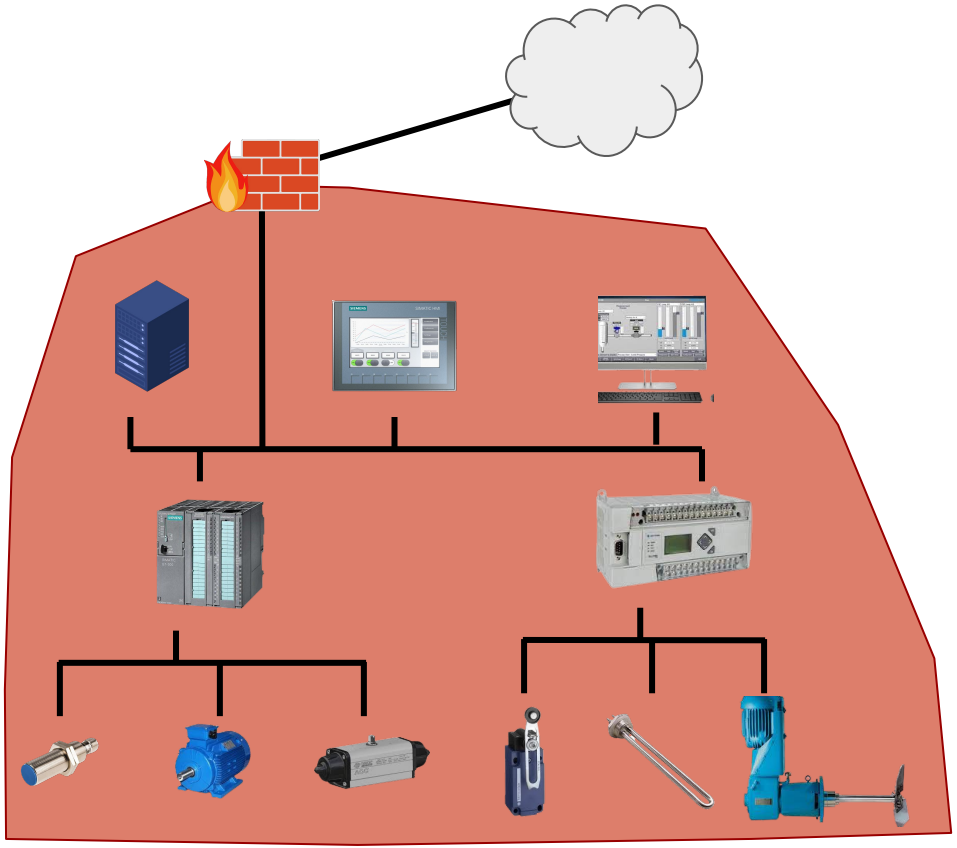
Hopper

Per-Device Nano Segmentation
for the industrial IoT

ETH zürich

Piet De Vaere, Andrea Tulimiero, Adrian Perrig

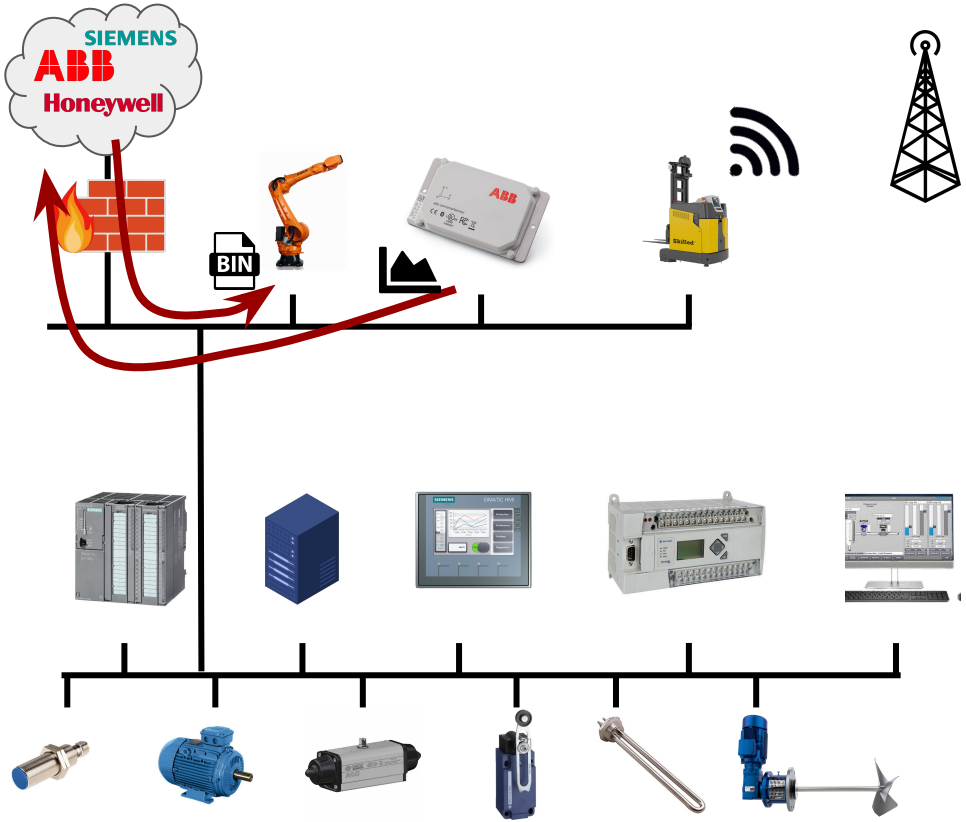
Industrial networks rely on perimeter protection



Traditional industrial networks are:

- Structured hierarchically
- Use specialized protocols
- Protected at the perimeter

