Byzantine-Resilient Routing and Key Management Protocols using Network Coding

Cristina Nita-Rotaru

Department of Computer Science and CERIAS
Purdue University

Acknowldegment

This work was funded by *NSF,* Secure Networking Using Network Coding

Relevant publications

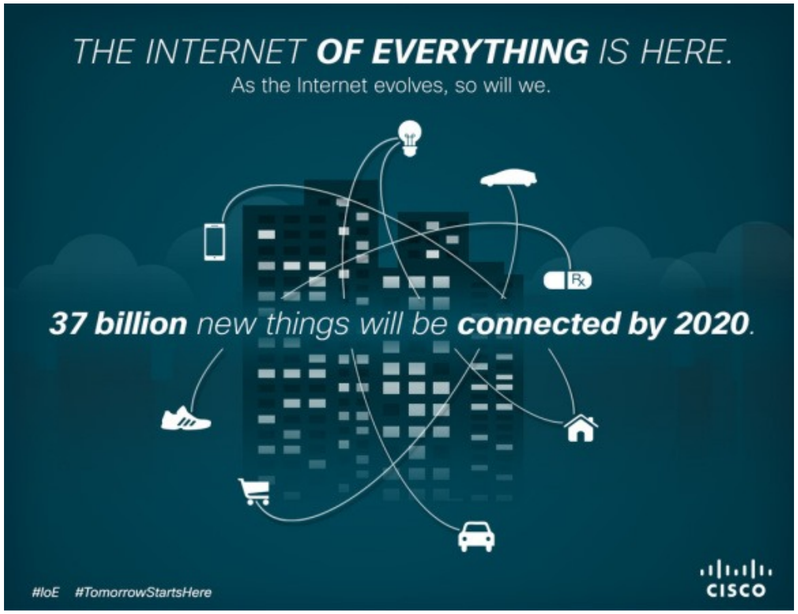
- **Node-Capture Resilient Key Establishment in Sensor Networks: Design Space and Protocols.** Andrew Newell, Hongyi Yao, Alex Ryker, Tracey Ho, and Cristina Nita-Rotaru. ACM Computing Surveys, Jan. 2015
- **On the Practicality of Cryptographic Defenses against Pollution Attacks in Wireless Network Coding.** Andrew Newell, Jing Dong, and Cristina Nita-Rotaru. In ACM Computing Surveys, June 2013.
- **Pollution Attacks and Defense in Inter-flow Network Coding Systems.** Jing Dong, Reza Curtmola, Cristina Nita-Rotaru, and David Yau. In IEEE Transactions on Dependable and Secure Systems, Sept. 2012.
- **Practical Defenses Against Pollution Attacks in Wireless Network Coding.** Jing Dong, Reza Curtmola, and Cristina Nita-Rotaru. In ACM Transactions on Systems and Information Security, vol. 14 no. 1, May 2011.

http://ds2.cs.purdue.edu

- Overarching goal:
  - Create and build distributed systems and network protocols that achieve **security**, **availability**, and **performance** in spite of *misconfigurations*, *failures*, and *attacks*
- Approach:
  - Combine theoretical principles and experimental methodologies from distributed systems, cryptography, networking, information theory, and machine learning

The Internet of everything is here …

- Computing services
  - Everything is connected
  - Many types of devices
  - Tremendous amount of data
  - Available via cloud computing, accessed via personal devices
- Higher expectations
  - Services must be available 24h, working correctly 100% of the time
  - Data-centric business, policy decisions



Users called 911 because Facebook was down !!!

What does it mean for security

- Large number of devices with different capabilities and vulnerabilities managed by different entities
  - Higher chances that some system components are going to be compromised
  - *The next attack is going to come from your kitchen*
- Subset of computing systems or protocol participants controlled by an adversary can influence
  - Communication and availability
  - Da

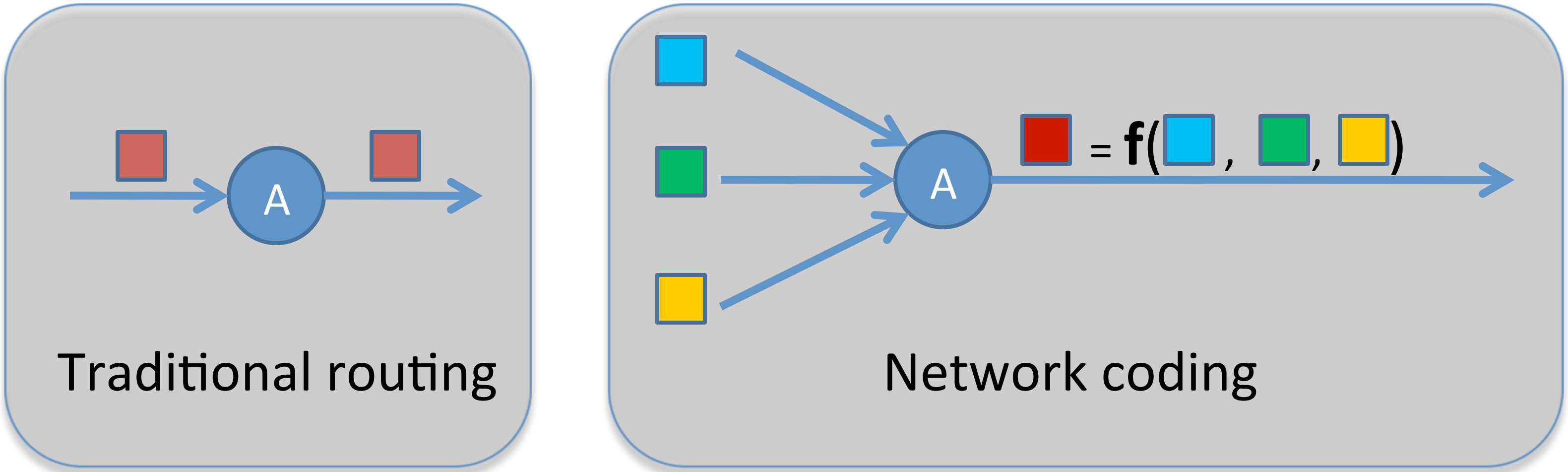
Designing systems resilient to only outsider attackers no longer sufficient, need for insider-resilient systems

Seeing the world through a Byzantine len

- An insider can not be trusted to correctly generate or process data (i.e. lie):
  - Trusting info limitations
    - Many insider nodes collude
    - Not enough history is available
    - Single source of information
- An insider can not be trusted to correctly deliver data:
  - Disseminating info limitations
    - Lack of non-adversarial paths
    - Not enough redundancy
    - Correlated failures

Network coding: A New paradigm

- **Key principle:** packet mixing at intermediate nodes

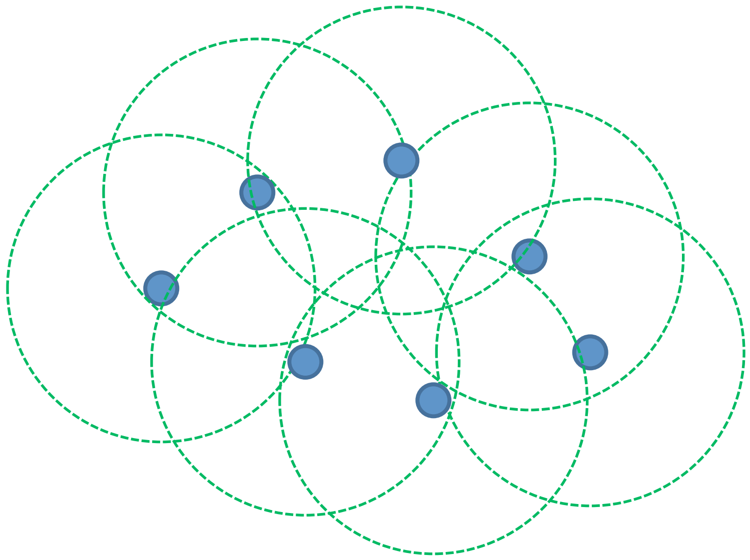


- **Benefits**: Higher throughput, reliability, robustness, energy efficiency
- **Applications**: wireless unicast and multicast, p2p storage and content distribution, delay-tolerant networks, vehicular networks

Network coding in wireless networks

- Opportunities
  - Broadcast advantage
  - Opportunistic listening

- Benefits
  - Improved throughput
  - Reduced delay
  - Improved reliability



This talk

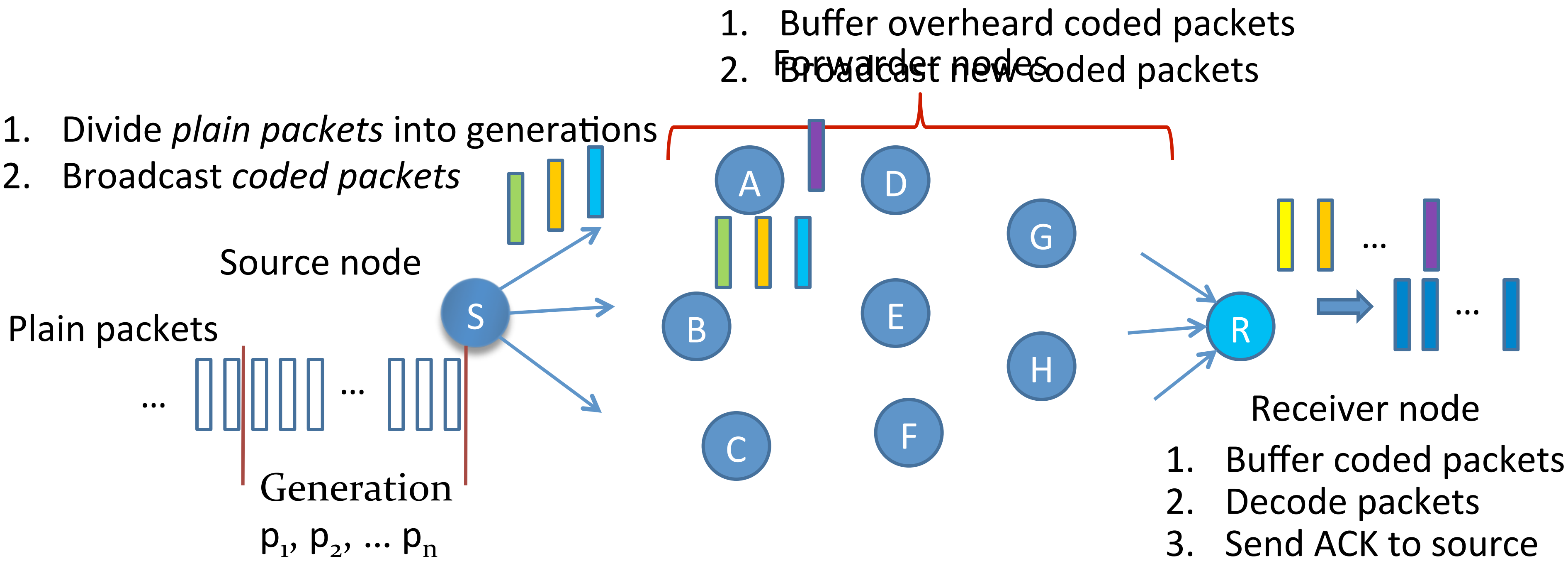
- Network coding under attack:
  - Pollution attacks in intra-flow network coding
- Network coding to the rescue:
  - All pairwise and connected graph key management resilient to node capture

