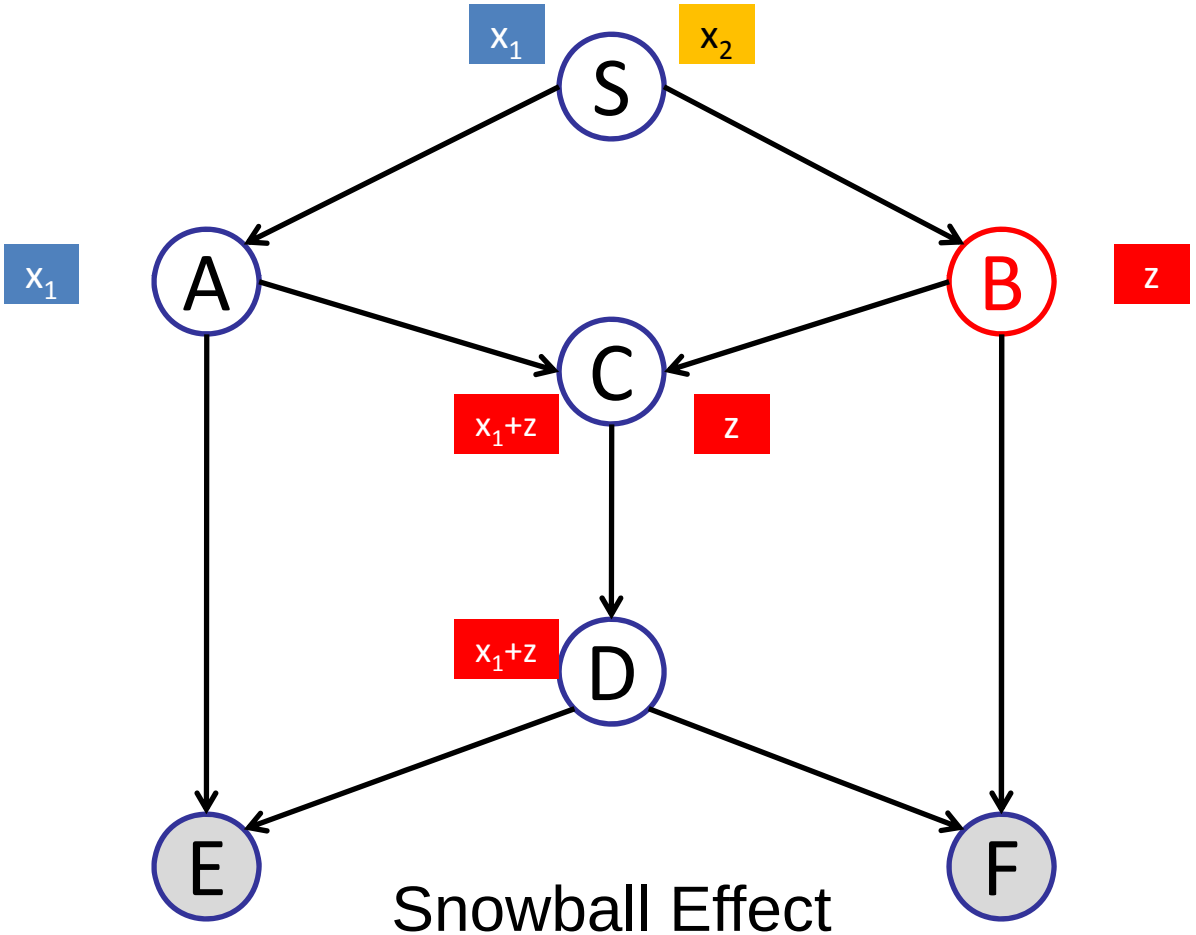
Locating Byzantine Attackers in Intra-Session Network Coding using SpaceMac

Anh Le, Athina Markopoulou
University of California, Irvine

Byzantine (a.k.a. Pollution) Attacks



Prior Byzantine Defense Mechanisms

	Error Correction	Attack Detection	Locating Attackers
Communications	- Error-correcting codes: use redundancy	- Extension of random linear NC - Subspace properties	- Subspace properties
Cryptography		-Homomorphic crypto. primitives: H.Hash, H.Mac, H.Signature	- Probabilistic Non-repudiation protocol

Prior Byzantine Defense Mechanisms

- Error Correction
  [Yeung and Cai, 2006], [Zhang, 2006], [Jaggi et al., 2007]

- Attack Detection
  [Ho et al., 2008], [Kehdi and Li, 2009], [Gkantsidis and Rodriguez, 2007], [Boneh et al., 2009], [Agrawal and Boneh, 2009], [Li et al., 2010]

- Locating Attackers
  [Jafarisiavoshani et al, 2008], [Wang et al., 2010]

