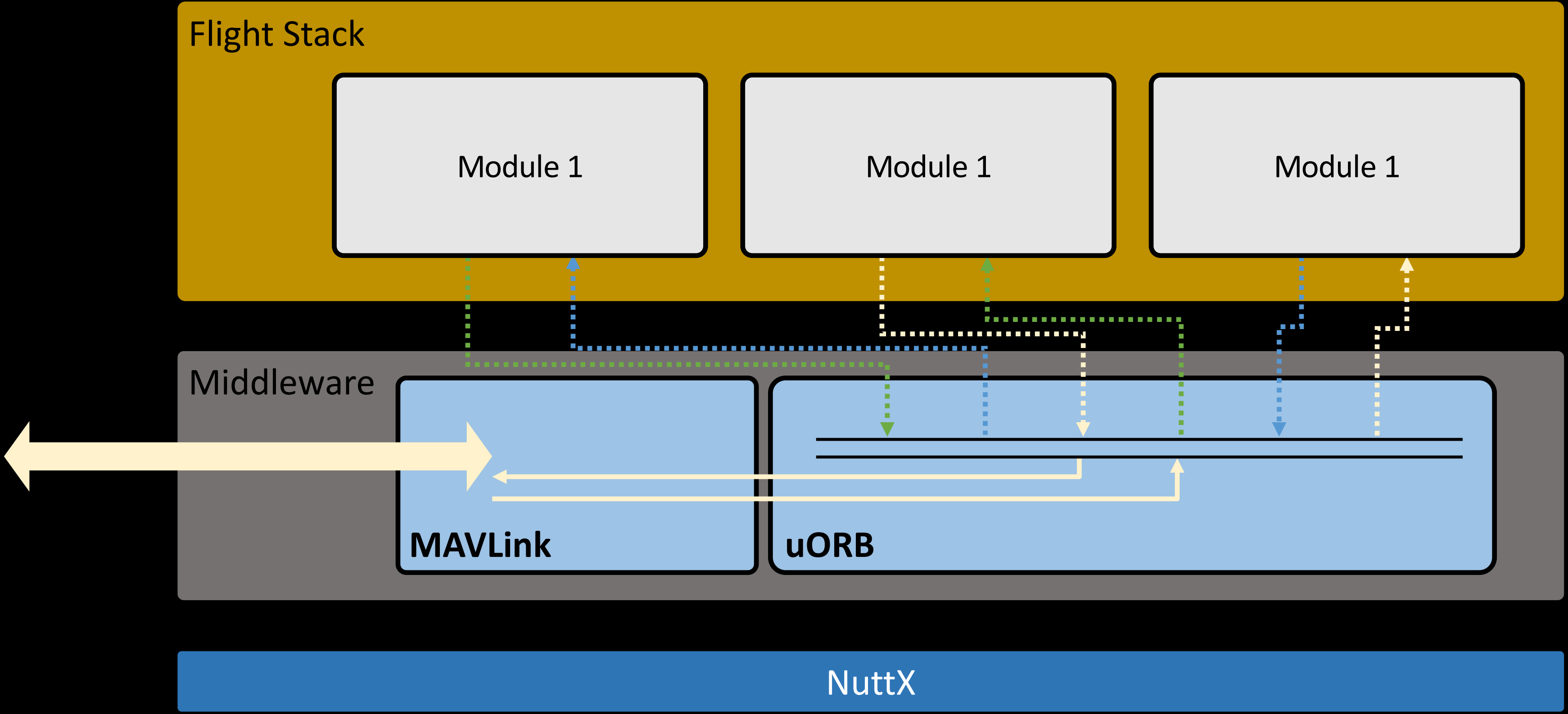
Implementation

Autonomous Drones

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

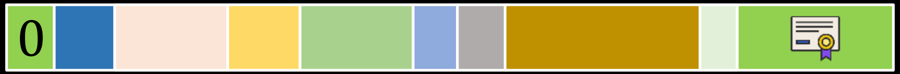
Extending Middleware to enable data-flow monitoring functionalities