Wireless network coding systems

- Intra-Flow Network Coding
  - Mix packets within individual flows
  - Examples: [Park; 2006], MORE [Chachulski; 2007], [Zhang; 2008a], [Zhang; 2008b], MIXIT [Katti; 2008], [Lin; 2008]

- Inter-Flow Network Coding
  - Mix packets across multiple flows
  - Examples: COPE [Katti; 2006], DCAR [Le; 08], [Das; 2008], [Omiwade; 2008a], [Omiwade; 2008b]

Intra-flow network coding



1. Buffer overheard coded packets
2. Broadcast new coded packets

Forwarder nodes

1. Divide *plain packets* into generations
2. Broadcast *coded packets*

Source node

Plain packets

Generation

$p_1, p_2, \dots p_n$

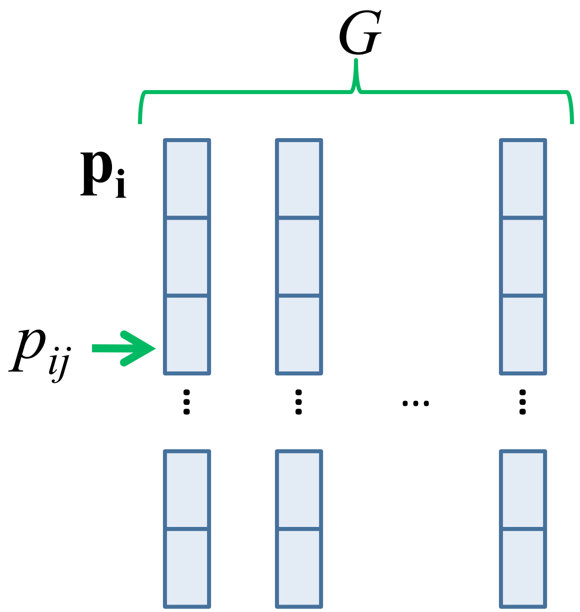
Receiver node

1. Buffer coded packets
2. Decode packets
3. Send ACK to source

Packet coding and decoding

- $\mathbf{p_i} = (p_{i1}, p_{i2}, \ldots, p_{im})^T, p_{ij} \in F_q$
- $G = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}]$
- Coding with random linear combination

$$\mathbf{c} = (c_1, c_2, \ldots, c_n),\ c_i \in F_q$$
$$\mathbf{e} = c_1\mathbf{p_1} + c_2\mathbf{p_2} + \ldots + c_n\mathbf{p_n} = G\mathbf{c}$$



- Decoding
  - Given n linearly independent coded packets $(\mathbf{c_1}, \mathbf{e_1}) \ldots (\mathbf{c_n}, \mathbf{e_n})$ solve a system of linear equations
- Attacks
  - **Packet Pollution**: injecting incorrect packets

Pollution attacks

Definition

- Pollution attacks are attacks where *attackers* inject ***polluted coded packets*** into the network.
- A coded packet (c, e) is a polluted *coded packet* if

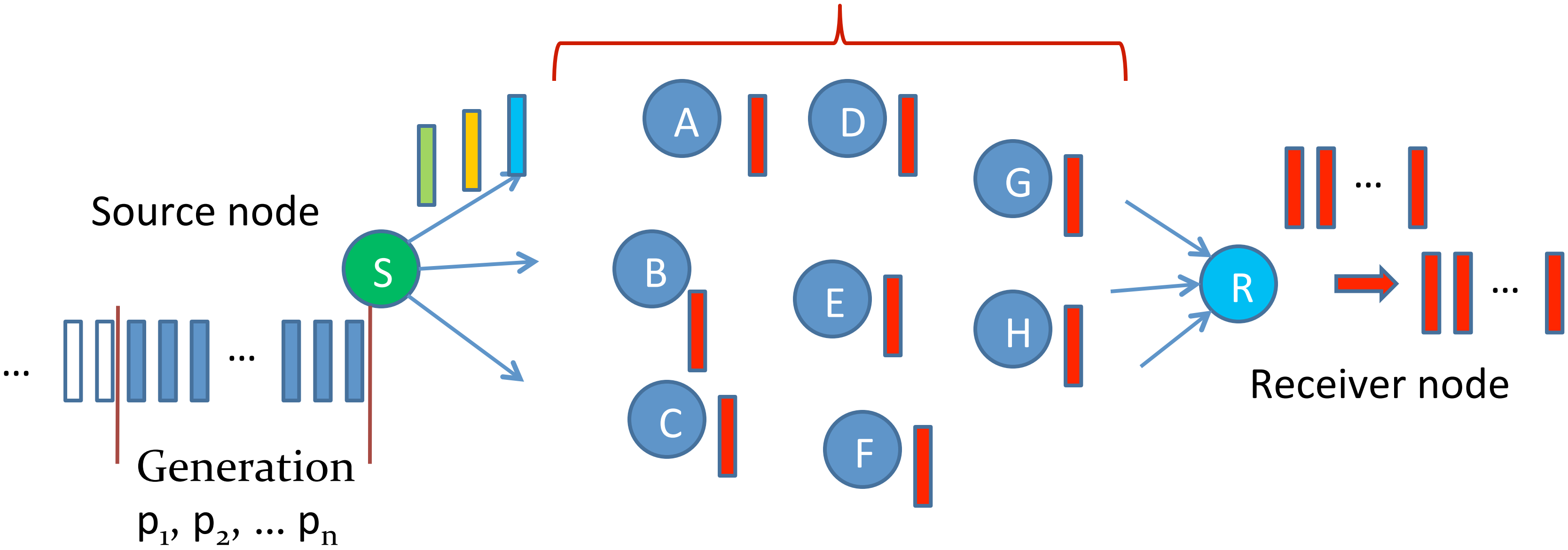
$$\mathbf{c} = (c_1, c_2, \ldots, c_n),\ \mathrm{c_i} \in F_q$$

but

$$\mathbf{e} \neq c_1\mathbf{p_1} + c_2\mathbf{p_2} + \ldots + c_n\mathbf{p_n}$$

- Generic attack to any network coding system

Impact of pollution attacks

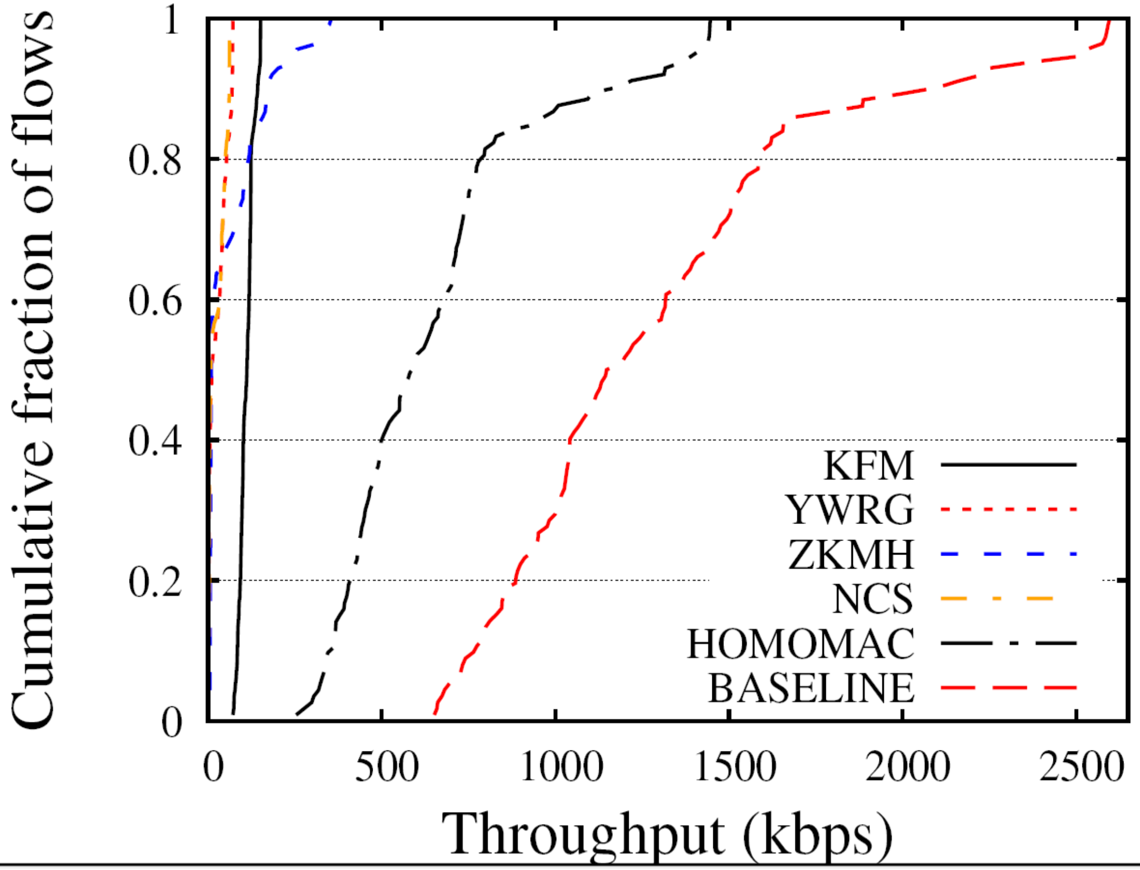


Epidemic attack propagation

Prior work

- Cryptographic approaches [Krohn; 2004], [Li; 2006], [Charles; 2006], [Zhao; 2007], [Yu; 2008], [Boneh; 2009]
  - Homomorphic digital signatures or hash functions
  - *Too expensive computationally*
- Information theoretic approaches [Ho; 2004], [Jaggi; 2007], [Wang; 2007]
  - Coding redundant information
  - *Low achievable throughput*
- Network error correction coding [Yeung; 2006], [Cai; 2006], [Silva; 2007], [Koetter; 2008]
  - Using error correction coding techniques
  - *Limited error correction capability, unsuitable for adversarial environment*