Our Proposal

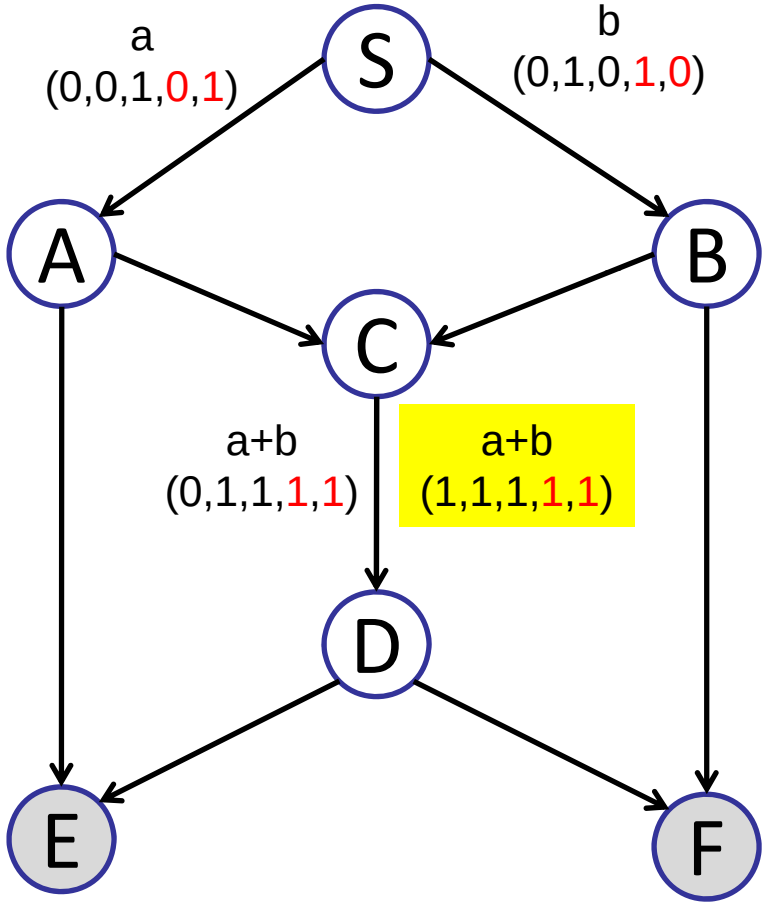
	Error Correction	Attack Detection	Locating Attackers
Communications	-Error-correcting codes: use redundancy	- Extension of random linear NC - Subspace properties (Null keys)	Subspace properties + SpaceMac for expanding spaces + non-repudiation protocol
Cryptography		-Homomorphic crypto. primitives: H.Hash, H.Mac, H.Signature	

Outline

- Background and Motivation
- Prior defense mechanisms
  - Error Correction
  - Attack Detection
  - Locating Attackers
- Our proposal
  - Key Observation
  - SpaceMac
  - Collusion Resistance
  - Evaluation Results
- Concluding Remarks

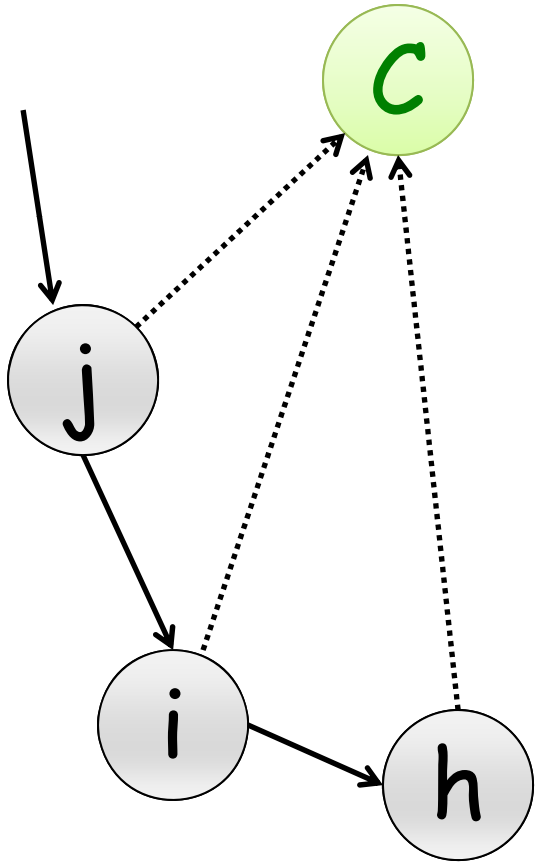
NC & Pollution: Background

- Augmentation
  v | global encoding vector
- Source space
  space spanned by augmented vectors sent by source
- Benign node send vectors belonging to source space
- Pollution attacker sends vectors not in source space