Functionalities:

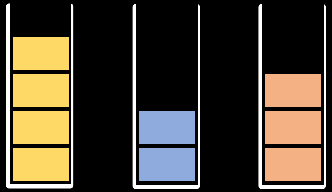
- Extending MAVLink message format to include attestation requests/response
- Extending uORB to record message subscription and data generation
- Flushing uORB data buffers before when sensitive data is requested

DFMonitor

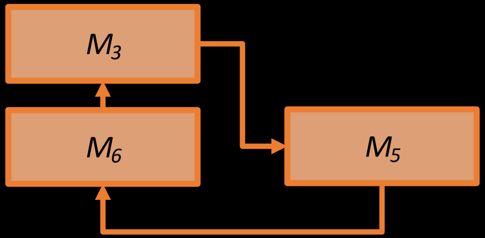
Extending MAVLink message format



Flushing uORB data buffers



Observing data flow between modules



CFMonitor

Objective

Observes execution of critical modules and records their control flow

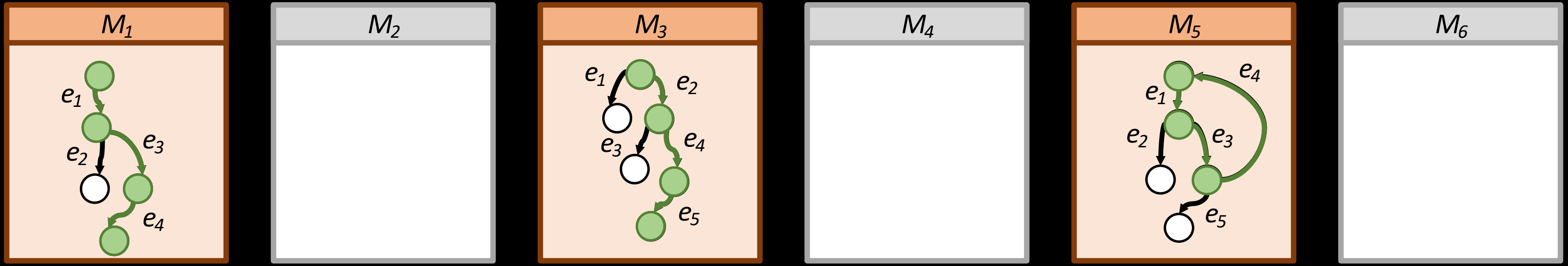
Realization

Instrumenting software modules with instructions that allow recording its control flow

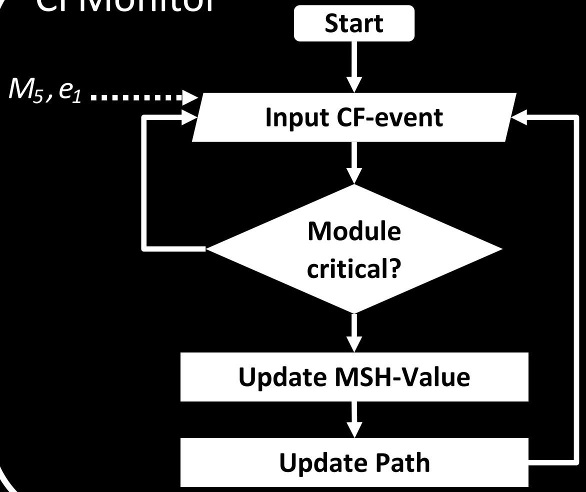
Functionalities:

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

CFMonitor



CFMonitor



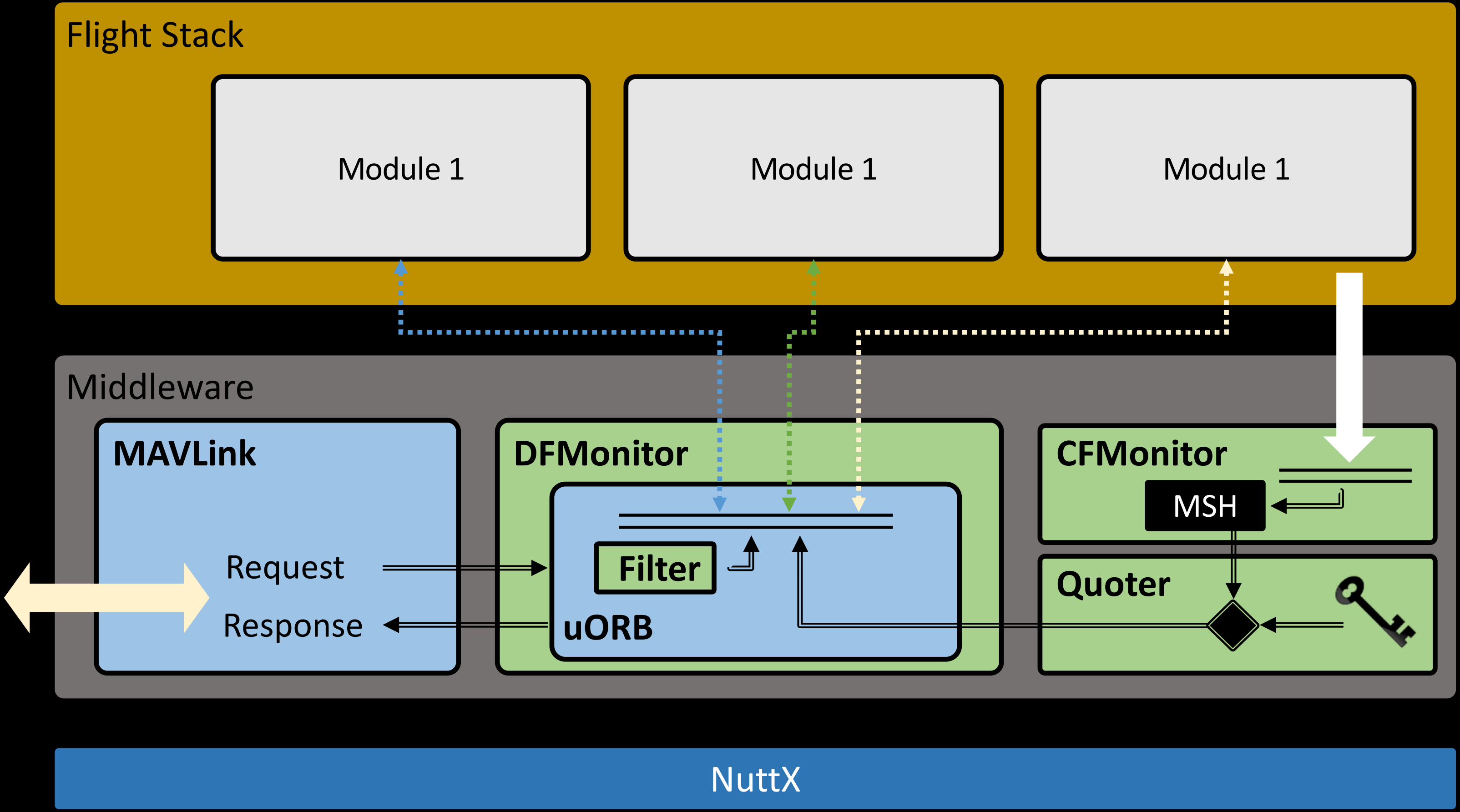
Critical Modules
1
3
5

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

Concept


Flight Stack
Module 1
Module 1
Module 1
Middleware
MAVLink
Request
Response
DFMonitor
Filter
uORB
CFMonitor
MSH
Quoter
NuttX

