Distillation Codes and DoS Resistant Multicast Authentication (NDSS 2004)

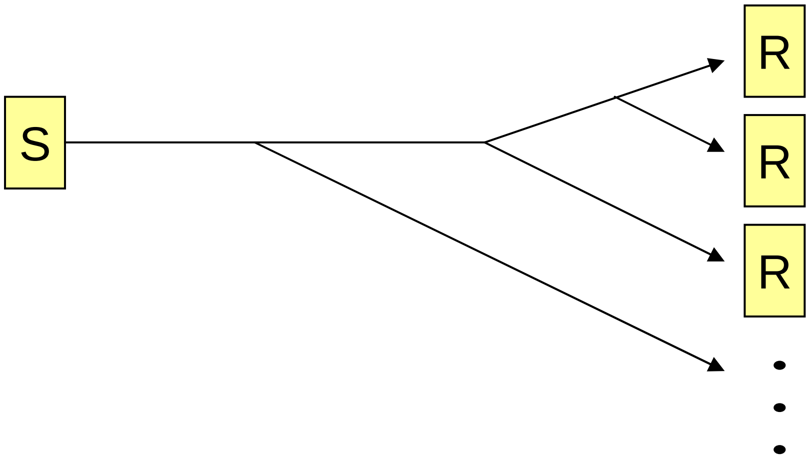
Chris Karlof, Naveen Sastry @ UC Berkeley
Yaping Li, J. D. Tygar @ UC Berkeley
Adrian Perrig @ CMU

2006.04.19
Presenter : JeeHyun Hwang

Acknowledgement : Slides were originally provided by Chris Karlof

Multicast authentication problem

- Security goals
  - Packet authenticity
  - DoS resistance
- Threat model
  - Injection
  - Modification
  - Dropping
  - Eavesdropping



Two multicast authentication protocols

- Two signature amortization schemes based on erasure codes
  - SAIDA (Oakland 2002)

    Signature Amortization using the Information Dispersal Algorithm
  - Pannetrat-Molva (NDSS 2003)

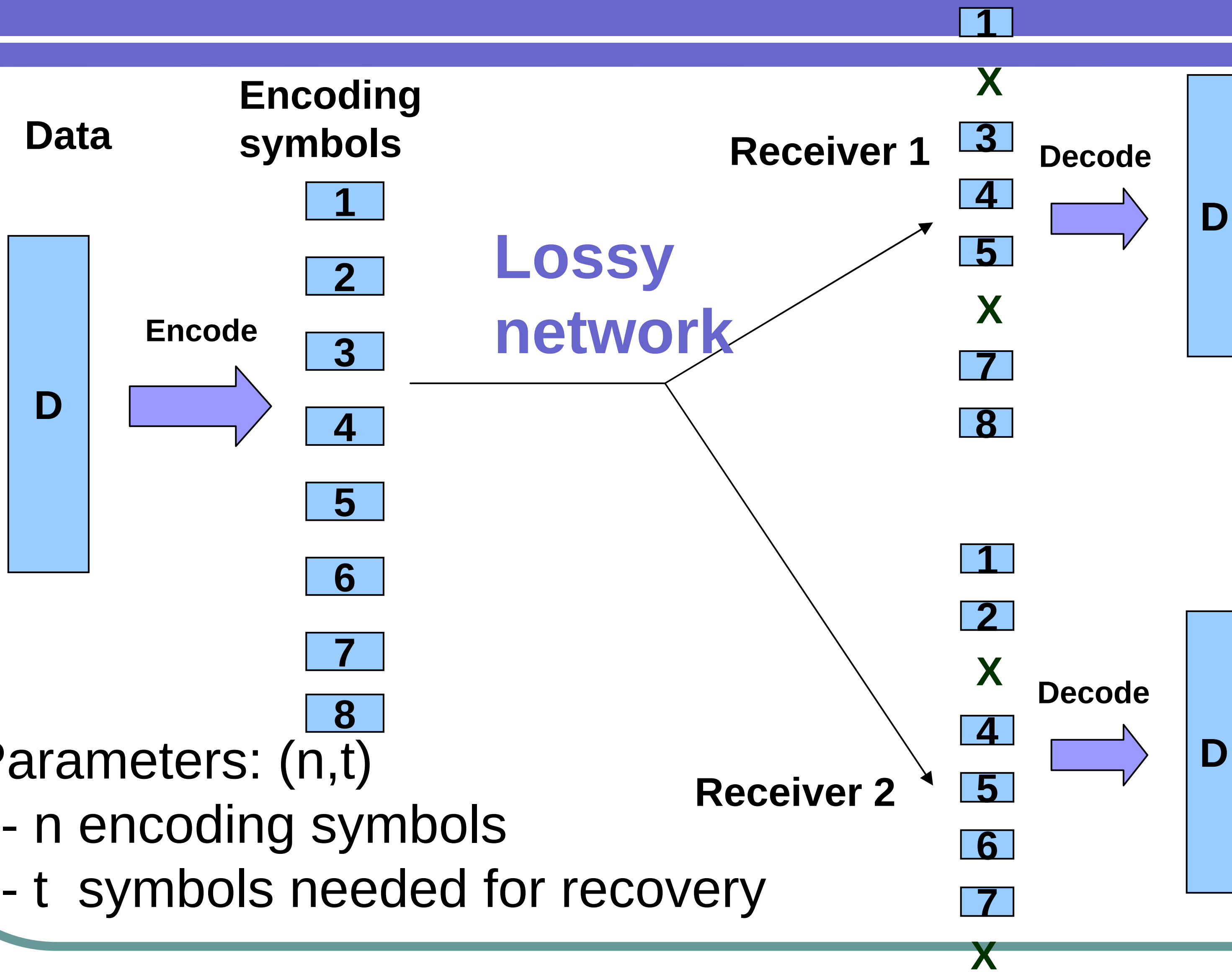
- Properties of SAIDA and Pannetrat-Molva
  - Guarantee authenticity of stream
  - Low overhead (12-23 bytes per packet)
  - Robust to packet loss