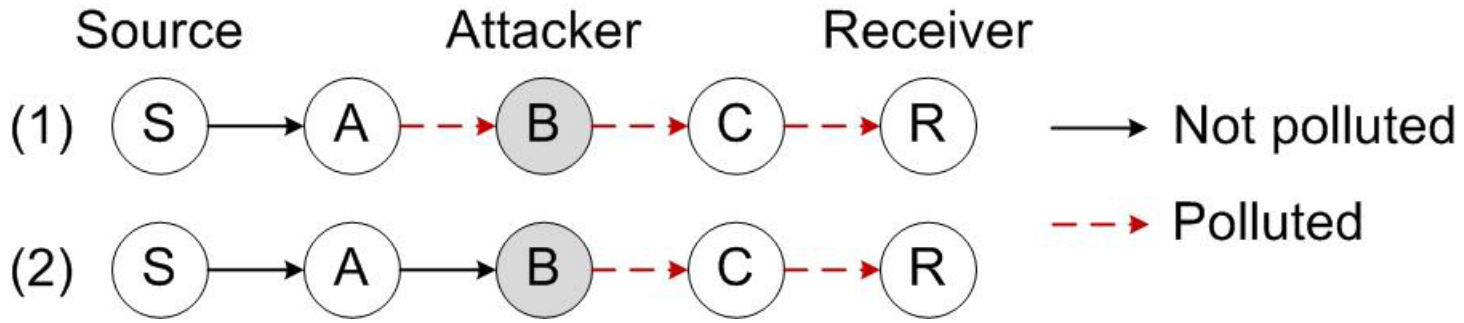


Locating attackers with subspace properties …
(Jafarisiavoshani et al., 2007)

- When a polluted packet is detected:
  1. Each node reports its incoming spaces to a controller
  2. Controller classifies space as polluted or not
  3. Nodes initiating polluted edges are identified as attackers