Evaluation

GPS Coordinates

MODULE	CFG SIZE	EXECUTION PATH	ATTESTATION TIME	VERIFICATION TIME
GPS	2922	22249	835	849
GYROSCOPE	912	20004	748	760
E-COMPASS	1468	18907	716	718
FMU	1828	38132	1510	1511
PX4IO	3661	12723	484	489
LE[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
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

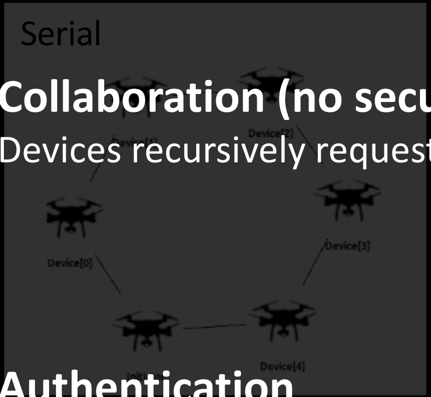
Modularity entails an improvement of 95% on runtime

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%

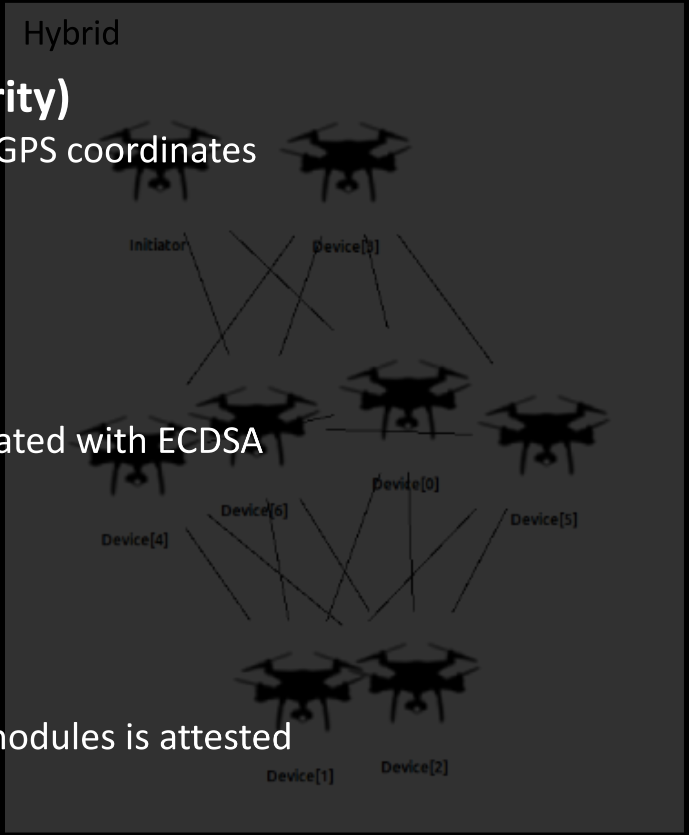
Scalability

Network Simulation



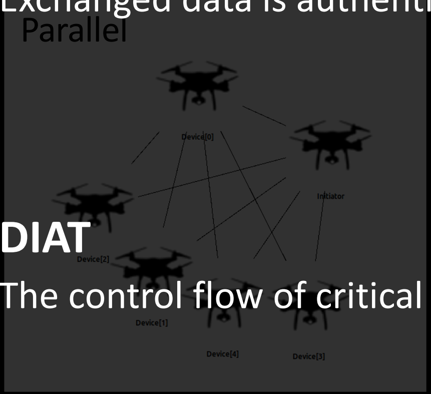
Collaboration (no security)