- Problem: receivers vulnerable to DoS caused by injection attacks

Outline

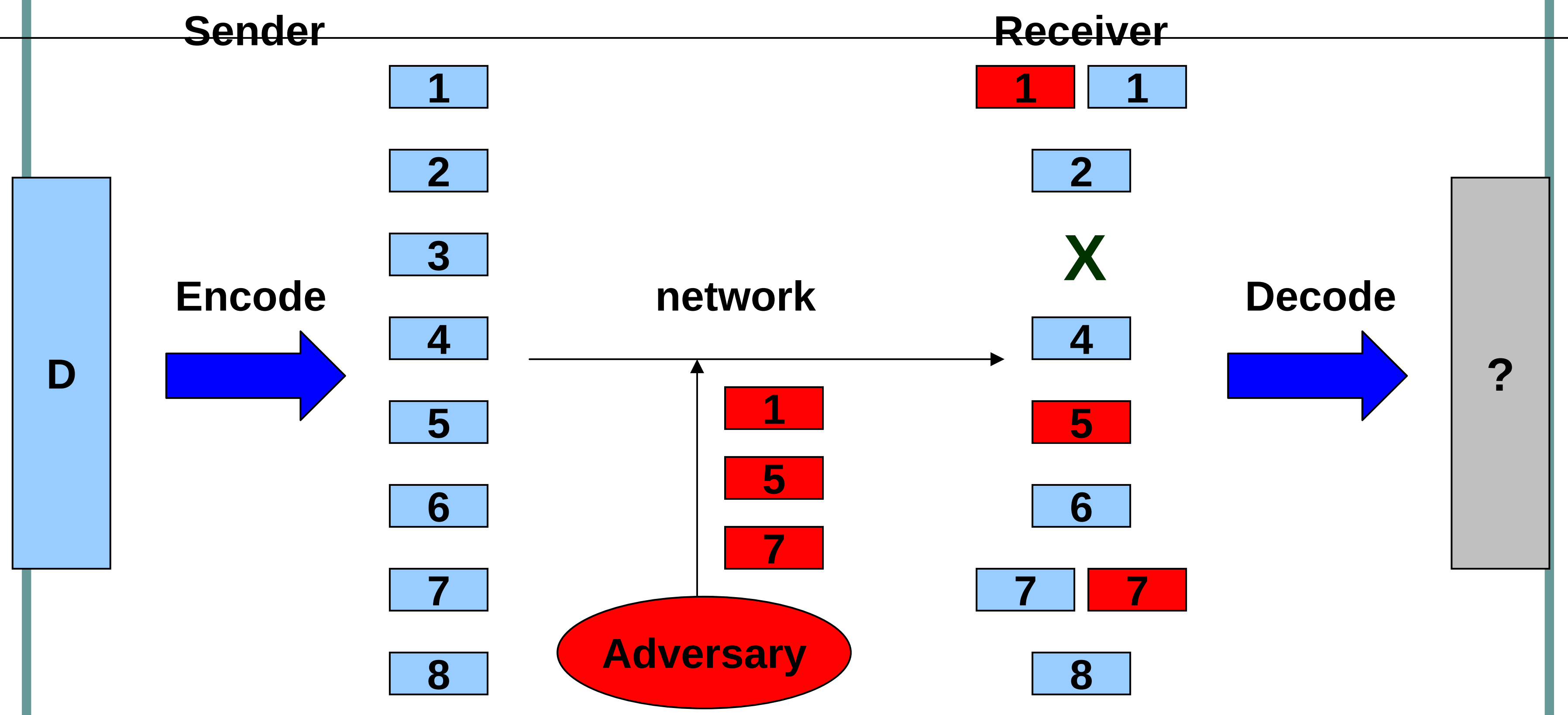
- Erasure codes & pollution attacks
- Our solution: Distillation codes
- Distillation codes for multicast authentication