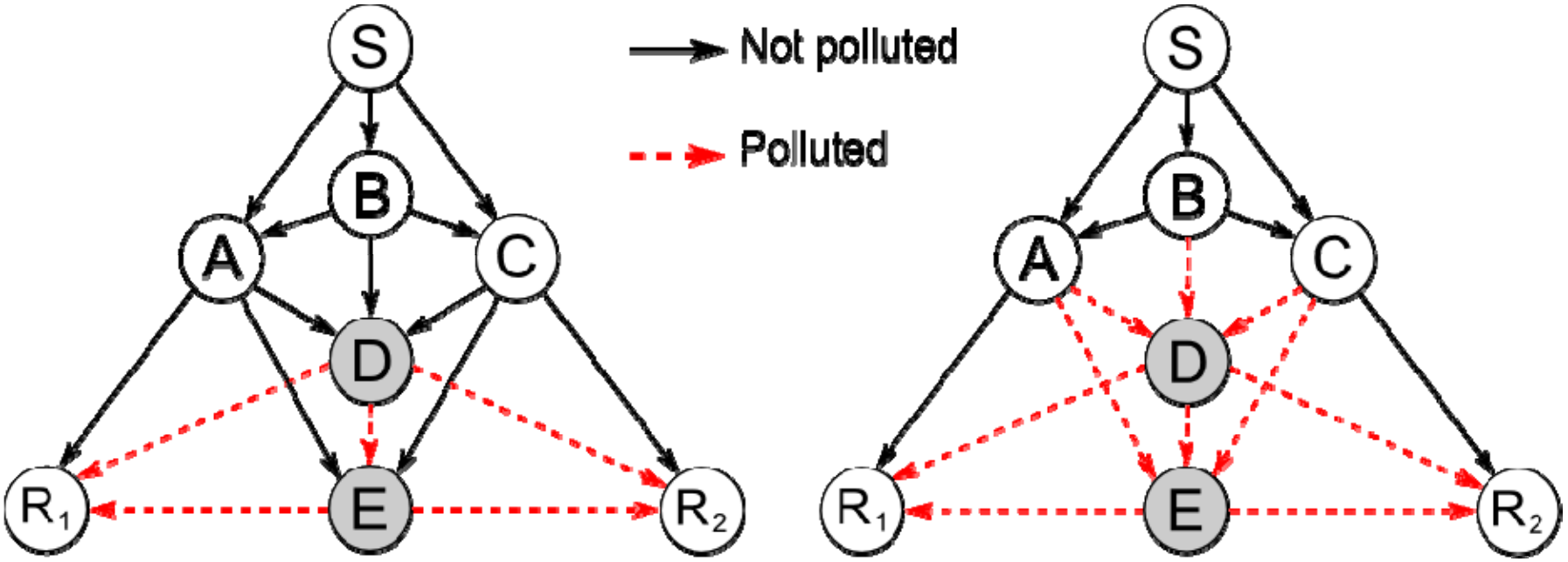
Example



- Scenarios:
  - (1) the attacker lies
  - (2) the attacker is honest

- Result: Attacker could be either A or B

Another Example

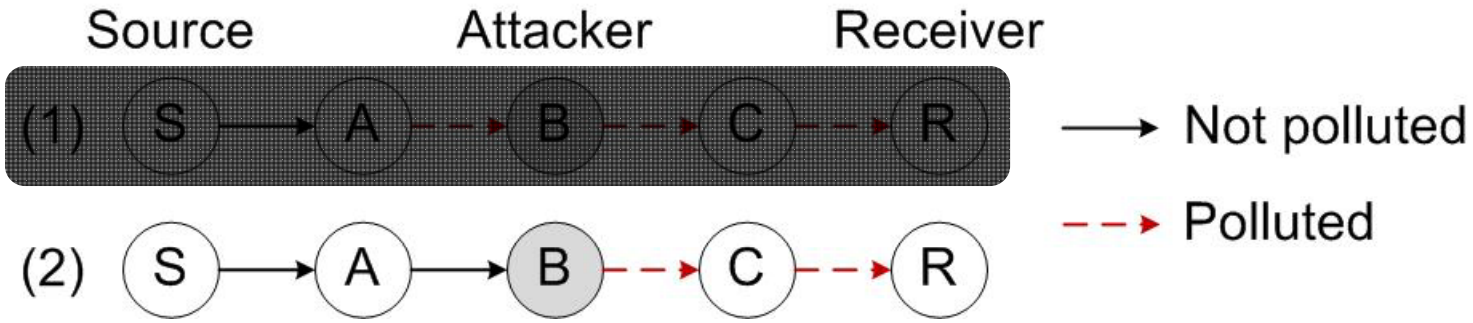


E and D are honest

E and D lie

- Suspected nodes: A, B, C, D, E

Key observation



- If every node cannot lie about its incoming space, …

  … then exact identification is possible

Overview of Our Proposal

- Child reports a random vector of each incoming space

- Use message authentication code (MAC) to prevent child from lying.

1. A malicious child can't compute a valid MAC tag for a vector out of his incoming space
2. A benign child is able to compute a valid MAC tag for any vector in his incoming space

SpaceMac

Our Proposal

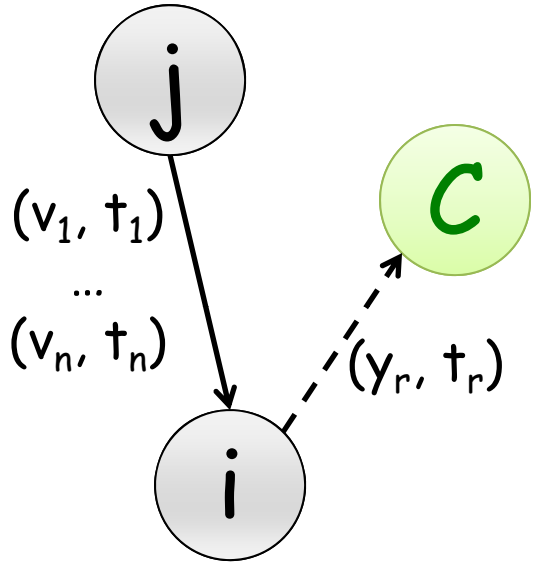
- Assumptions
  - Controller knows topology and source space
  - Reliable channels btw controller and nodes
  - Shared symmetric keys

- Pollution Detection
  - In-network: Homomorphic MAC [HomMac, RIPPLE]

  - At receiver: application specific
    e.g. by corrupted video frame

SpaceMac: Send and Report

- When j sends vectors, it sends SpaceMac tags generated using the shared key between j and the controller C
- When i reports, tag of the random reported vector is computed using tags that j sends
- SpaceMac allows for generating tag of any linear combination of v_i's but not vector out of span(v_i)



$$\mathbf{y}_r = \sum_{i=1}^{n} \alpha_i \, \mathbf{v}_i$$

$$t_r = \sum_{i=1}^{n} \alpha_i \, t_i$$

SpaceMac: Construction

– $\mathsf{Mac}(k, \mathrm{id}, \mathbf{y})$:

(1) $\mathbf{r} \leftarrow (F(k, \mathrm{id}, 1), \cdots, F(k, \mathrm{id}, n+m))$

(2) $t \leftarrow \mathbf{y} \cdot \mathbf{r} \in \mathbb{F}_q$

– $\mathsf{Combine}((\mathbf{y}_1, t_1, \alpha_1), \cdots, (\mathbf{y}_p, t_p, \alpha_p))$:

(1) $t \leftarrow \sum_{i=1}^{p} \alpha_i \, t_i \in \mathbb{F}_q$

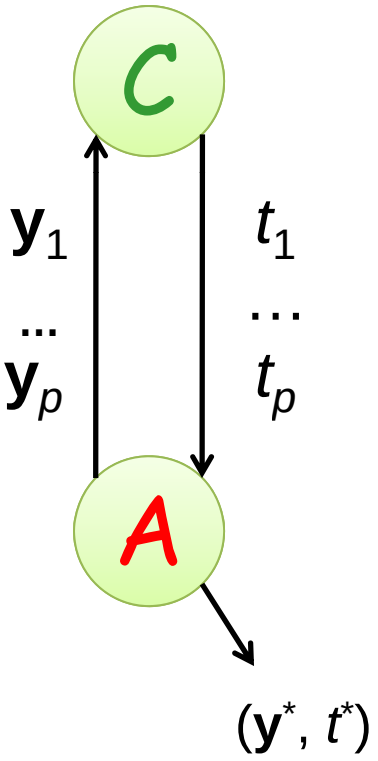
– $\mathsf{Verify}(k, \mathrm{id}, \mathbf{z}, t)$:

(1) $\mathbf{r} \leftarrow (F(k, \mathrm{id}, 1), \cdots, F(k, \mathrm{id}, n+m))$

(2) $t' \leftarrow \mathbf{z} \cdot \mathbf{r}$

(3) If $t' = t$, output 1; otherwise, output 0

SpaceMac: Attack Game



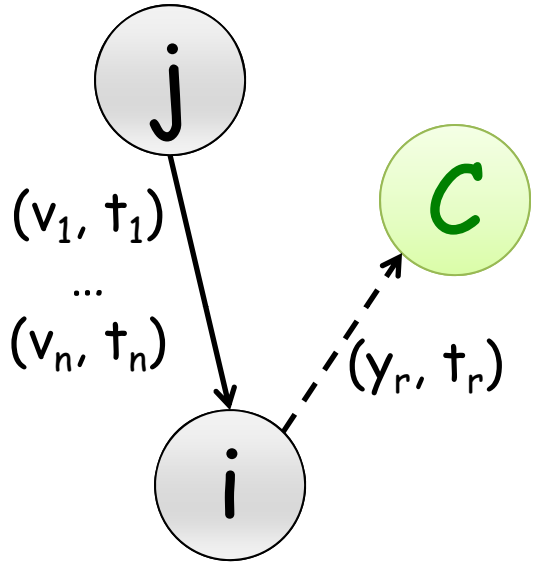
- Adversary wins if:

$$\begin{aligned} &\text{(i)} \quad \mathbf{y}^* \neq \mathbf{0} \\ &\text{(ii)} \quad \mathbf{y}^* \notin \text{span}(\mathbf{y}_1, \cdots, \mathbf{y}_p) \\ &\text{(iii)} \quad \text{Verify}(k, \mathbf{y}^*, t^*) = 1 \end{aligned}$$

- Results:
  Adversary wins with prob at most *1/q*

Expanding Space

- Note that span(v_i) expands over time



$$\mathbf{y}_r = \sum_{i=1}^{n} \alpha_i \, \mathbf{v}_i$$

$$t_r = \sum_{i=1}^{n} \alpha_i \, t_i$$

Related Work: Agrawal and Boneh' HomMac

– Sign(k, id, $\mathbf{v}$, i): To generate a tag for an ith basis vector $\mathbf{v} \in \mathbb{F}_q^{n+m}$ using key $k = (k_1, k_2)$ do:

(1) $\mathbf{u} \leftarrow G(k_1) \in \mathbb{F}_q^{n+m}$

(2) $b \leftarrow F(k_2, (\mathsf{id}, i)) \in \mathbb{F}_q$

(3) $t \leftarrow (\mathbf{u} \cdot \mathbf{v}) + b \in \mathbb{F}_q$

– Combine$((\mathbf{v}_1, t_1, \alpha_1), \cdots, (\mathbf{v}_m, t_m, \alpha_m))$:

(1) $t \leftarrow \sum_{j=1}^{m} \alpha_j t_j \in \mathbb{F}_q$,

– Verify$(k, \mathsf{id}, \mathbf{v}, t)$: Let $\mathbf{v} = (v^{(1)}, \cdots, v^{(n+m)})$, do:

(1) $\mathbf{u} \leftarrow G(k_1) \in \mathbb{F}_q^{n+m}$

(2) $b \leftarrow \sum_{j=1}^{m} \left[v^{(n+j)} \cdot F(k_2, (\mathsf{id}, j)) \right] \in \mathbb{F}_q$

(3) $a \leftarrow (\mathbf{u} \cdot \mathbf{v}) \in \mathbb{F}_q$

(4) If $a + b = t$ output 1; otherwise, output 0

Related Work: RIPPLE [Li et. al, 2010]