Throughput CDF when no attack happens



The high overhead of crypto-based schemes render them impractical for wireless networks

Our approach

Non-cryptographic checksum created by the source

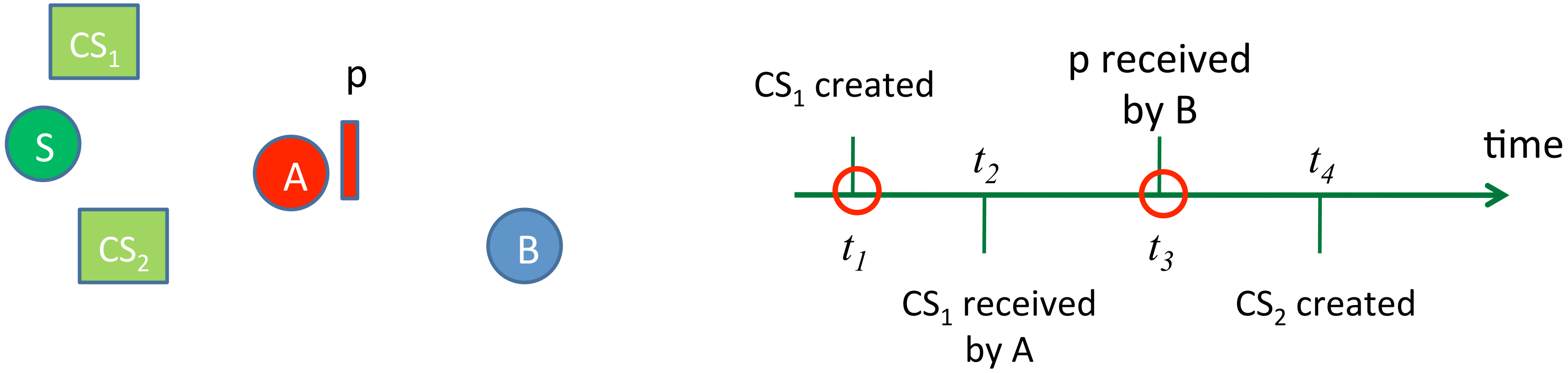
Based on lightweight random linear transformations
Carries the timestamp of when it was created
Disseminated by the source in an authenticated manner
Not pre-image or collision resistant!

Security relies on time asymmetry checksum verification

A node verifies a packet against a checksum that is created ***after*** the packet is received

Our approach: Example

Attacker can not inject a checksum or modify timestamp because checksum is signed by source



Packet p will be verified against CS$_2$ and not CS$_1$. The attacker does not gain anything by observing CS$_1$.

DART and EDART

- DART
  - Forwarder nodes buffer packets checksum verification
  - Only verified packets are combined to form new packets for forwarding
  - Polluted packets are dropped at first hop, eliminating epidemic propagation
- EDART
  - Improves performance with optimistic forwarding

Checksum computation and verification

- A generation of packets $G = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}]$

Checksum computation

- Compute H_s a random $b \times m$ matrix from a seed s
- Compute the checksum

$$\mathrm{CHK}_s(G) = H_s G$$

- b is a system parameter that trades off security and overhead

Checksum verification

Given $\mathrm{CHK}_s(G)$, s *and* t, check if a coded packet $(\mathbf{c}, \mathbf{e})$ is valid

- Check

$$\mathrm{CHK}_s(G)\ \boldsymbol{c} = H_s \boldsymbol{e}$$

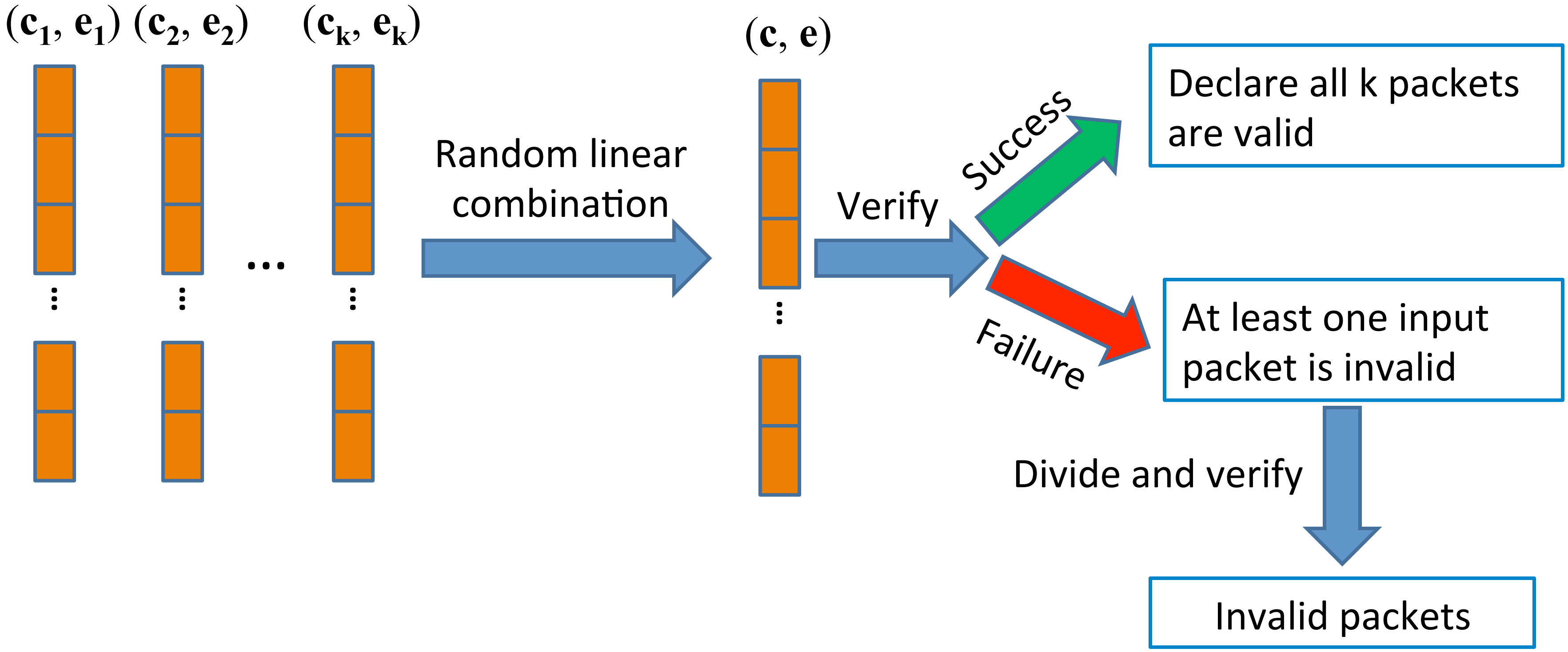
- Why?

$$\mathrm{CHK}_s(G)\mathbf{c} = (H_s G)\mathbf{c} = H_s(G\boldsymbol{c}) = H_s\mathbf{e}$$

- **No false positive, may have false negative**

Batch Checksum Verification

- Verify a set of coded packets $\{(\mathbf{c}_1, \mathbf{e}_1), \ldots, (\mathbf{c}_k, \mathbf{e}_k)\}$ at once



$(\mathbf{c}_1, \mathbf{e}_1)$ $(\mathbf{c}_2, \mathbf{e}_2)$ $(\mathbf{c}_k, \mathbf{e}_k)$

Random linear combination

$(\mathbf{c}, \mathbf{e})$

Verify

Success

Declare all k packets are valid

Failure

At least one input packet is invalid

Divide and verify

Invalid packets

- For higher accuracy, we can repeat the procedure

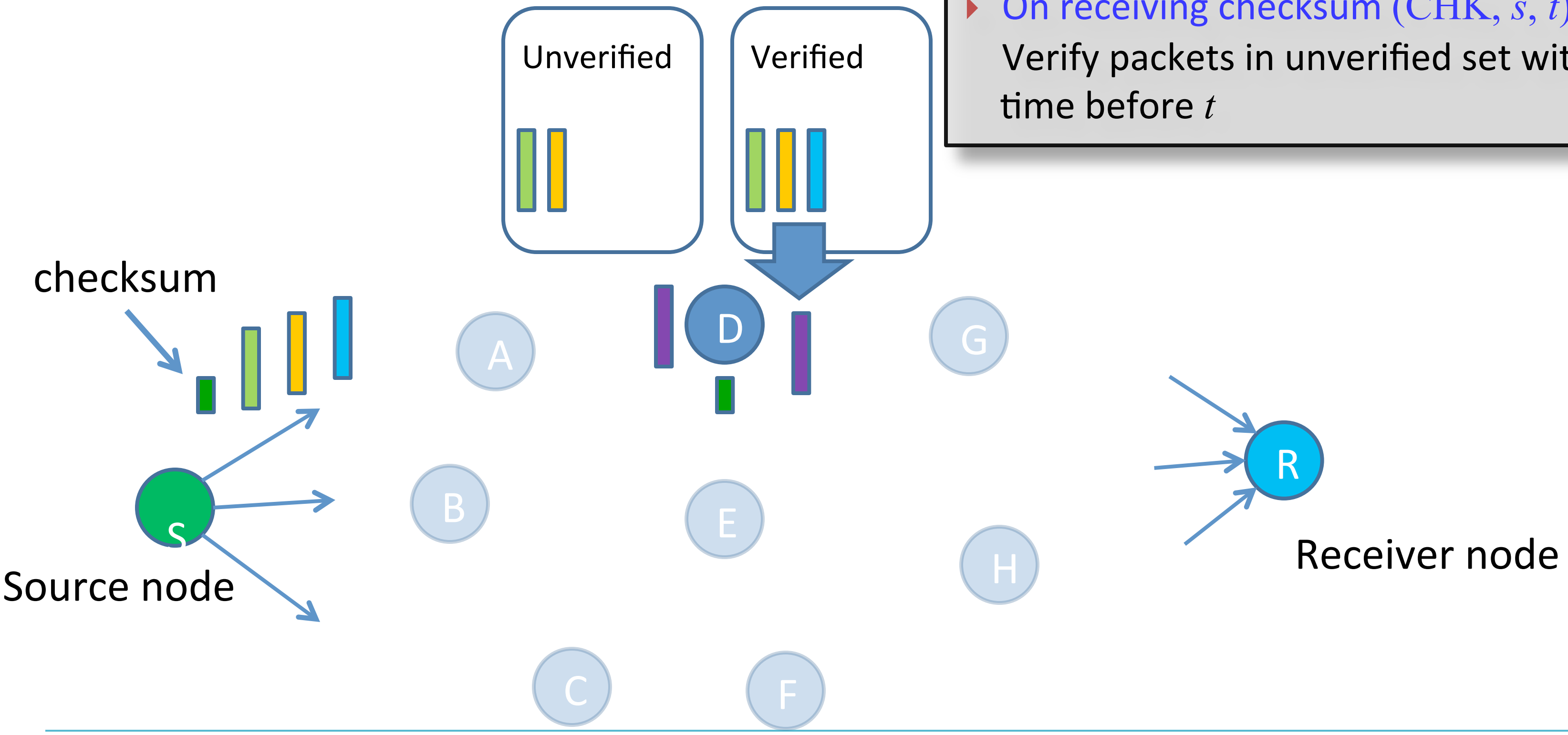
DART Algorithm

Source node

- Disseminate coded packets as usual
- Periodically disseminate a signed random checksum (CHK, s, t)