Devices recursively request GPS coordinates



Authentication

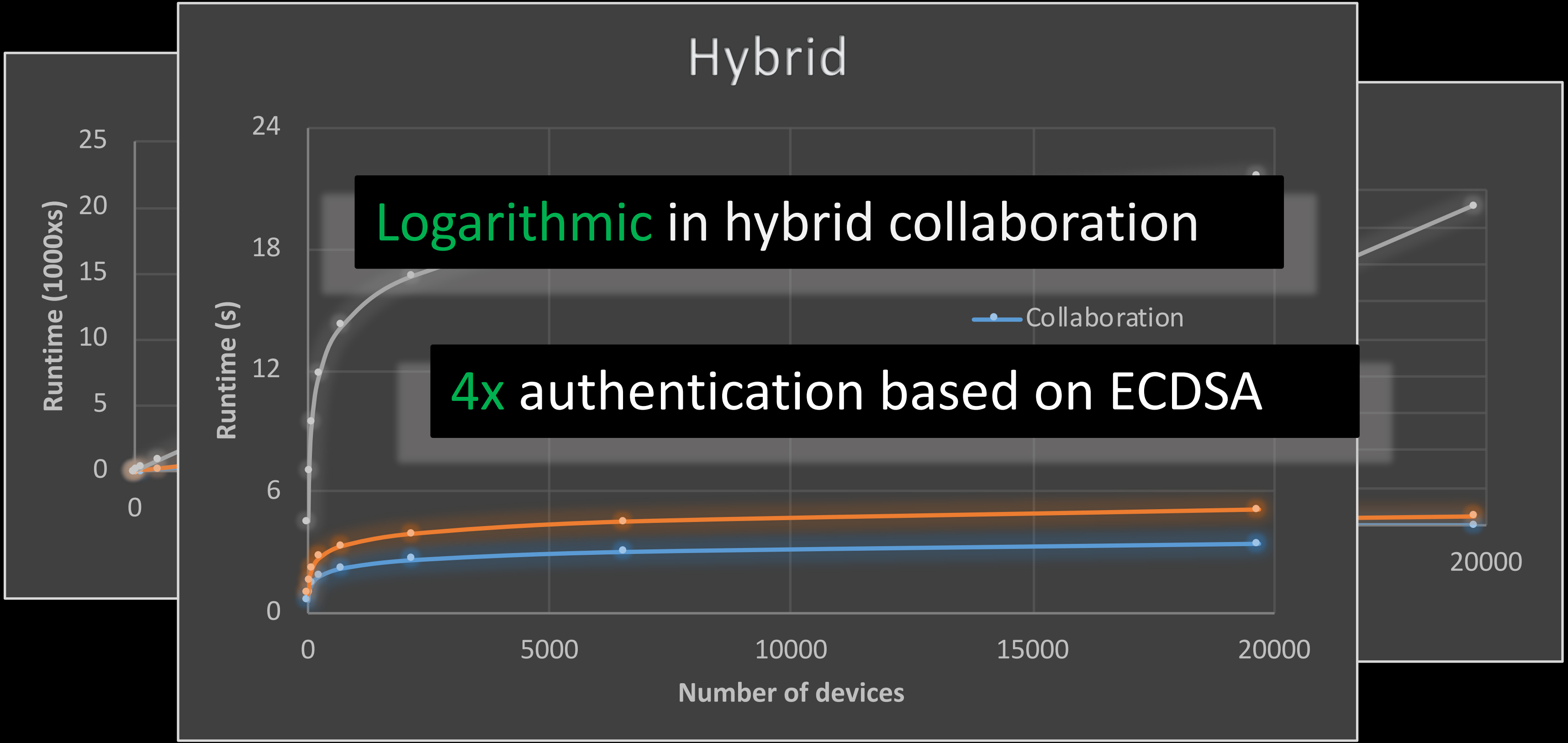
Exchanged data is authenticated with ECDSA