- Inner product MAC

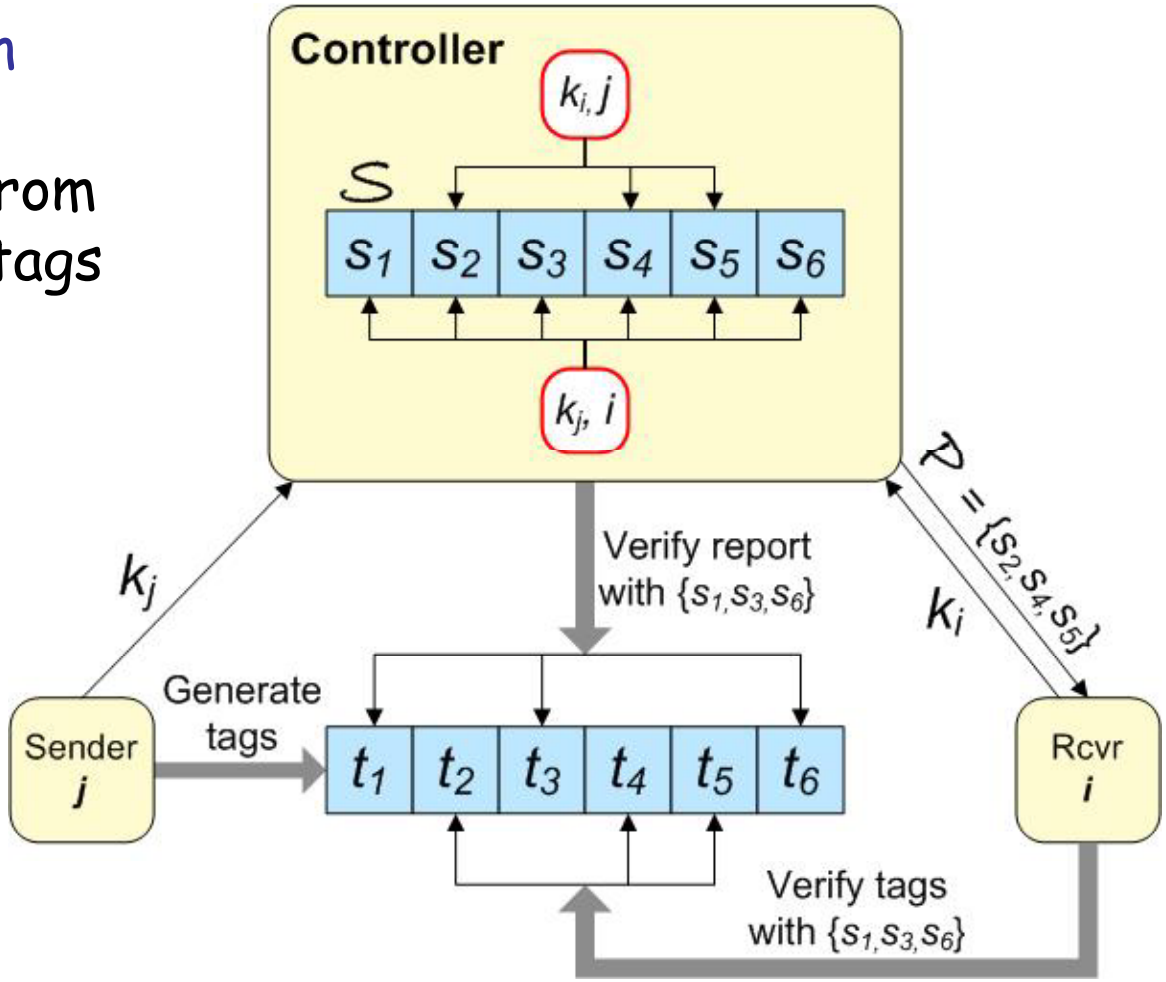
  - **Generate:** Sample $K \xleftarrow{R} \mathbb{F}_q^{n+m}$.
  - **MAC:** Given $M, K \in \mathbb{F}_q^{n+m}$, output $t = \langle M, K \rangle \in \mathbb{F}_q$.
  - **Verify:** Given (K, M, t), check that $\langle M, K \rangle = t$.
  - **Combine:** Given $(M_i, t_i, \alpha_i)_{i=1}^{w}$ with $d \leq m$, output $\sum_{i=1}^{w} \alpha_i t_i$.