New trends are challenging the perimeter defence



The network fabric is being unified

Devices are growing increasingly complex

Device-cloud communication

Wireless communication

Classical (micro) segmentation is limited

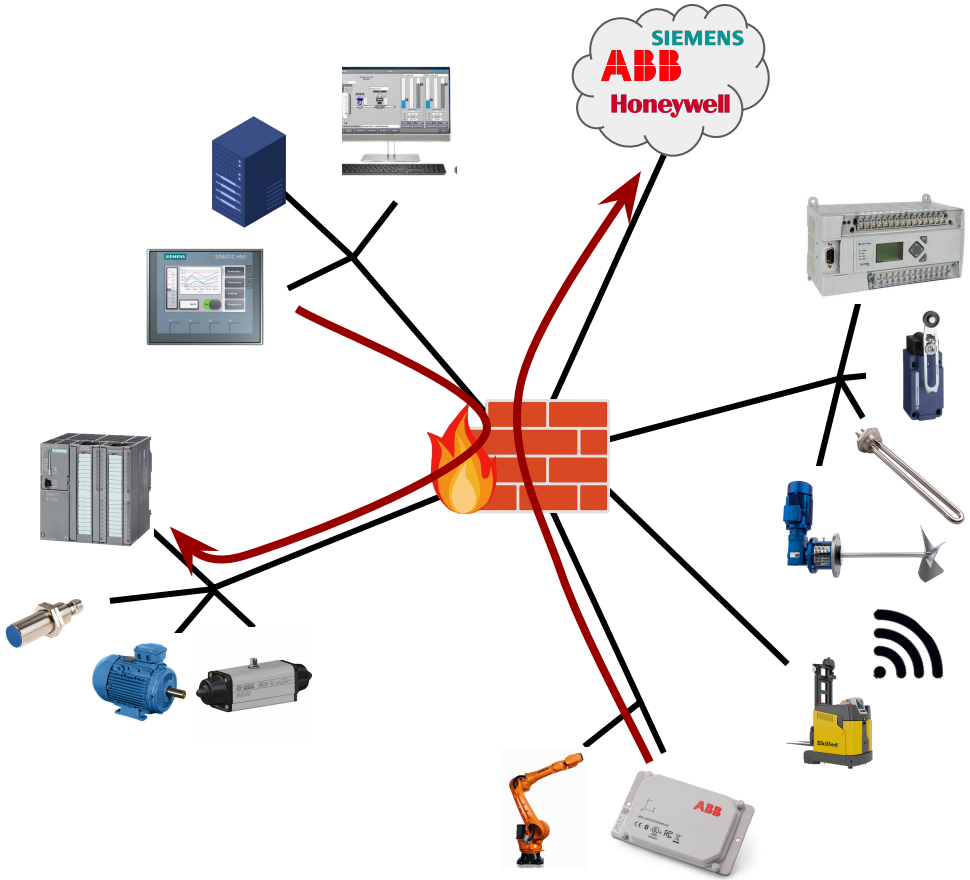
Segment sections / services

Hard to scale

Overlay with indirect routing

Single point of failure

No protection within segments



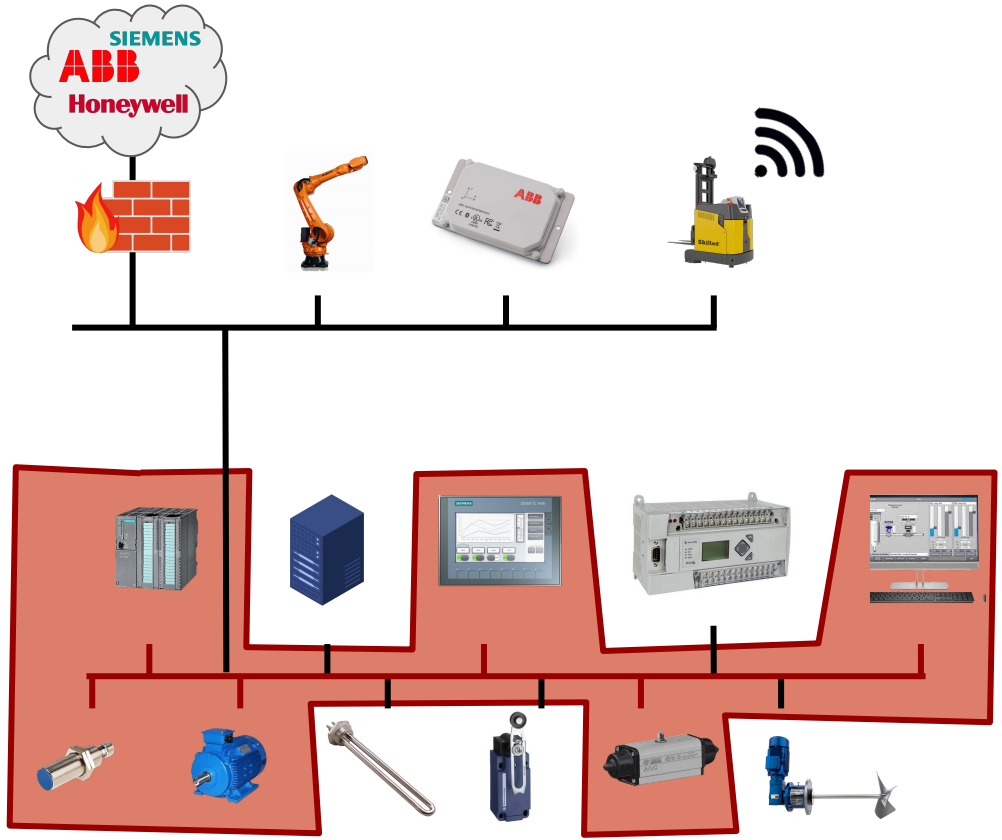
Hopper introduces per-device nano segmentation

Each device's network access is limited to the flows needed for its operation

Enforced in-network at each node and at the receiver

This generates a per-device nano-segment

Compatible with low-end and constrained devices



Nano-segmentation can be broken down to three main security goals

Least Privilege

Only whitelisted flows are allowed

Isolation

Compromised nodes can only affect their own network area

Authentication

Each packet is source authenticated in the network & by receiver

Hopper's design leverages common industrial network properties

Ownership Centrality

One entity sets network policy

Orchestration Centrality

One entity decides which flows are desired

Task-focused

Small number of temporally stable workloads

⇒ A priori whitelist + deny-by-default policy

Hopper allows each node (forwarding elements and receivers) to verify at least part of each authentication tag