DIAT

The control flow of critical modules is attested

Runtime



Logarithmic in hybrid collaboration

4x authentication based on ECDSA

Security

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

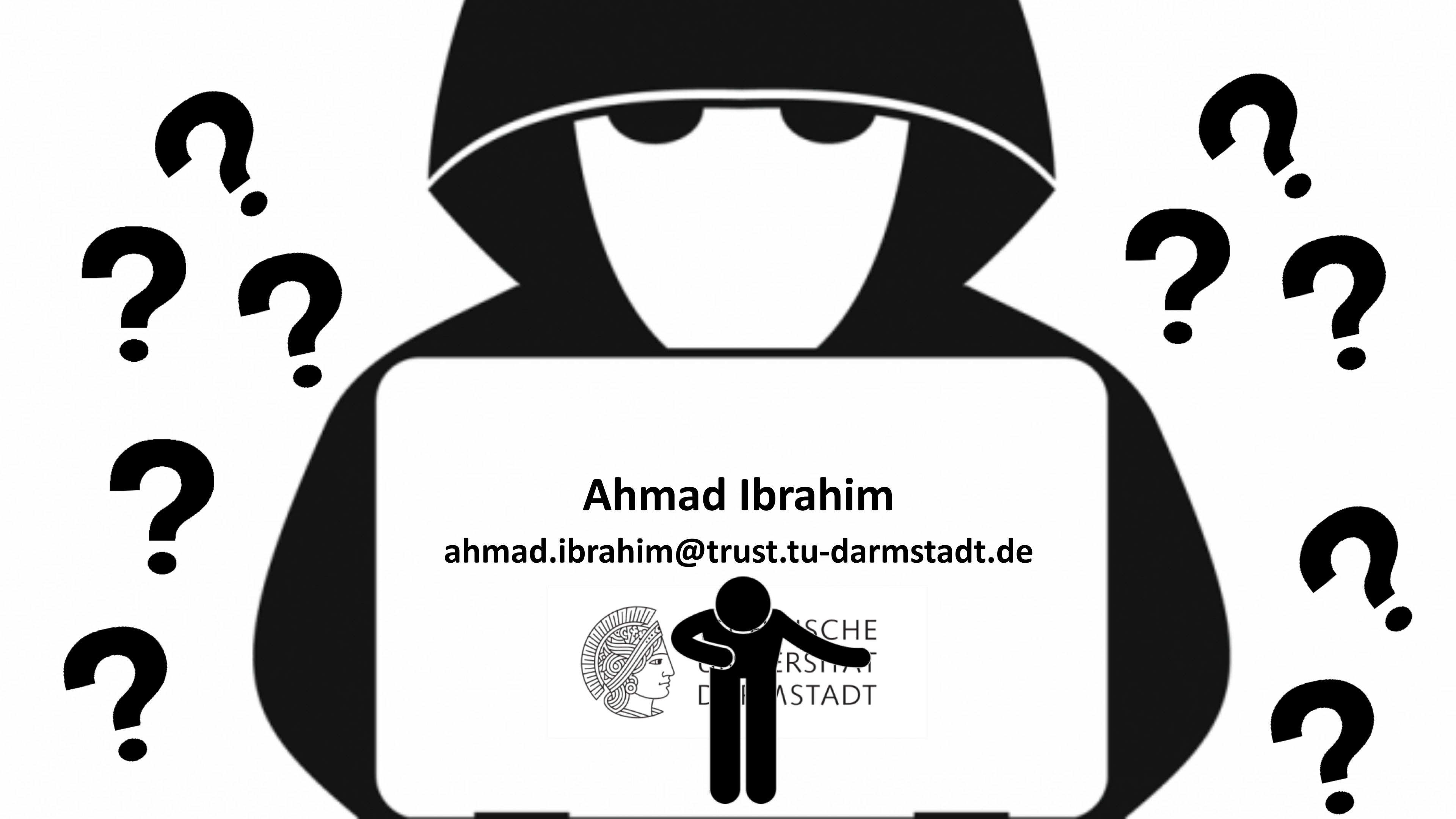
Conclusion

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



Ahmad Ibrahim

ahmad.ibrahim@trust.tu-darmstadt.de