- Support nested MACs

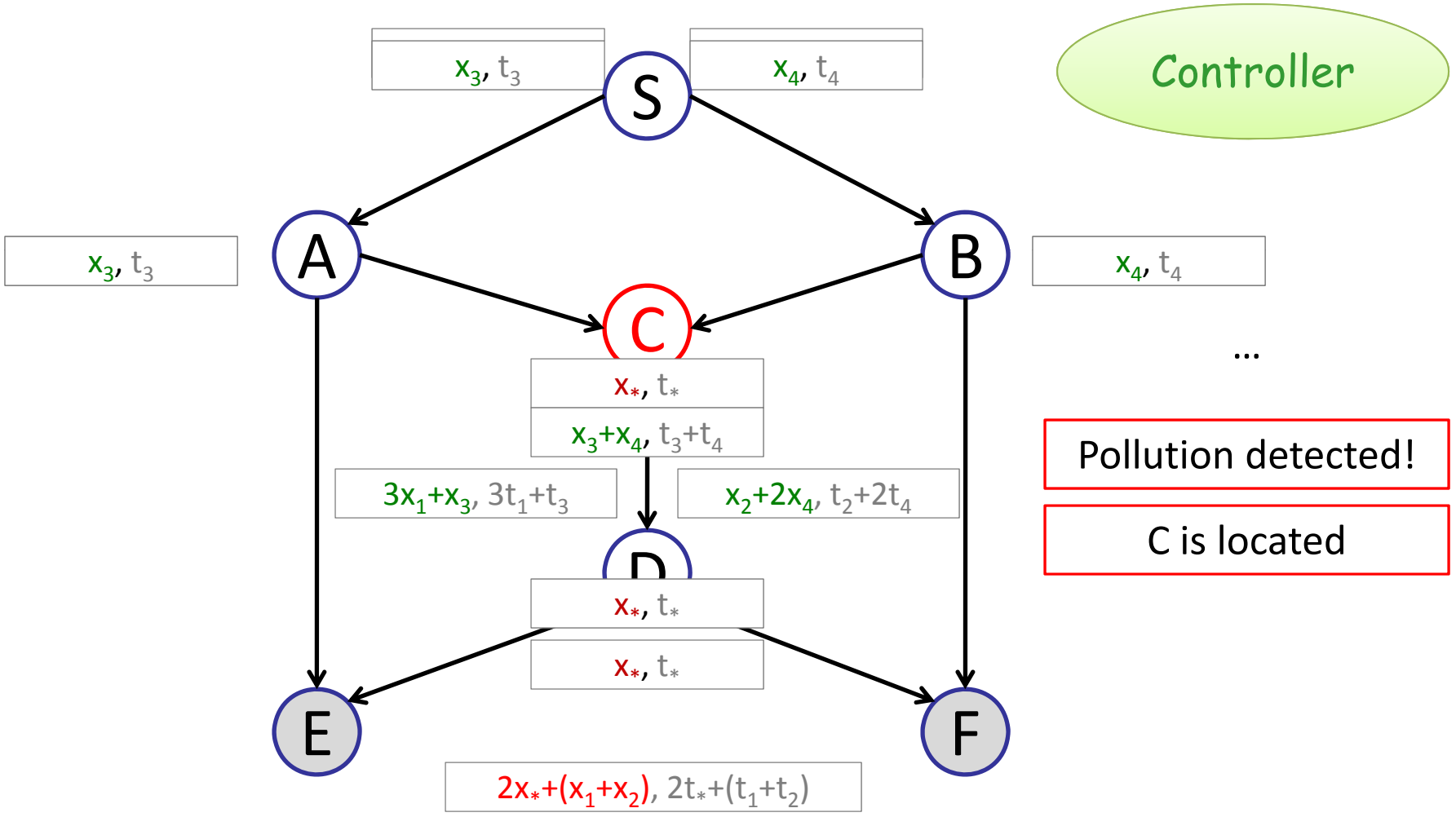
- Focus on in-network detection

To prevent parents from lying … (Wang et al., 2010)