Forwarder node

- On sending a packet
  Code packets in verified set
- On receiving coded packet p
  Add p to unverified set, record receive time
- On receiving checksum (CHK, s, t)
  Verify packets in unverified set with receive time before t



DART Overhead Analysis

- Computation overhead
  - Checksum computation
    - $\mathrm{CHK}_s(G) = H_s G$
  - Checksum verification
    - $\mathrm{CHK}_s(G)\mathbf{c} = H_s\mathbf{e}$
- Communication overhead
  - Dissemination of checksum packet $(\mathrm{CHK}_s(G), s, t)$
    - s: random seed, e.g. 4 bytes
    - t: timestamp, e.g. 4 bytes
    - $\mathrm{CHK}_s(G)$: b × n matrix over F_q
      - ☐ Example: b=2, n=32, $q=2^8$, $\mathrm{CHK}_s(G)$ is 64 bytes

DART security analysis

Claim

- The probability that a polluted packet can pass the checksum verification is $1/q^b$
- In batch verification, the probability that a polluted packet passes w independent batch verification is $1/q^b + 1/q^w$

- Example: $q = 2^8$, $b = 2$
  - 1 in 65536 polluted packets can pass first hop neighbor
  - 1 in over 4 billion polluted packets can pass second hop neighbor

EDART

- DART delays packets for verification, increasing latency

Ideally,

- Delay polluted packets for verifying
- Forward correct packets without delay

But,

- We do not know which packets are correct and which are polluted

EDART overview

- Packets are **always** verified BUT
- Nodes *"closer"* to the attacker **delay** packets for verification
- Nodes *"farther"* away from the attacker **forward** packets without delay and will verify them when possible

- Polluted packets are restricted to a region around the attacker
- Correct packets are forwarded without delay
- In case of no attack, all packets are forwarded without delay – **almost no impact on performance**

How to decide when to delay?

- h_{uv} : Add a hop count that captures the number of hops a packet has traveled since the last verification
  - All verified packets will have h_{uv} set to 0
  - **Packets that traveled less than δ hops will be forwarded without delay, otherwise a node delays them**
  - When coding a new packet, set $h_{uv} = h_{max} + 1$ to be the maximum h_{uv} in the packets used to create the new packet
  - If pollution was detected, the node will switch for a time proportional with how big h is to delaying all packets

EDART Algorithm

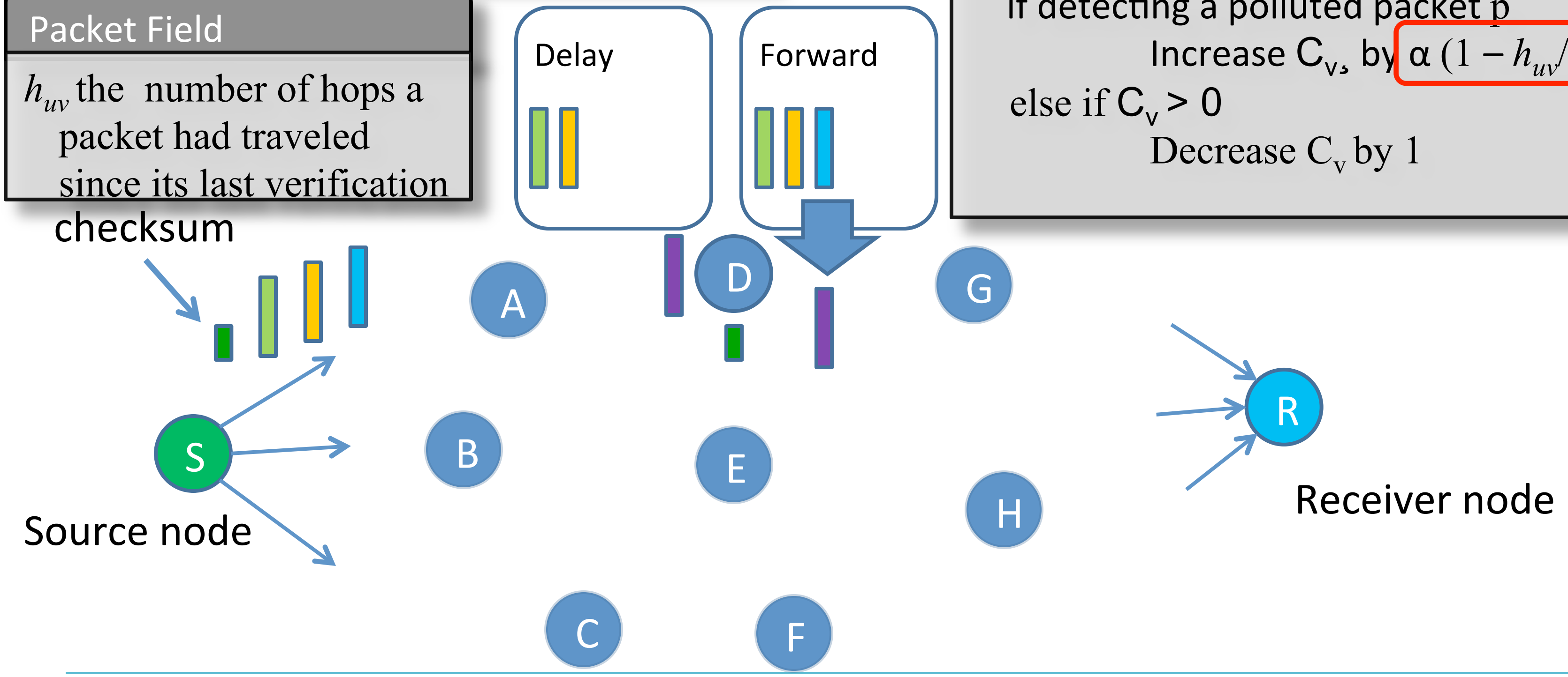
Forwarder Node State

Node mode
- Delay mode
- Forward mode

Delay forward timer
- $C_v = 0 \rightarrow$ in forward mode

Packet Field

h_{uv} the number of hops a packet had traveled since its last verification



Forwarder Node Algorithm

```
On sending a packet
    Code packets in forward set
On receiving coded packet p
    if $C_v > 0$ or $h_{uv} \geq \delta$
        Add p to delay set
    else
        Add to forward set
On receiving checksum $(CHK, s, t)$
    Verify unverified packets (delayed or not)
    if detecting a polluted packet p
            Increase $C_v$, by $\boxed{\alpha (1 - h_{uv}/\delta)}$
    else if $C_v > 0$
            Decrease $C_v$ by 1
```

EDART security analysis

- Maximum pollution scope
  - Bounded by $\delta+1$
- Average pollution scope
  - Bounded by δ/α
- Maximum pollution success frequency
  - Bounded by δ/α

Security

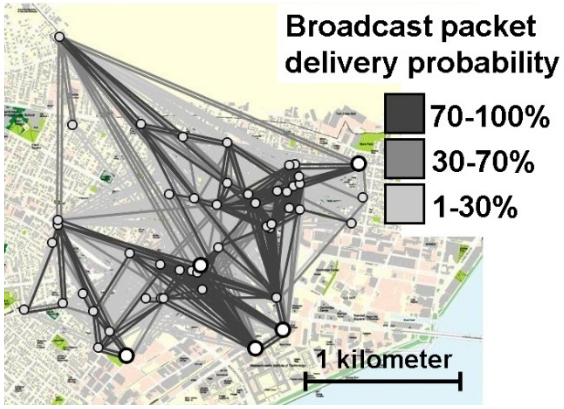
- Unnecessary delay
  - Nodes at i hops away from the attacker $(2 < i < \delta-h-1)$: $\alpha(1 - (h+i)/\delta)$
  - Nodes more than $\delta-h-1$ hops away: 0

Performance

The selection of δ and α trades off security and performance

Experimental evaluations

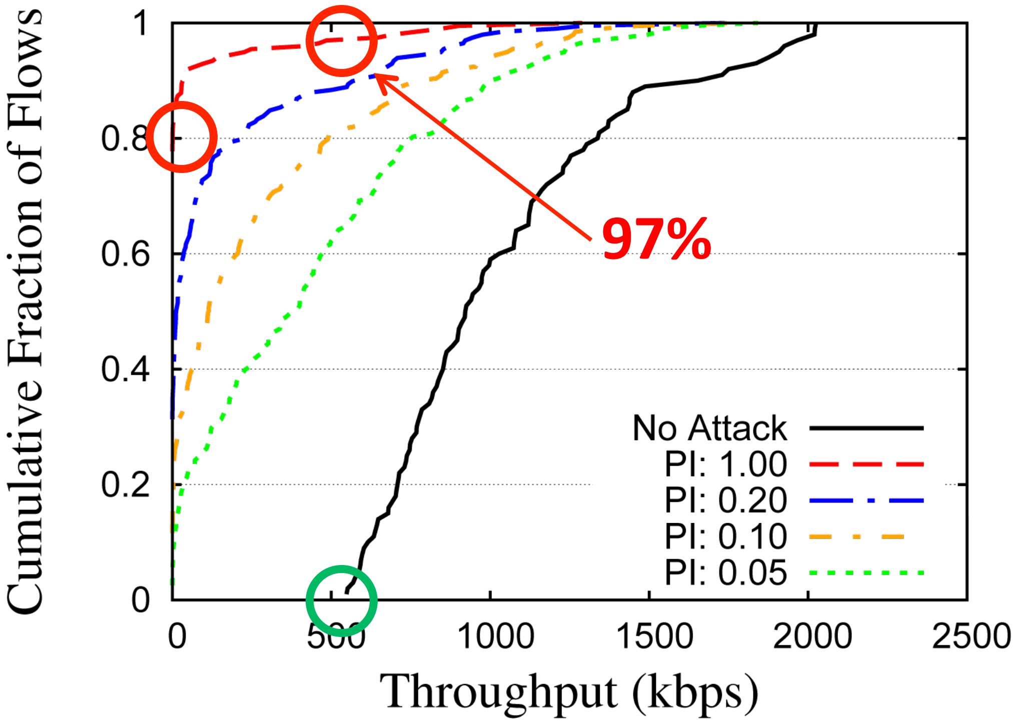
- Network coding system: MORE
- Simulator: Glomosim
- Trace driven physical layer
  - MIT Roofnet trace