Erasure codes



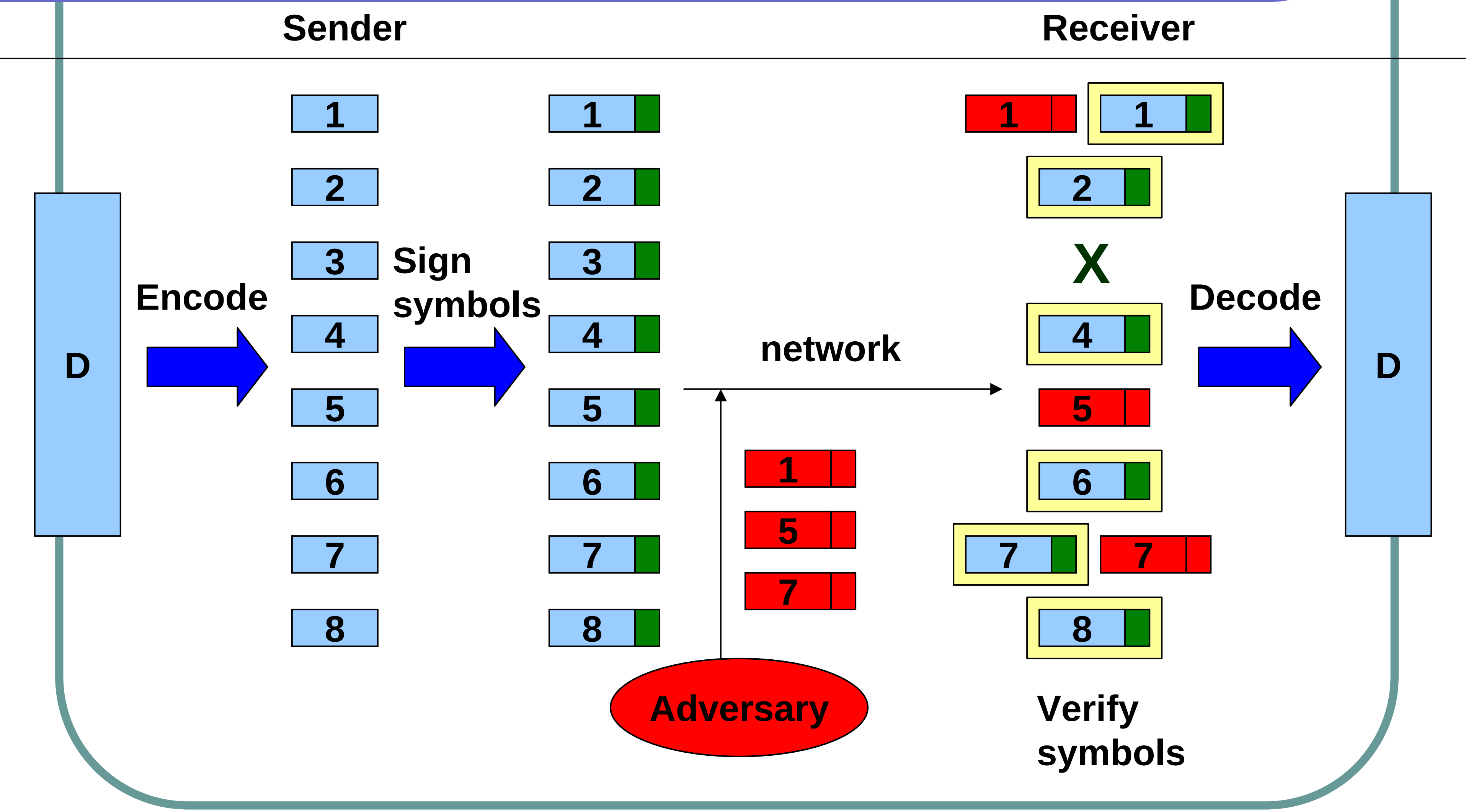
Parameters: (n,t)
- n encoding symbols
- t symbols needed for recovery