Main idea:

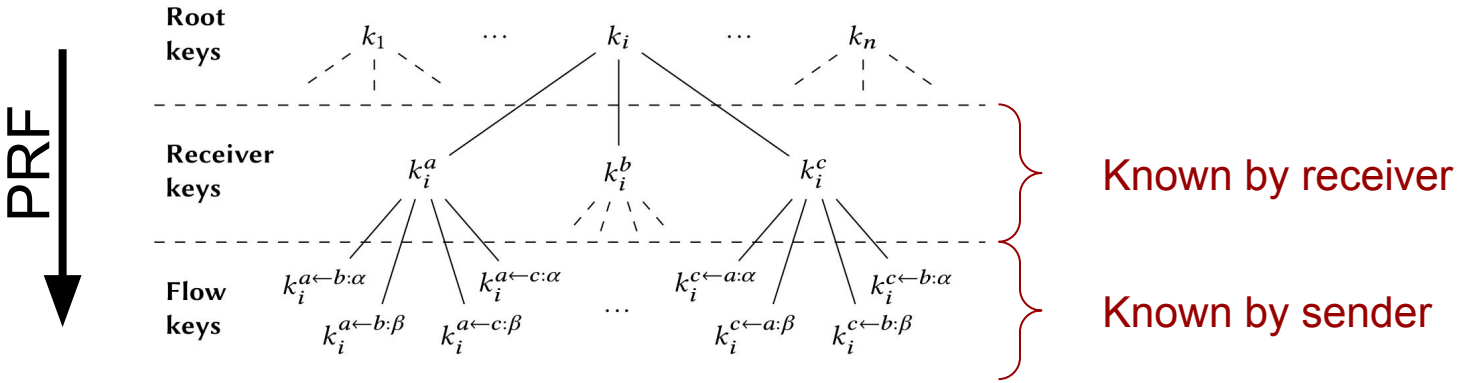
1. Add tags consisting of multiple MACs to each packet

$$\tau = \mathrm{MAC}(k_1^{r \leftarrow s:p}, \mathrm{payload}) \| \ldots \| \mathrm{MAC}(k_n^{r \leftarrow s:p}, \mathrm{payload})$$

2. Distribute keys so each node can verify (part of) the tag.

Forwarding elements & Receivers

Hopper constructs a hierarchical, PRF-based key forest



Lower key = PRF(upper key || *key identifier*)