- MORE setup
  - GF(2^8), generation size 32, packet size 1500 bytes
- Defense setup
  - RSA-1024 digital signature
  - Checksum size parameter b = 2
  - EDART setup δ = 8, α = 20

Impact of pollution attacks

Throughput CDF under a single pollution attacker with various pollution intensity



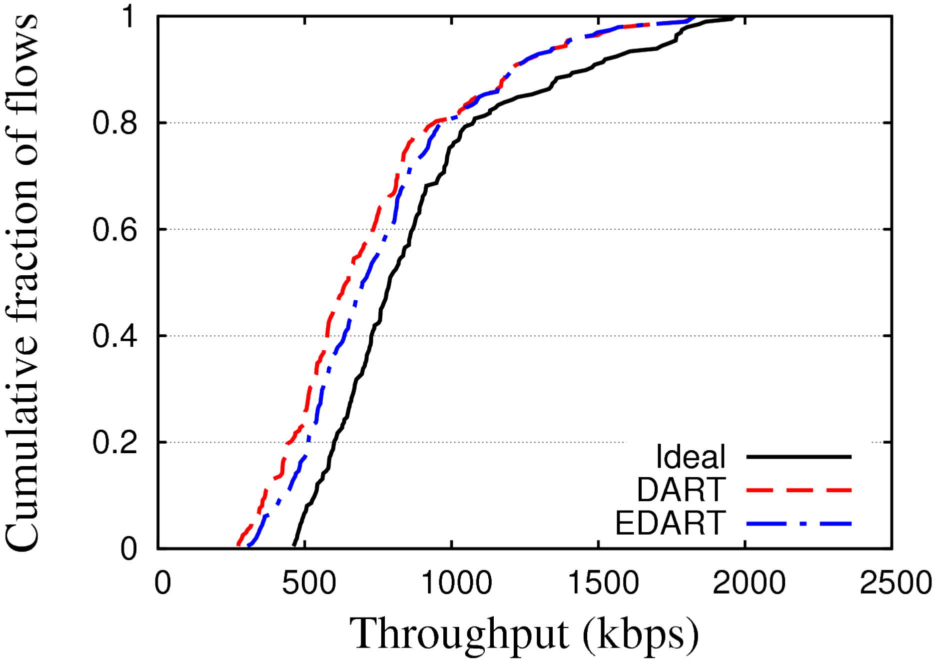
Pollution intensity (PI): number of polluted packets injected per packet received

Even a single pollution attacker can be extremely detrimental!

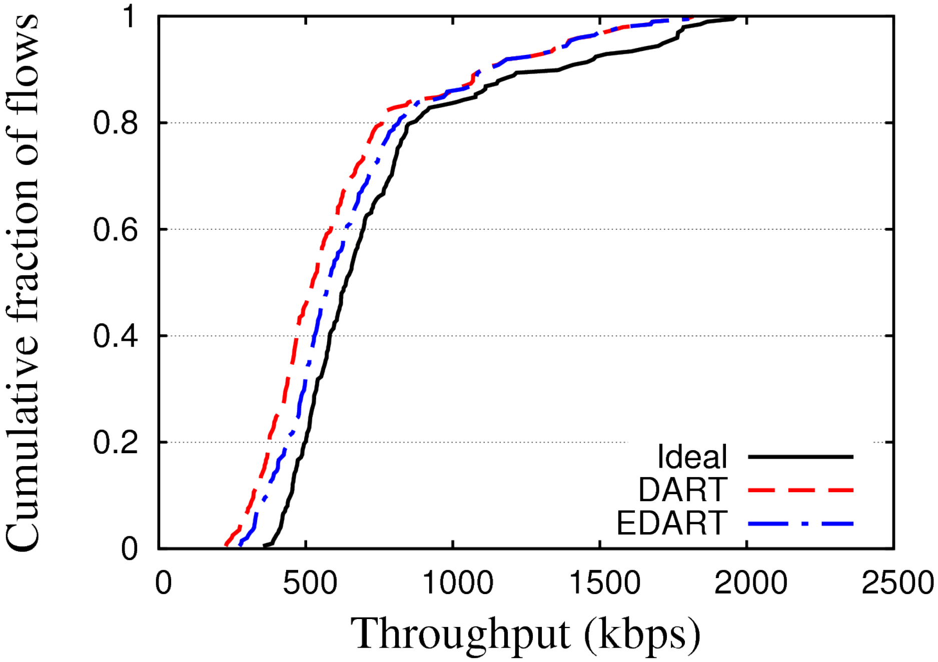
Effectiveness of DART and EDART

Ideal Defense: defense scheme that drops polluted packets with zero overhead

Defense under 5 attackers

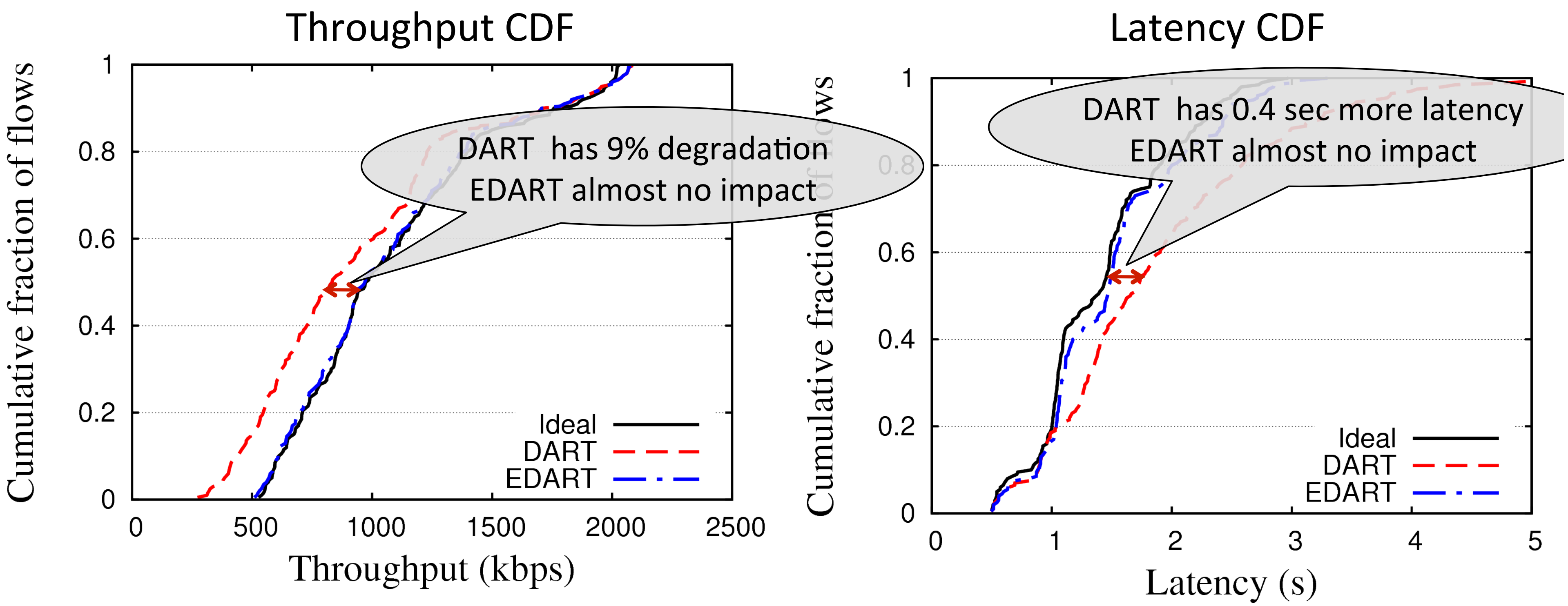


Defense under 10 attackers



Both DART and EDART are very effective against pollution attacks

Performance in benign networks


Throughput CDF

DART has 9% degradation
EDART almost no impact

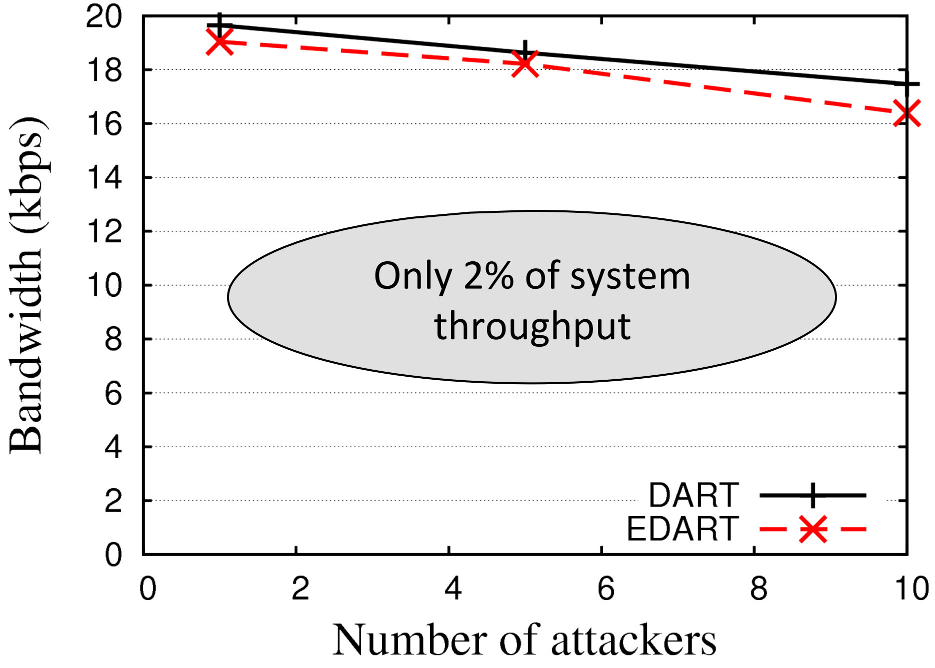
Latency CDF

DART has 0.4 sec more latency
EDART almost no impact

Both DART and EDART have good performance
EDART has almost zero performance impact