Pollution attacks



Decoding with invalid symbols results in an error

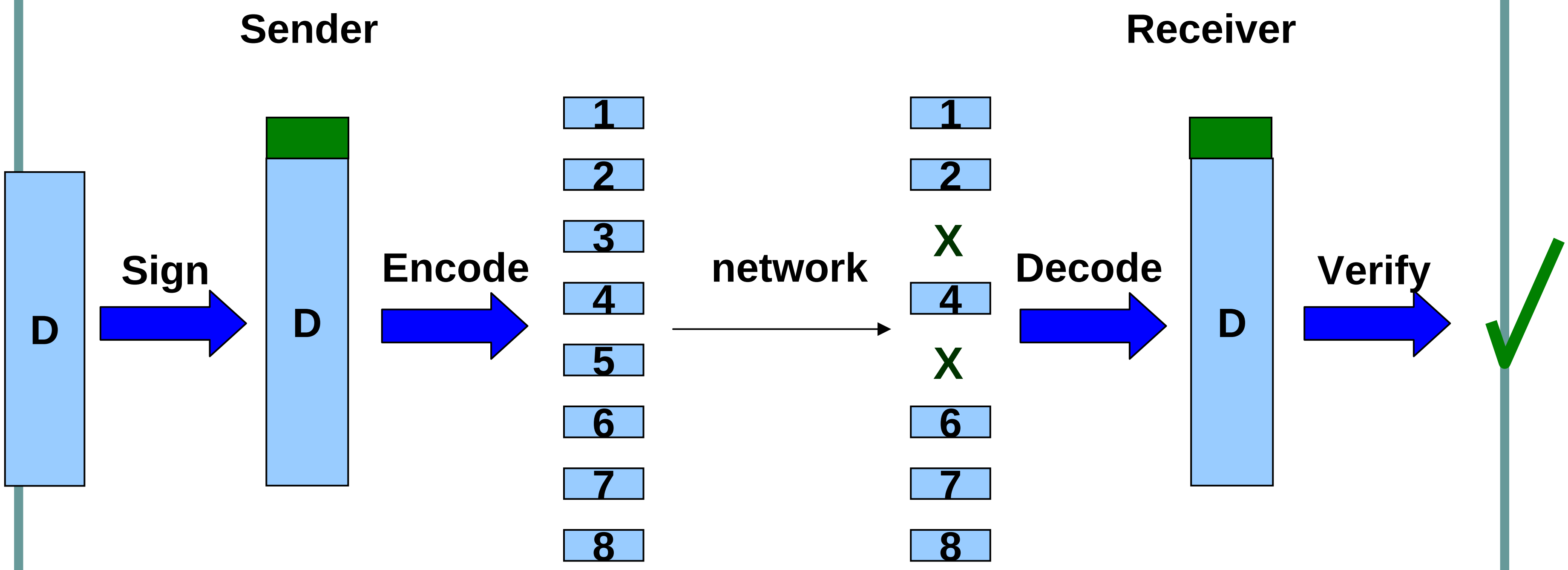
Naive solution: Sign every symbol



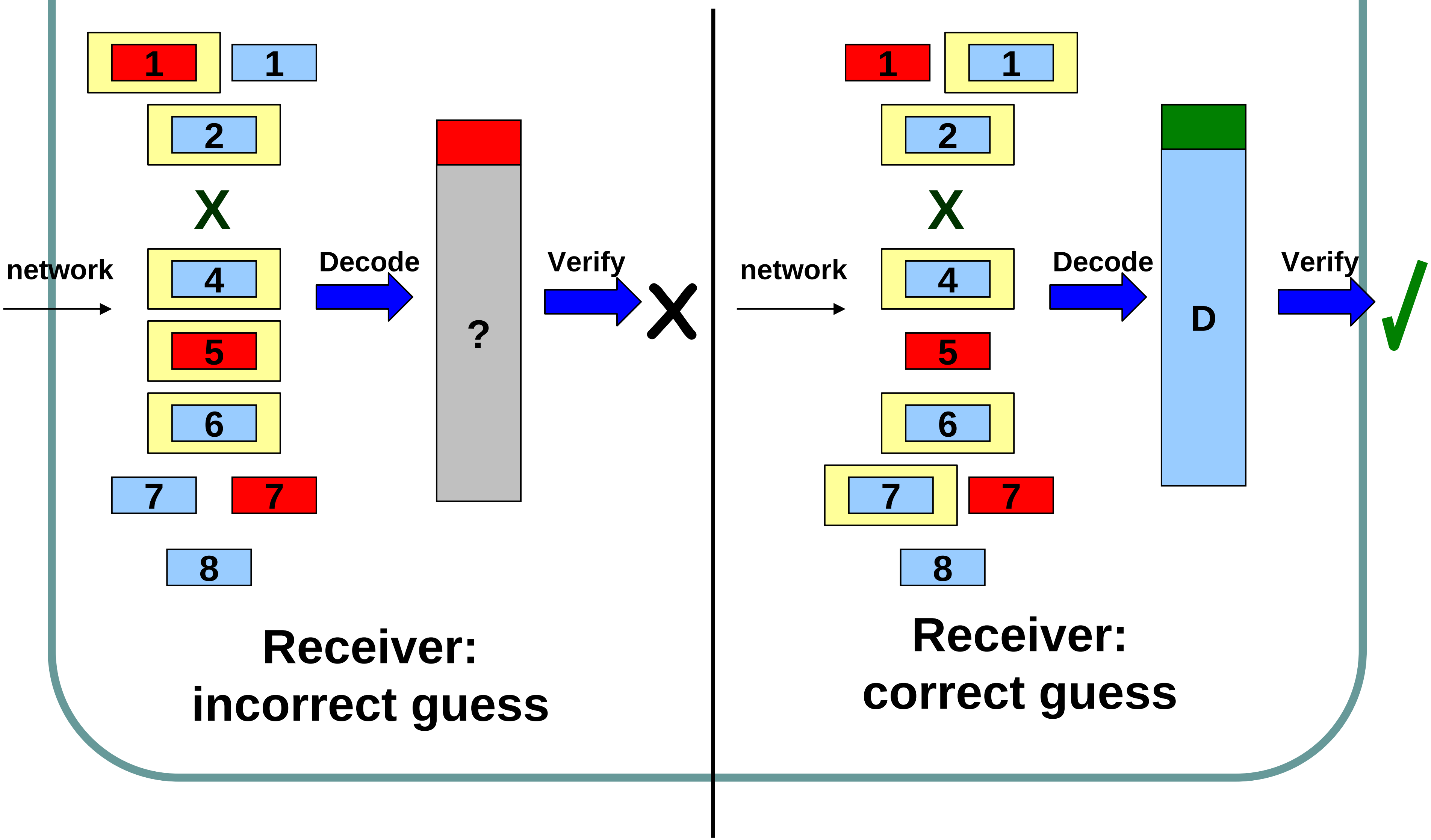
Signatures are expensive

- Generation must be cheap (real time streams)
- Verification must be cheap (real time streams, DoS)
- Overhead must be small (tens of bytes)
- No known signature satisfies all these requirements