Hierarchical structure
⇒ receiver can derive all incoming-flow keys

Senders attach a tag consisting of multiple MACs

Receiver
Sender
Port

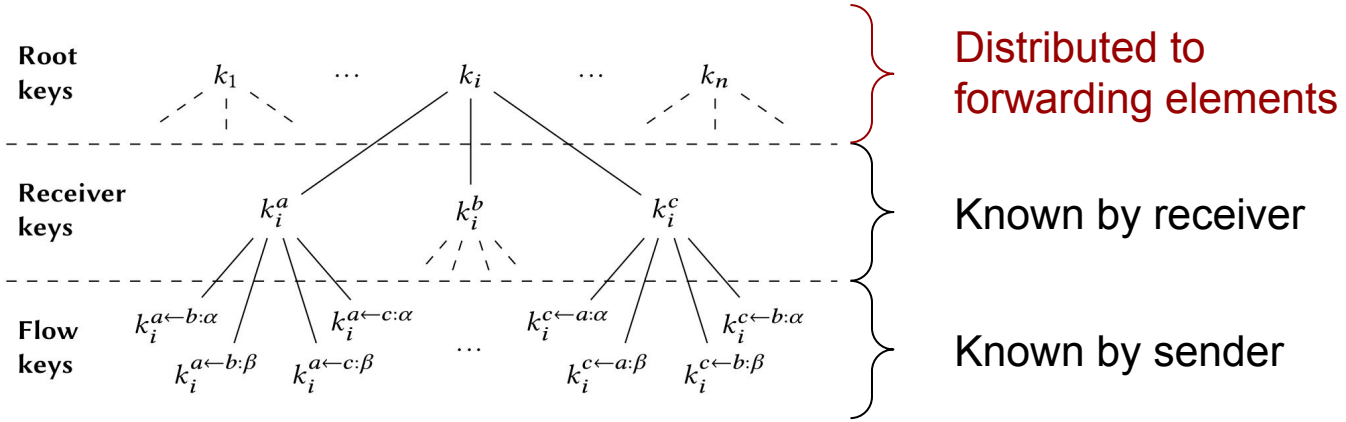
$$\tau = \mathrm{MAC}(k_1^{r \leftarrow s:p}, \mathrm{payload}) \| \ldots \| \mathrm{MAC}(k_n^{r \leftarrow s:p}, \mathrm{payload})$$

Strength of tag is dependent on total length
⇒ individual MACs can be short

Each MAC uses a full-sized key
⇒ Key recovery attacks impossible

Full tag is checked by receiver
⇒ Packets are source authenticated

Forwarding elements receive a subset of root keys

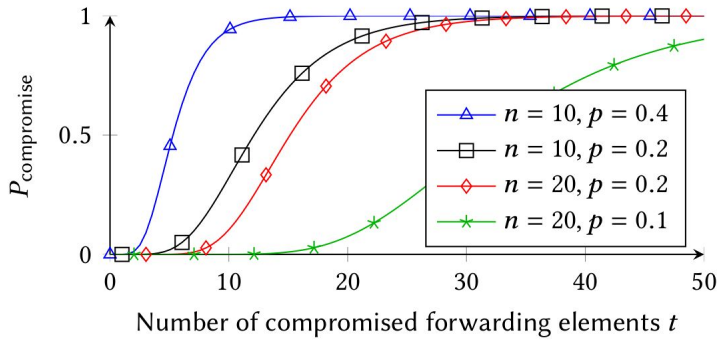


forwarding element can derive all flow keys
for which it has the corresponding root keys
⇒ report & drop packets with incorrect MACs

Distribution schemes:
Cover-free families, random distribution, manual distribution

Random key-distribution schemes provide good properties

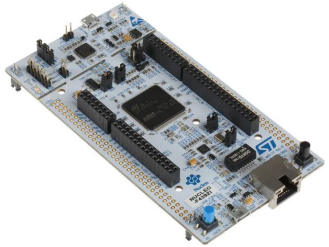
Given *n* root keys, provision each key
on each forwarding element with probability *p*



Receiving hosts verifies full tag + per-packet tags + each bad tag is reported
⇒ Even with 1 uncompromised root key, attacker will be detected

Tricks to further improve scalability are in the paper

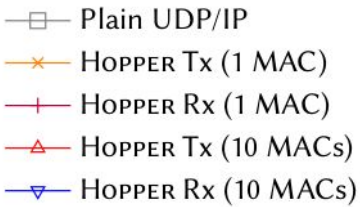
Hopper performs well on IoT-class hardware