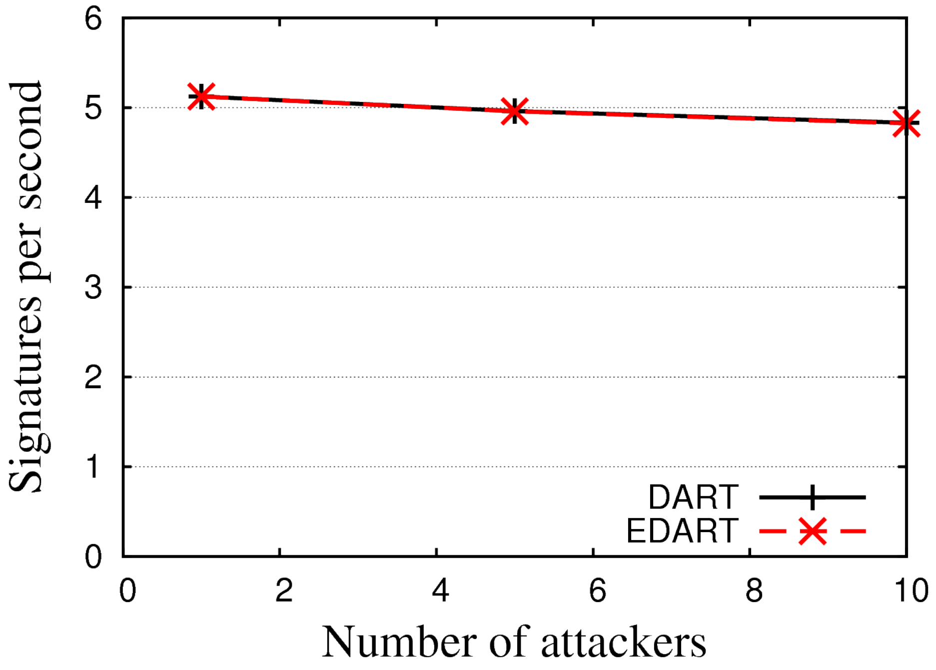
Overhead of DART and EDART

Bandwidth overhead



Only 2% of system throughput

Computation overhead



Both DART and EDART incurs small bandwidth and computation overhead

Null Keys

- Valid coded packets belong to a subspace **A**
- A null key **K** is a subspace of **N(A)**, **N(A)** is the null space of **A**
  - If **c** in **A** then **c * K = 0**
  - If **c** not in **A** then **c * K ≠ 0** with high probability
- Low computational overhead for verification compared to digital signature/hash schemes

A basic approach

- Source distributes null keys to some forwarders
- Forwarders exploit subspace property of null keys to combine their null keys for other forwarders
- Path diversity ensures a forwarder's null keys do not span the space of a downstream node's null keys

- However
  - No path diversity in wireless
  - Null keys are very large

Our Approach

Splitting the null keys

- Generation independent part
  - Large (7340 bytes in our typical scenario)
  - Constant for multiple generations
- Generation dependent part
  - Small (160 bytes in our typical scenario)
  - Updated each generation
- Source distributes large independent parts once
- Source periodically updates smaller dependent parts

Advantages

Low communication overhead

No need for forwarders, source can send the key updates

Splitting Null Keys

Notation

- n – number symbols in coding header
- m – number symbols of coded data
- w – Size of null key
- K – null key ((n+m) x w matrix)
- K_d – generation dependent null key (n x w matrix)
- K_i – generation independent null key (m x w matrix)
- X – data for generation (n x m matrix)

Key Splitting

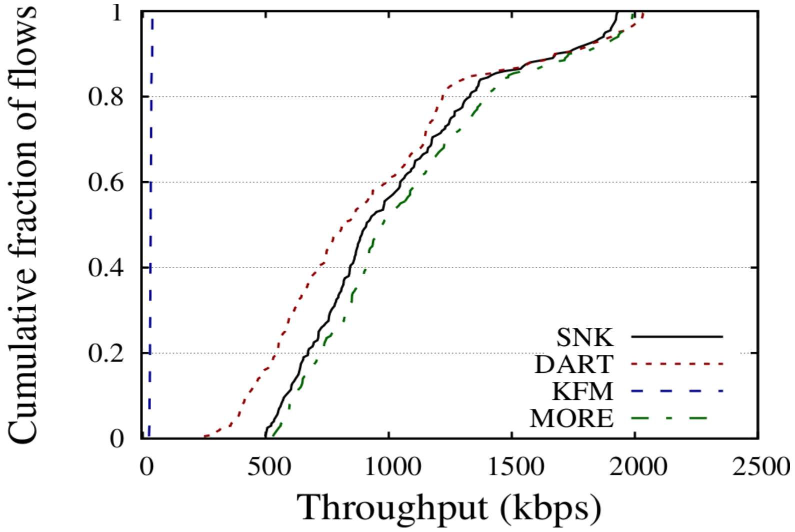
1) Initialize $\mathbf{K_i}$ randomly
2) $\mathbf{K_d} := \mathbf{X} * \mathbf{K_i}$
3) $\mathbf{K} = [\ \mathbf{K_d}^T \mid \mathbf{K_i}^T\ ]^T$

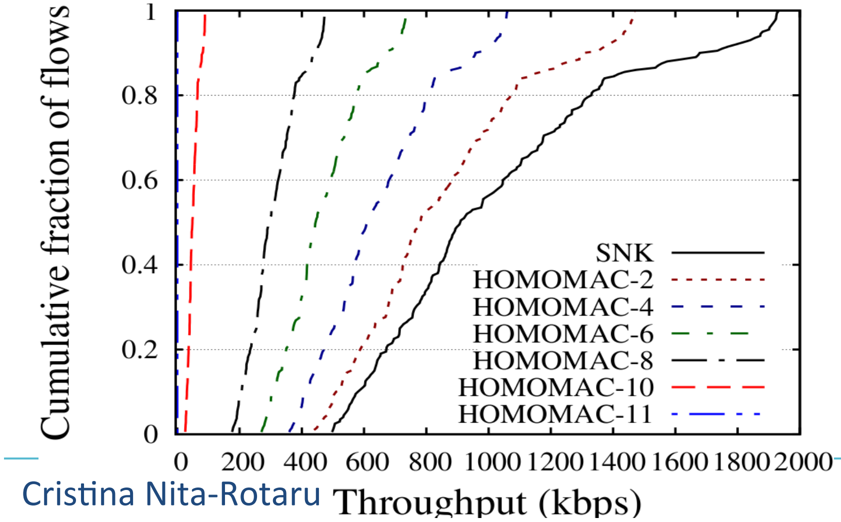
Packet Verification

$\mathbf{c} * \mathbf{K} = \mathbf{0}$ if **c** from **X**

n << **m** so $\mathbf{K_d} << \mathbf{K_i}$

Comparison with pollution defenses

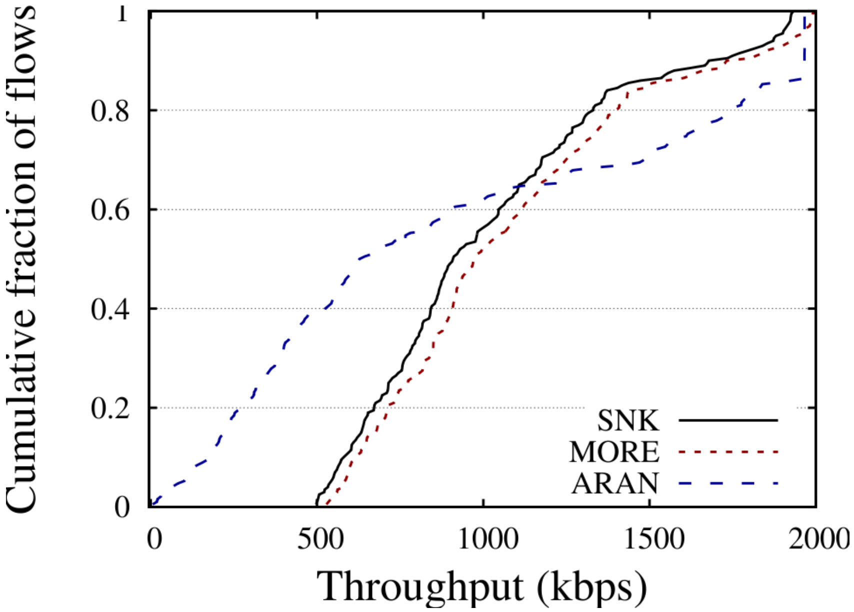




- **SNK** – Split Null Keys
- **DART** – Wireless defense based on time-sensitive checksums
- **KFM** – Representative crypto-based scheme
- **MORE** – Network coding without defense overhead
- **HOMOMAC-*x*** – MAC-based scheme resilient to *x* attackers

- SNK outperforms other defenses
  - Low computational overhead
  - No delaying of packets
  - Not sensitive to multiple attackers

Retains coding gains



- **SNK** – Split Null Keys
- **MORE** – Network coding without defense overhead
- **ARAN** – Secure best-path-routing protocol

- SNK retains coding gains of MORE while providing defense against attackers

This talk

- Network coding under attack:
  - Pollution attacks in intra-flow network coding
- Network coding to the rescue:
  - All pairwise and connected graph key management resilient to node capture

Key distribution in wireless network

How to bootstrap trust in a wireless (sensor) network?

- Establish secret keys
  - All pairwise keys: Symmetric keys are established between every pair of nodes in the network
  - Connected graph: Enough keys are established to ensure that the network graph is connected
- By using different types of communication
  - Direct: nodes communicate directly
  - Multi-hop: nodes communicate through intermediate nodes
    - Single path
    - Multi-path

Resilience to node capture

How many keys get compromised when a node is captured?

- All nodes share the same key
  - Compromise of a node means compromise of the entire network
- Pairwise keys
  - Only the keys shared by the compromised node with other nodes in the network get compromised
- Connected graph
  - Each node requires fewer keys, but can result in high communication overhead as the shortest path over secure links may be larger than the shortest path over all possible links.