Naive alternative: Amortize signature



Guess and check



Guess and check: analysis

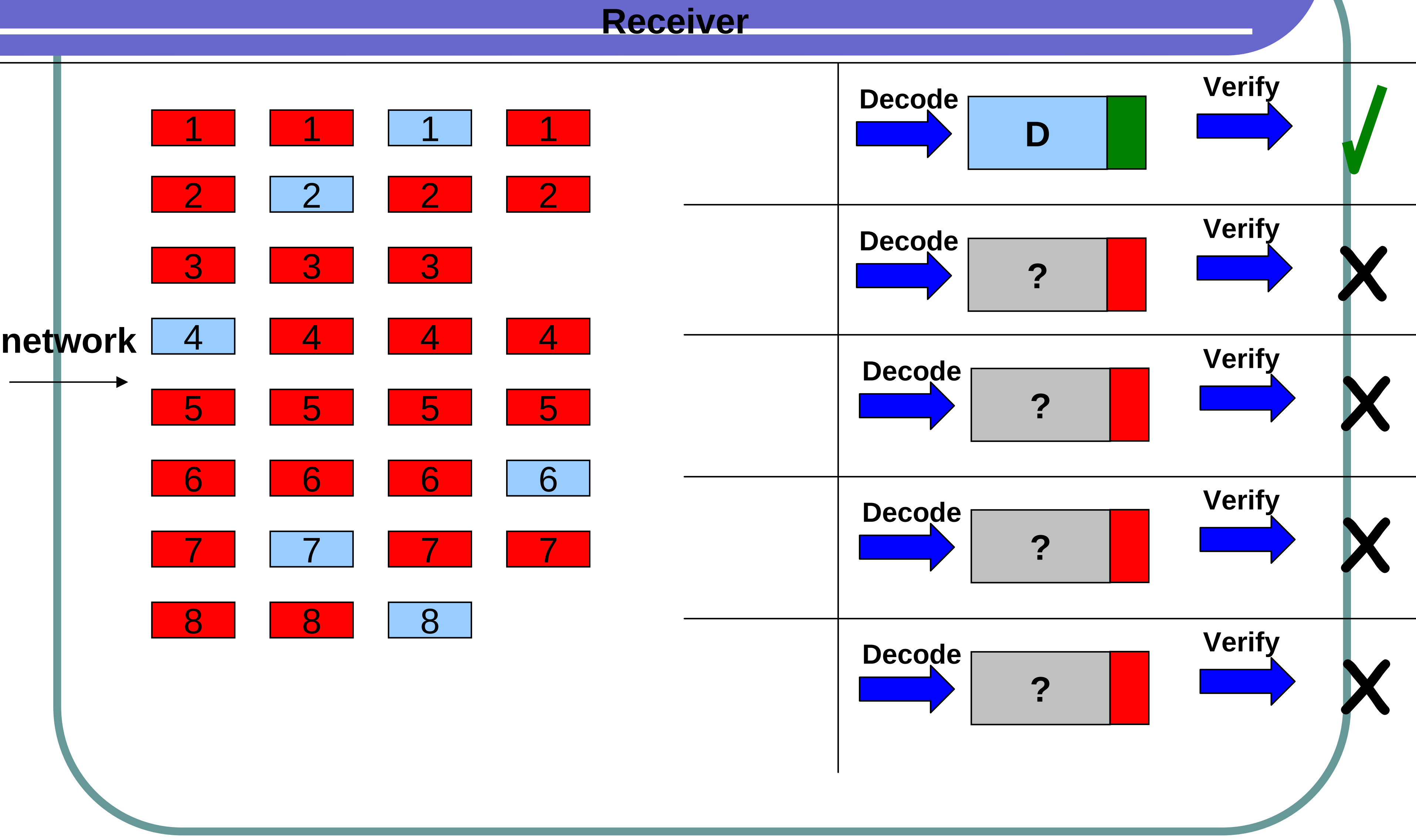
- *Attack factor* f = $\frac{\text{attack traffic}}{\text{valid traffic}}$
  - Attack factor 5 : 50 packets are injected per legitimate 10 packets.
- $O((f+1)^t)$ erasure decodings and verifications needed to reconstruct the valid data
- (n,t) = (128,64), f = 1 → at least 2^{64} operations

Separating good from bad is difficult

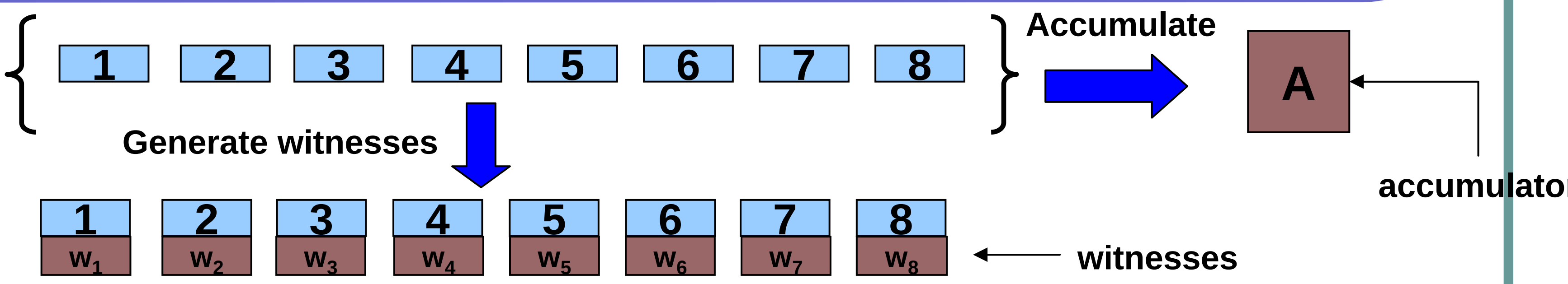
Outline

- Erasure codes & pollution attacks
- Our solution: Distillation codes
- Distillation codes for multicast authentication