- Non-repudiation protocol:
  - to prevent j from sending invalid tags

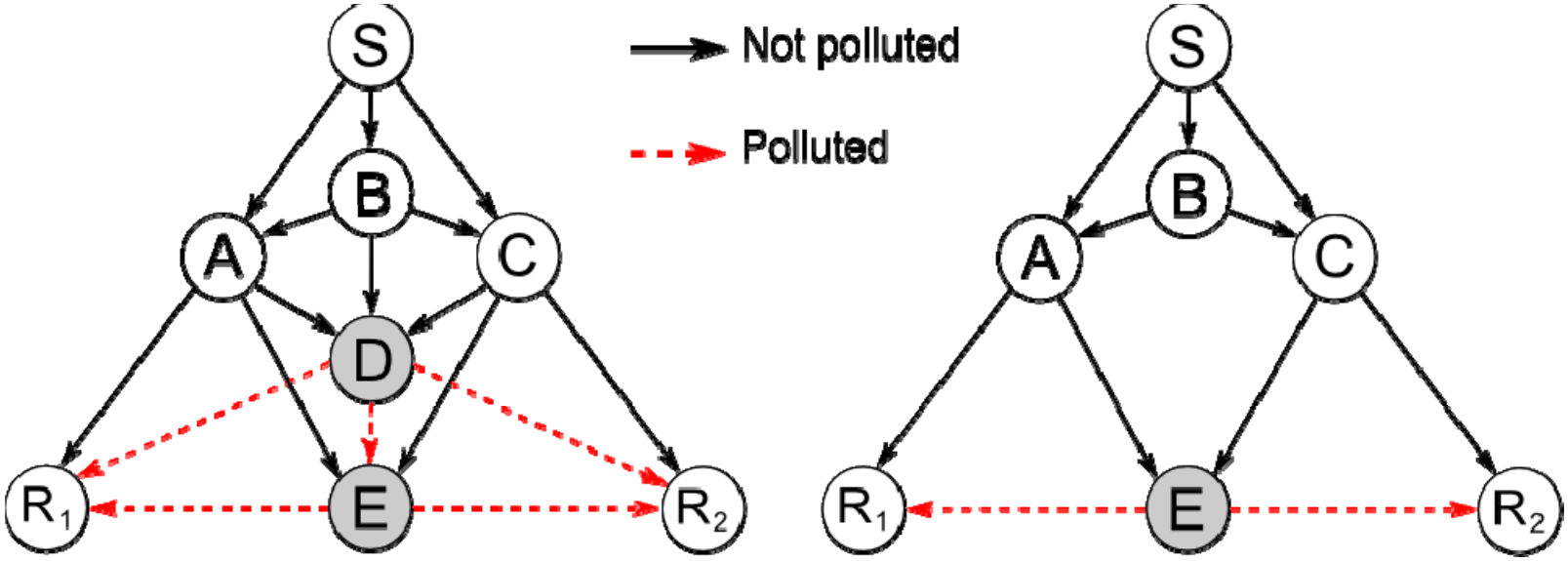


SpaceMac: Illustrated



Locating Attackers

In a network with M attackers, with high probability, all attackers can be identified after N generations which experience pollution attack, where $N \leq M$.

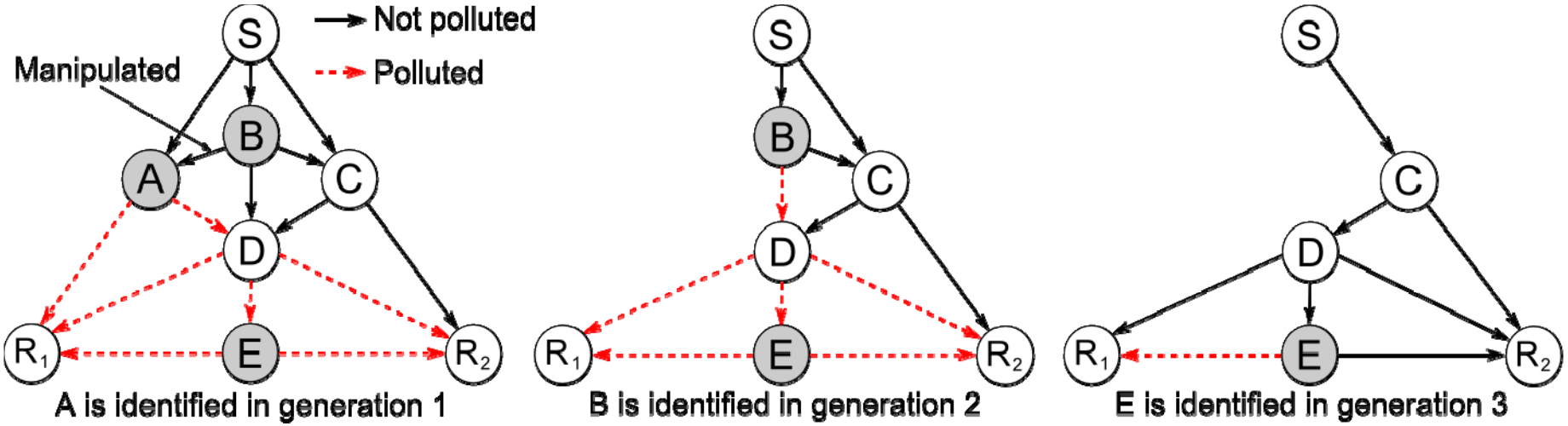


D is identified in generation 1

E is identified in generation 2

Collusion Resistance

Collusion affects the order in which the attackers are identified.



Performance Evaluation

- Communication Overhead:

Prob. Child blames Parent	Prob. Parent tricks Child	Overhead (1 byte per tag)
2^{-14}	2^{-16}	25 bytes
2^{-16}	2^{-21}	30 bytes

- Computation Overhead (per tag):

Parameters	Mac	Verify	Combine
$q=2^8$, m=5, m+n=1024	<1000 μs	<1000 μs	<1 μs

- Locating latency:

Number of attackers	12	16	20
Average number of generations	3.85	4.69	4.89

Conclusion

	Error Correction	Attack Detection	Locating Attackers
Comm.	-Error-correcting codes	- Extension of random linear NC - Null Keys	- Subspace properties
			- SpaceMac
Crypto.		- Homomorphic cryptographic primitives: H.Hash, H.Mac, H.Signature	- Non-repudiation protocol

(+) Exactly locating all attackers

(+) Low computation and communication overhead

(+) Can deal with large collusion

- Questions