Typical key establishment steps

- Network operator first initializes each sensor with a set of secret keys chosen from a large pool
- Sensor nodes are dispersed randomly and uniformly in an environment
- Sensor nodes discover their physical neighbors determined by a fixed transmission range
- Pairs of physical neighbors aim to establish a secret key by using their pre-shared keys
  - communicating directly
  - communicating with other nodes over multi-hop paths

Factors in the design space

- Secrecy and correctness (i.e. integrity, i.e. resilience) of the keys – depending on adversarial model during the key establishment
- Memory constraints
  - How many keys does a node store?
- Network resilience to attacks
  - How many secure links (secret keys) are compromised when a node is compromised: security constraints
- Communication overhead
  - Communication overhead needed to establish keys and communicate securely

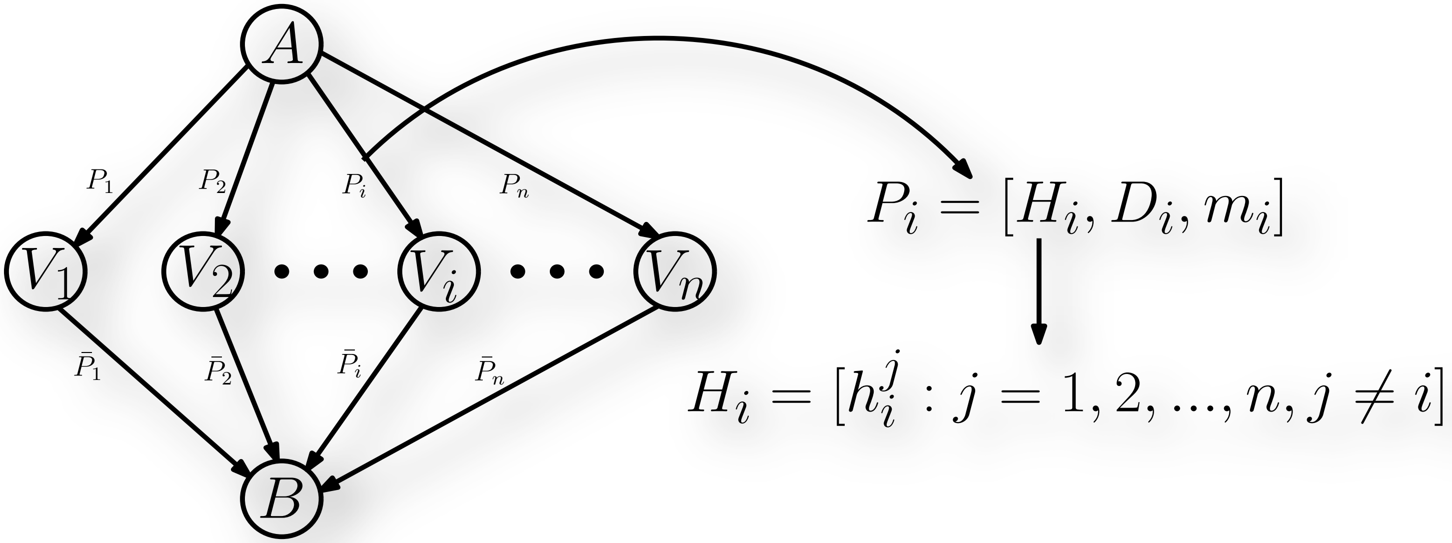
Our approach

- New coding technique
  - Single-path scheme
  - Multi-path scheme for both connected component and all pairwise keys
  - Provides both secrecy and correctness
  - Maximal rate

  Based on H. Yao, D. Silva, S. Jaggi, and M. Langberg. 2010. Network codes resilient to jamming and eavesdropping. In *NetCod 2010*
- Assume attackers are present during key establishment

Coding technique

- Secrecy and correctness under bounded number of adversaries



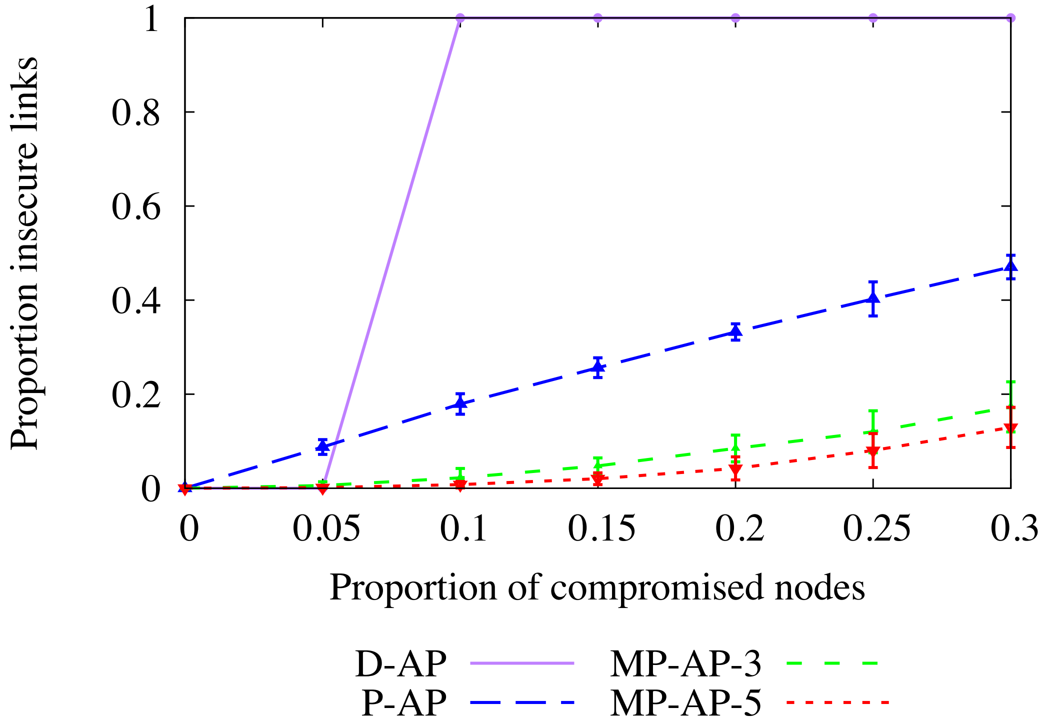
$$P_i = [H_i, D_i, m_i]$$

$$H_i = [h_i^j : j = 1, 2, ..., n, j \neq i]$$

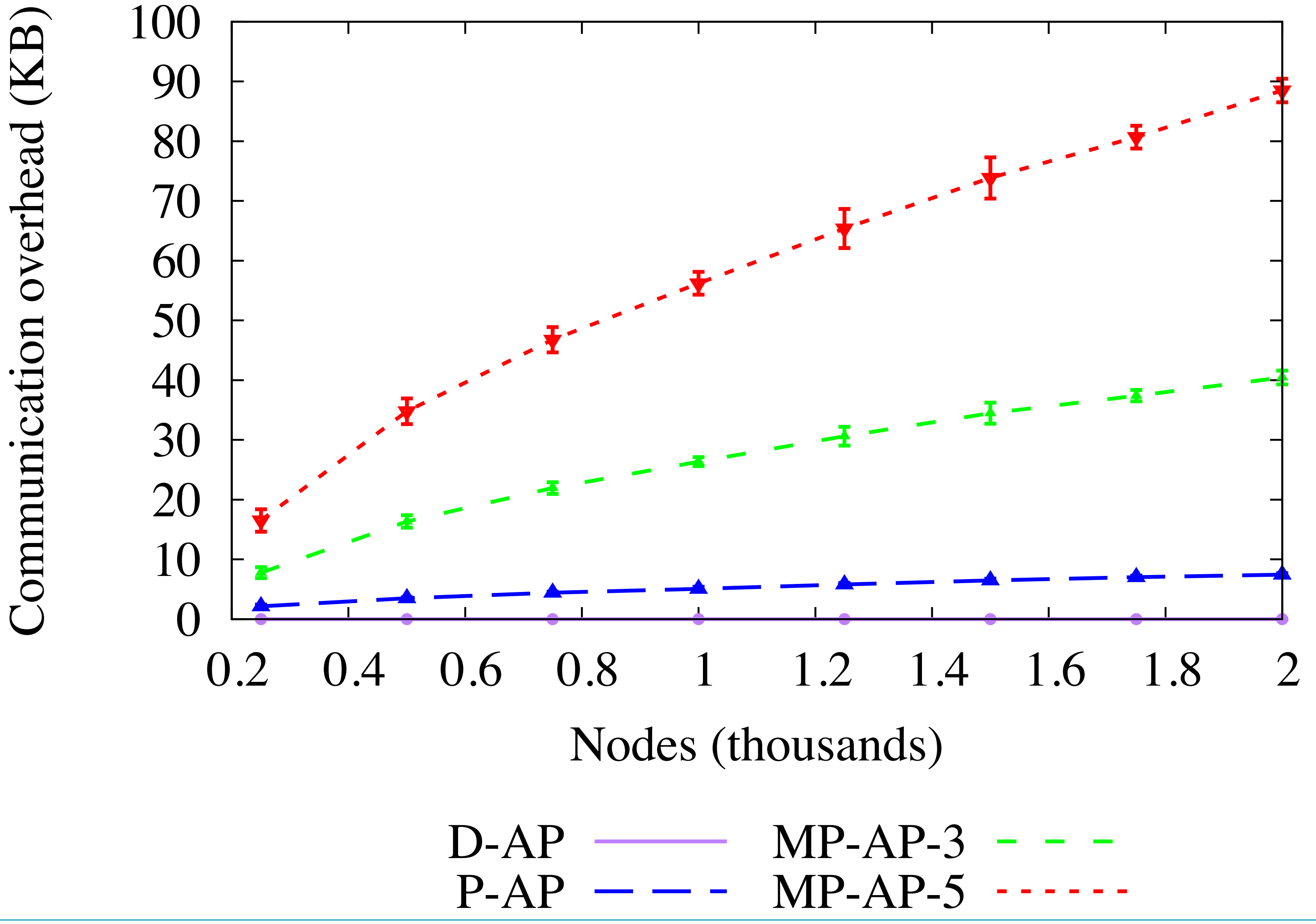
Evaluation goals

- How do changes in the proportion of compromised nodes, available memory and network size affect the resilience to node compromises for each scheme
- How do changes in the network size and density affect the communication overhead for each scheme
- How do all pairwise keys schemes compare with connected graph schemes
- How do changes in the number of disjoint paths for the multi-path schemes affect overhead and security