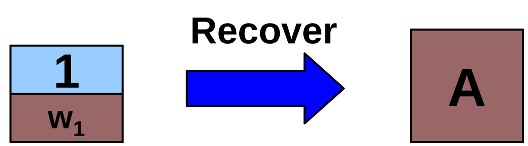
Distillation Codes: Overview



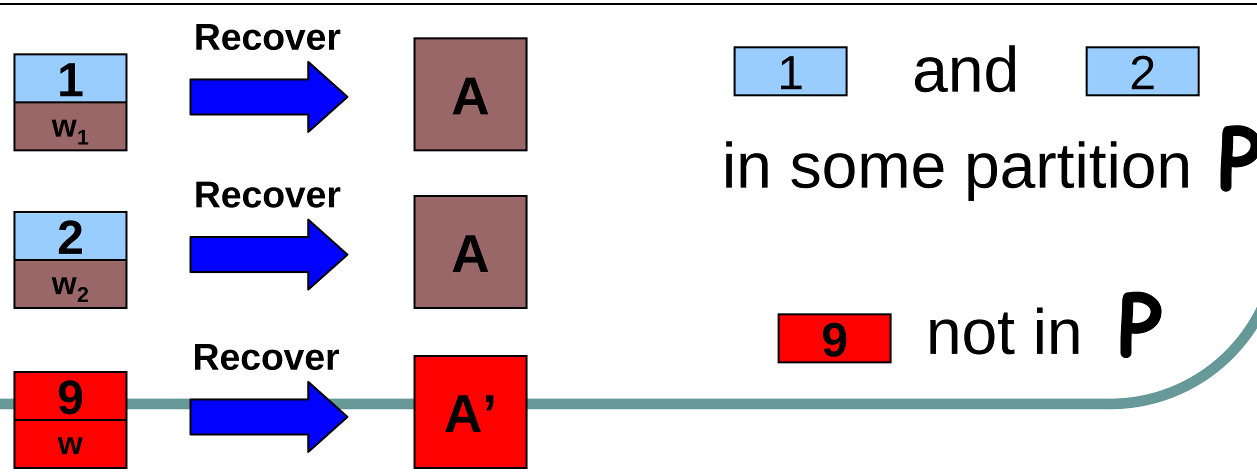
Accumulators



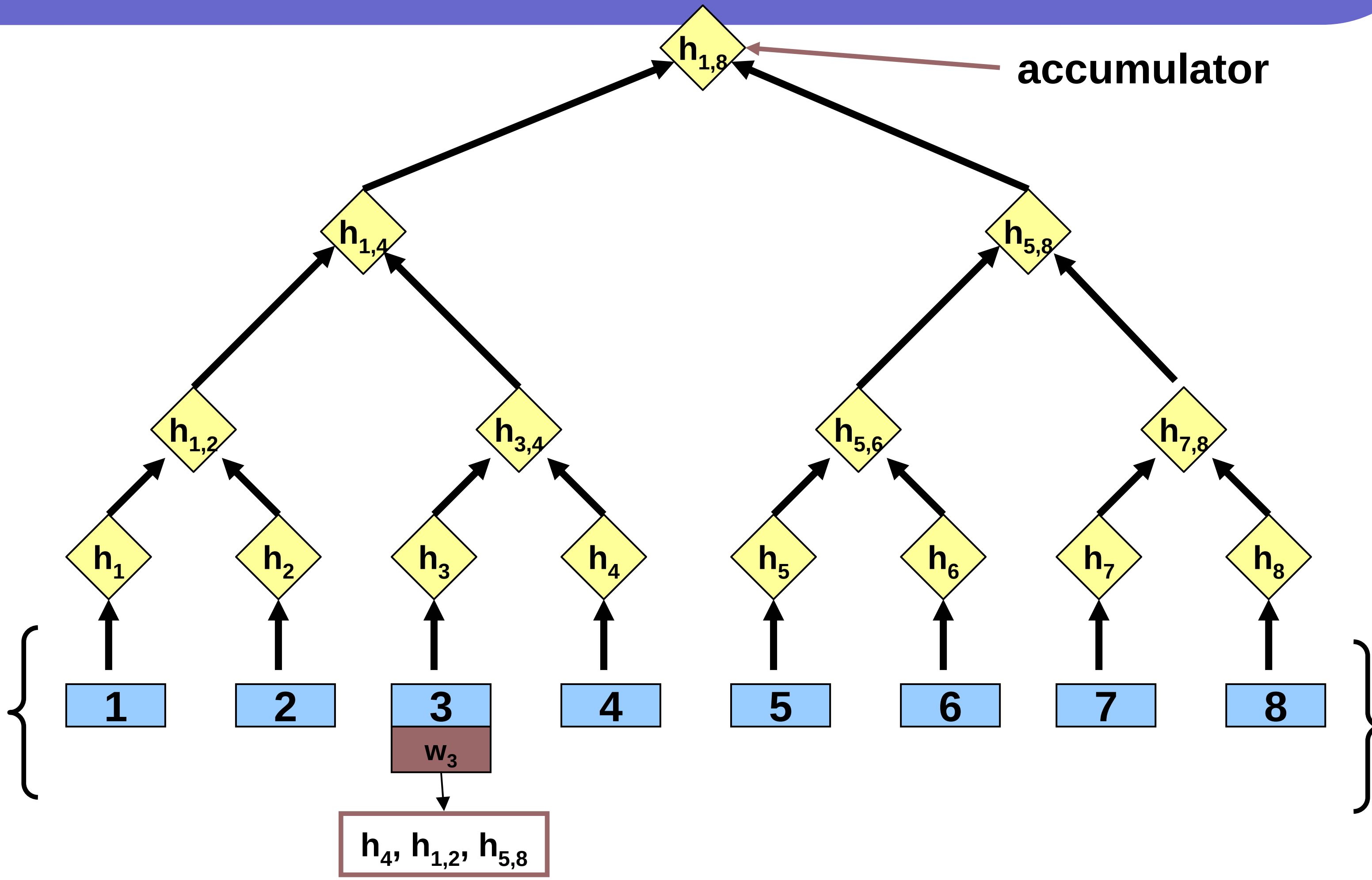
Recover operation



Accumulators for partitioning