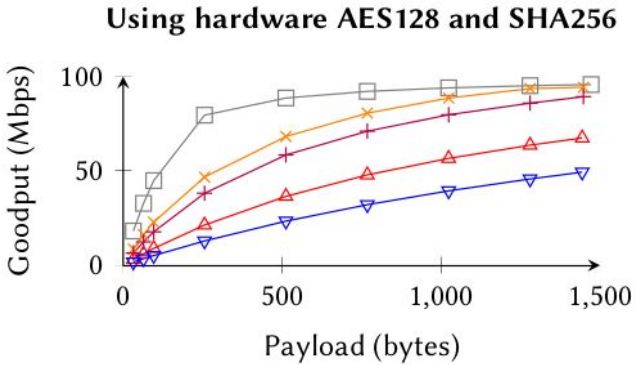


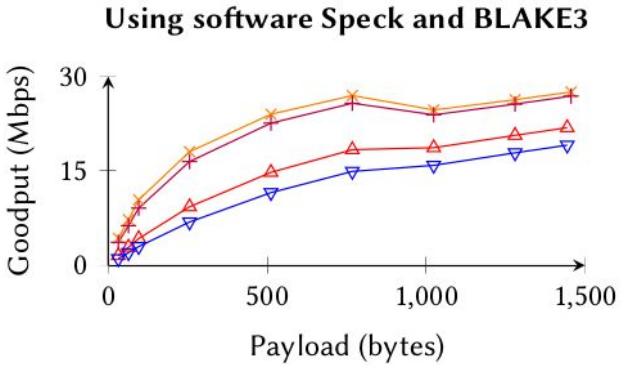
Arm Cortex-M4 @ 180 MHz

lwIP library extensions add
Hopper to UDP and IP

1x128-bit MAC, or 10x16-bit MACs





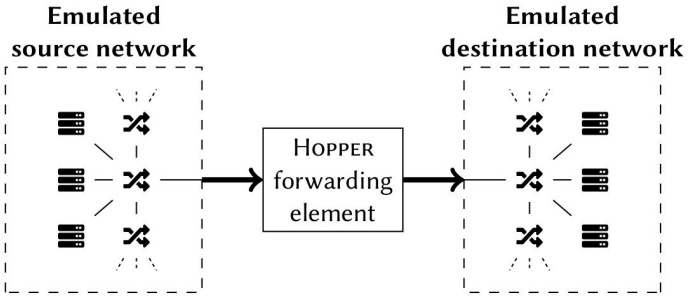


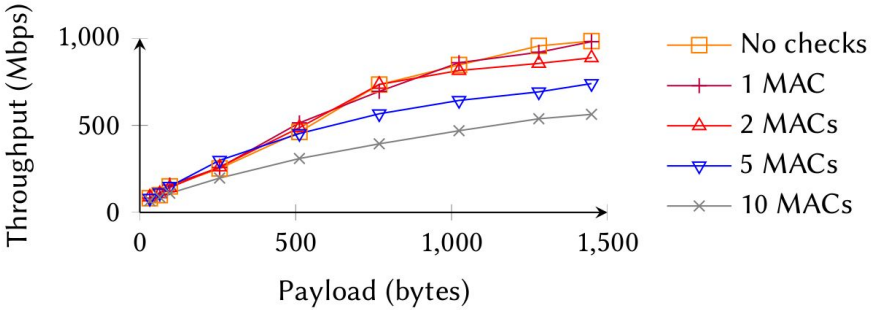
Hopper can be implemented on low-end network hardware

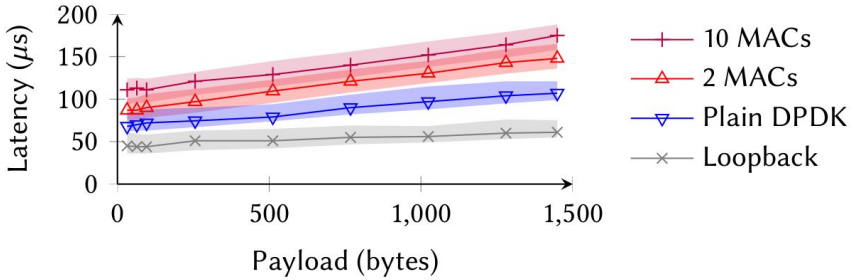
Quad Core AMD @ 1 GHz
4 GB RAM

DPDK + OpenSSL

Invalid packets are dropped







Conclusion

New trends in industrial networks
create new attack opportunities

Current industrial networks
do not protect against lateral movement

Hopper enables lightweight Nano-Segmentation
of industrial IoT networks

⇒ Maximally isolate resources & Minimize lateral movement