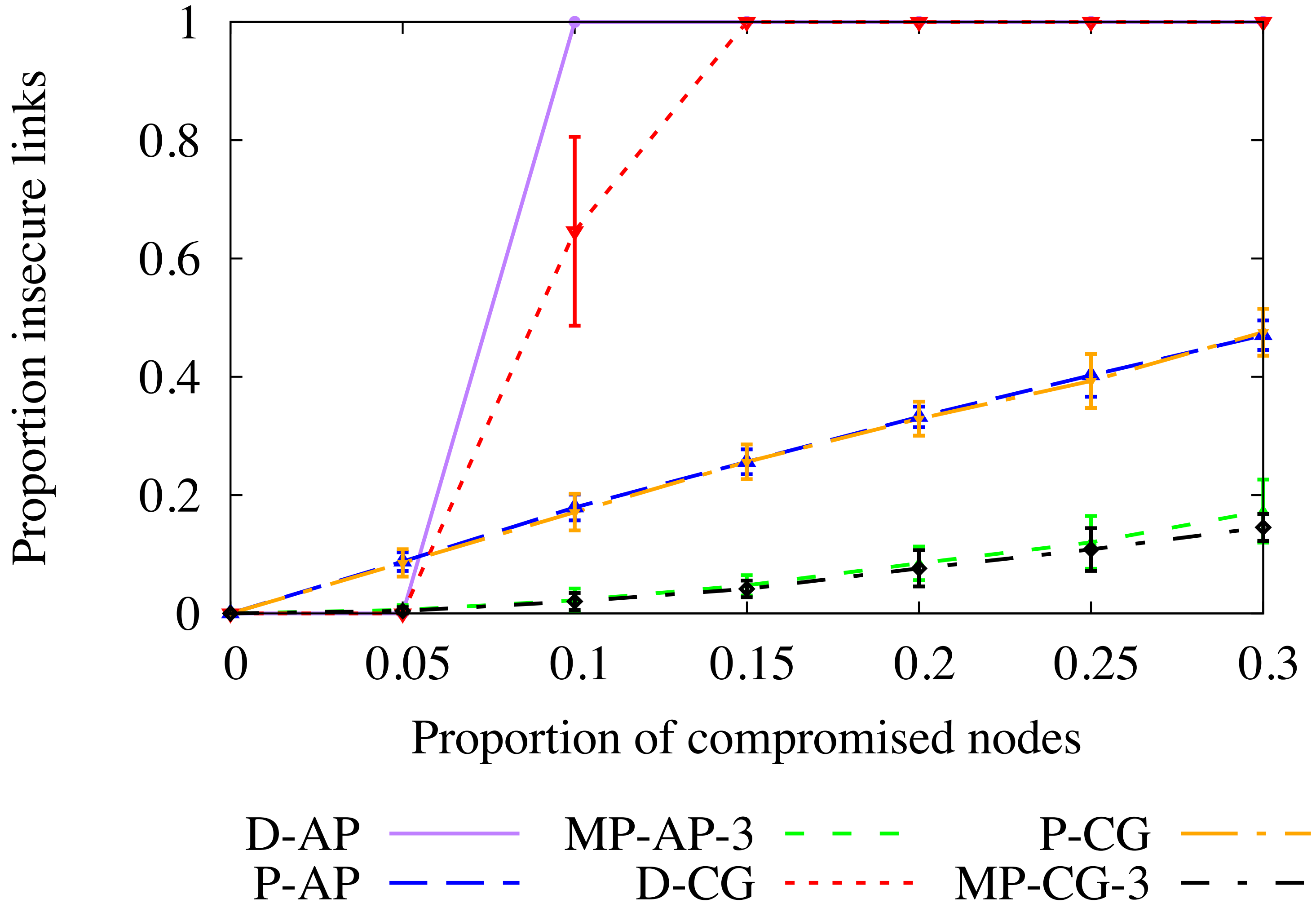
All pairwise: Proportion of insecure links



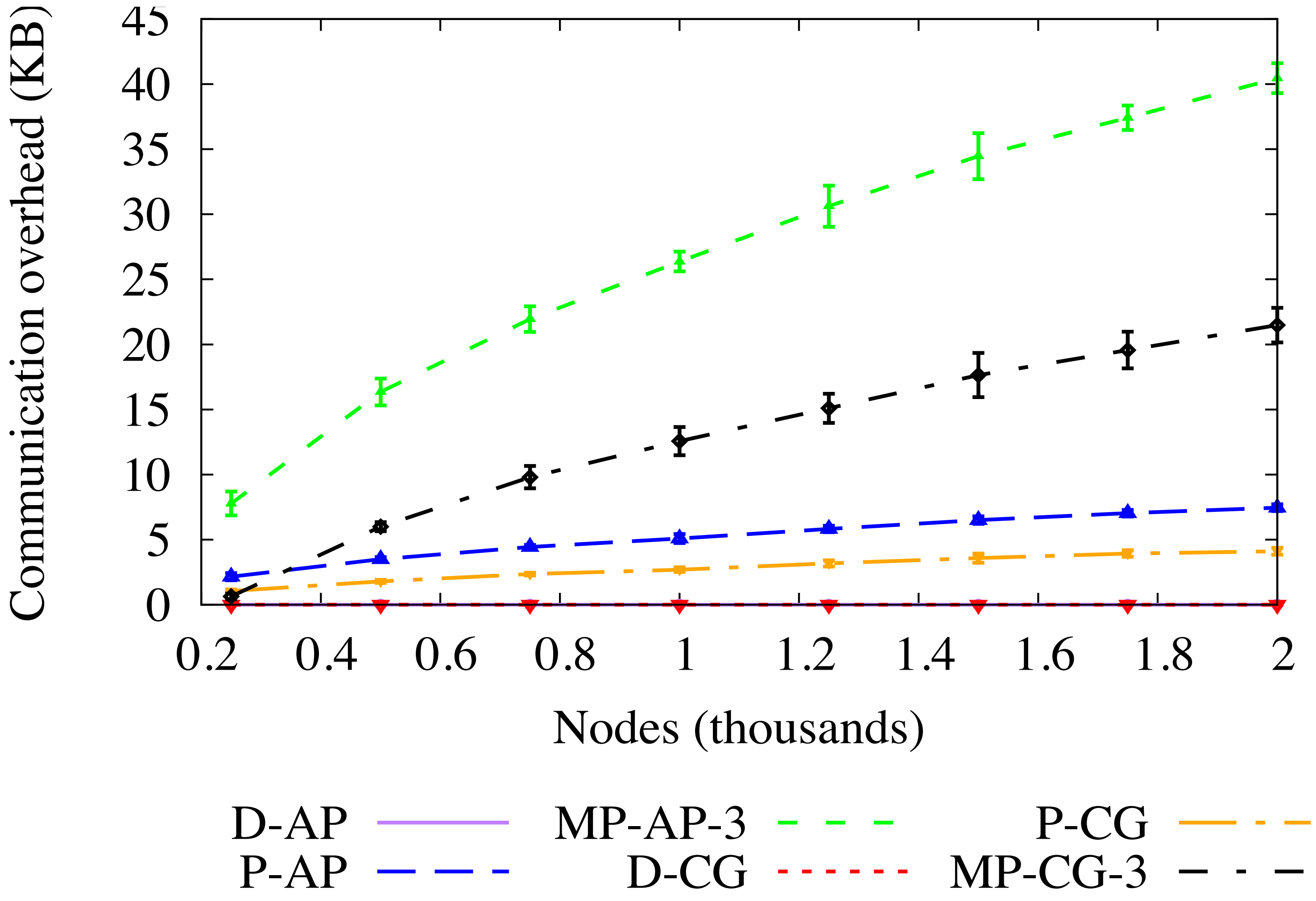
All pairwise: Communication overhead



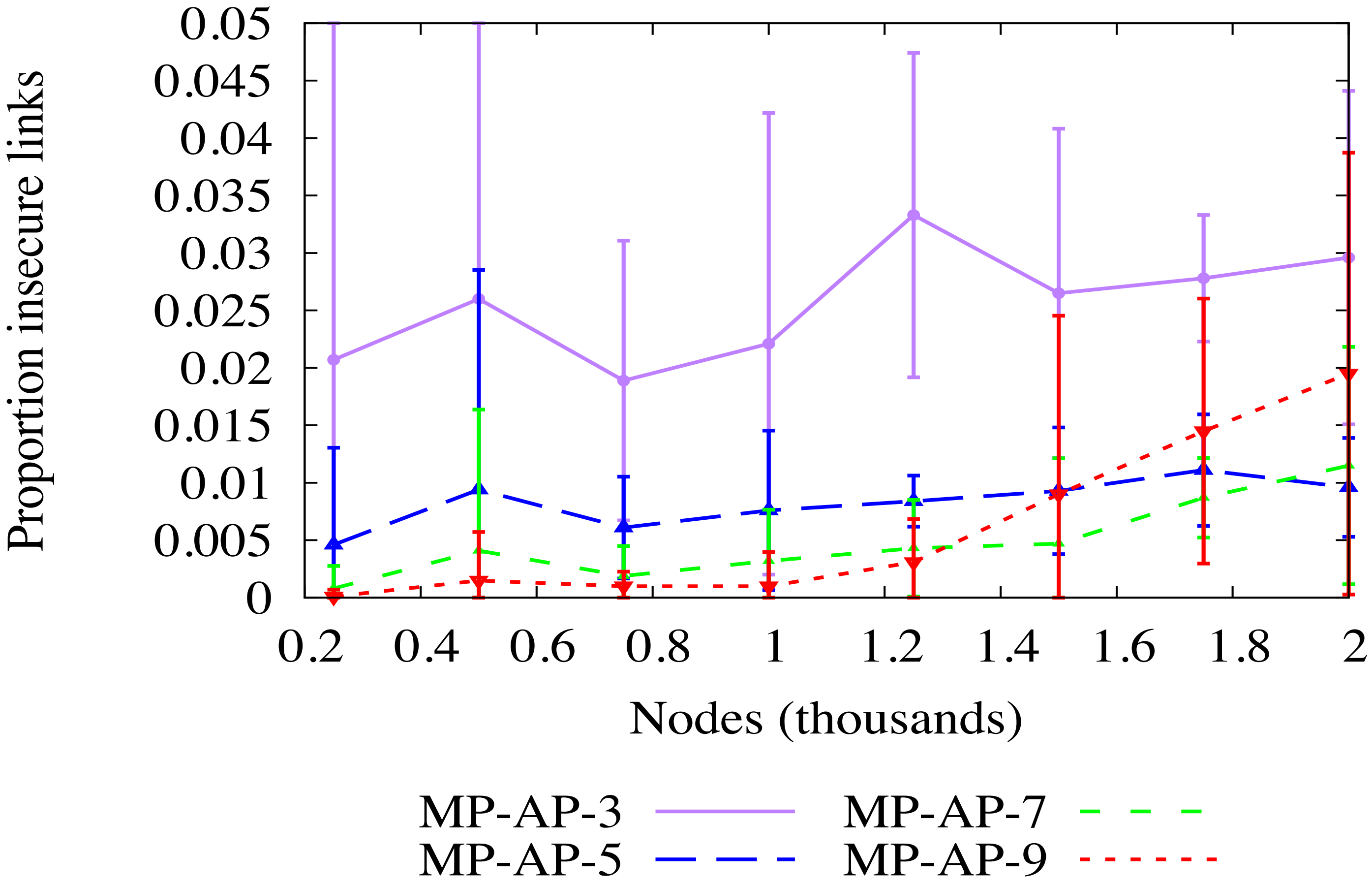
Connected graph: Proportion insecure links



Connected graph: Communication overhead



Multi-path


Proportion insecure links
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Nodes (thousands)
MP-AP-3
MP-AP-5
MP-AP-7
MP-AP-9

Summary

- Network coding brings new challenges and opportunities
- Challenge
  - Defenses against particular types of attacks against network coding: pollution
- Opportunity
  - Design of key management for sensor networks that leverage network coding and multi-path