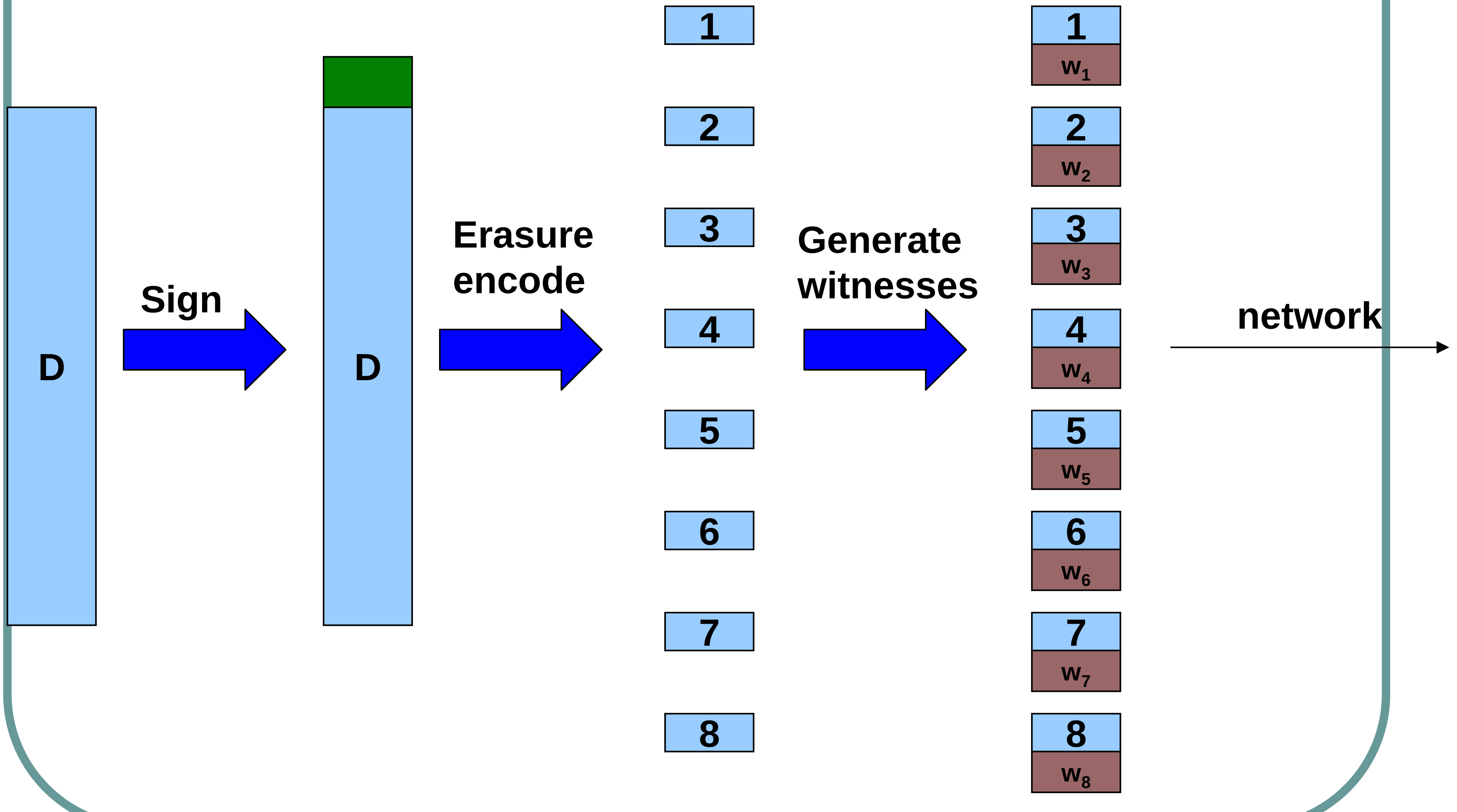


A fast accumulator using Merkle Hash Trees

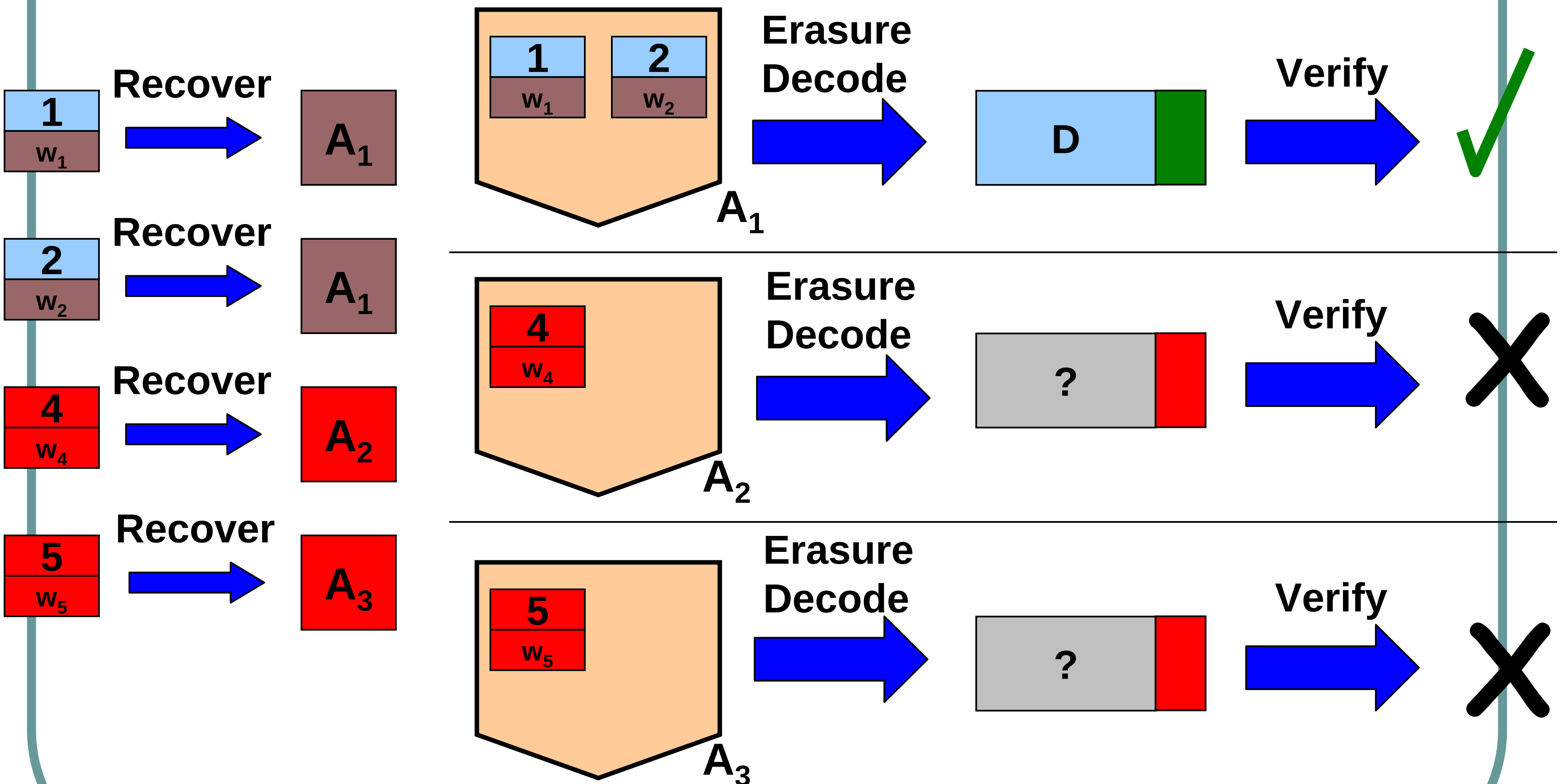


- Set accumulator value: root
- Witness: siblings on root to leaf path. Size= O(log n)

Distillation Codes: Encoding



Distillation Codes: Decoding



Distillation code performance

	Erasure codes (with signatures)	Distillation Codes
Encoding	1 erasure encoding 1 signature generation	1 erasure encoding 1 signature generation 2n hashes
Decoding attack factor f	$O(f^{n})$ erasure decodings $O(f^{n})$ signature verifications	2f+1 erasure decodings 2f+1 signature verifications $(f+1) \cdot n \cdot \log(n)$ hashes

Assume (n,n/2) erasure code

Outline

- Erasure codes & pollution attacks
- Our solution: Distillation codes
- Distillation codes for multicast authentication

Multicast authentication using SAIDA

- Given packets: p_1 p_2 p_3 p_4 • • • p_n

  signature string S =

  $$h(p_1) \,||\, h(p_2) \,||\, h(p_3) \,||\, h(p_4) \,||\, \cdots \,||\, h(p_n) \,||\, Sign(\quad)$$

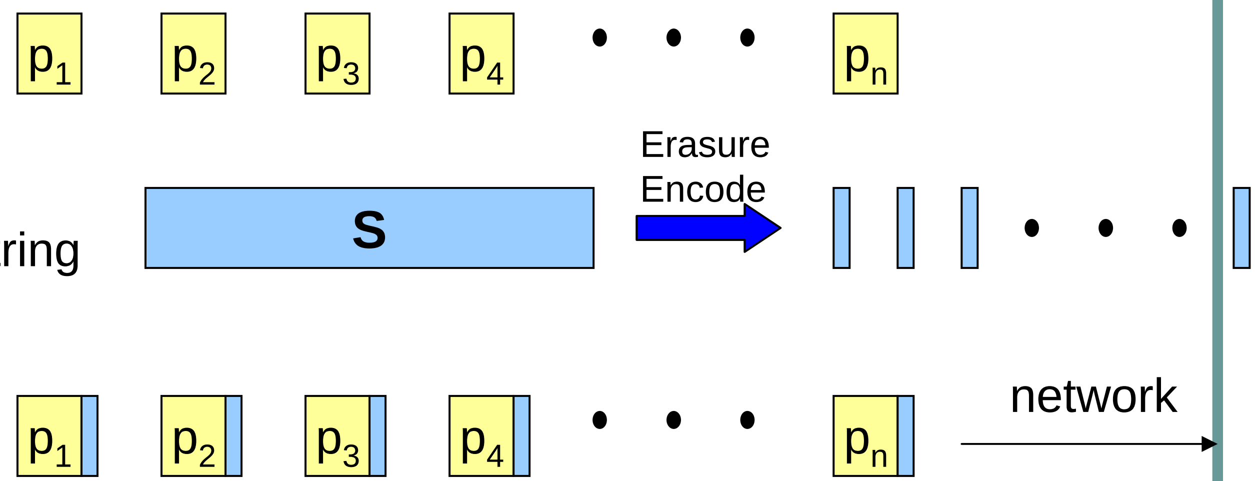
- Receiver can authenticate any packet with S
- How to deliver signature string?
  - Can’t send S with each packet: too big
  - Amortize cost by including piece with each packet

Multicast authentication using SAIDA

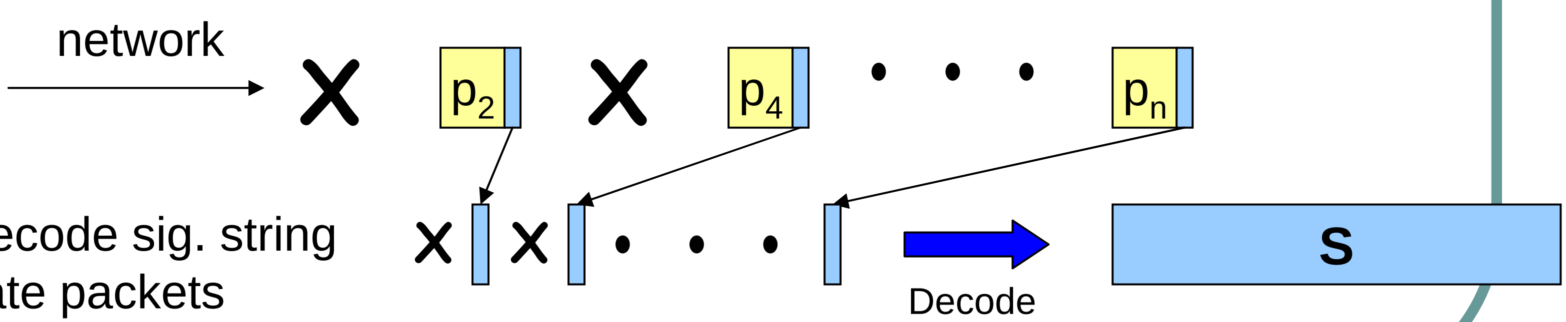
Sender:

1) Break stream into
 blocks of n packets

2) Compute a signature string
 and erasure encode



Receiver:



1) Erasure decode sig. string
2) Authenticate packets

Recall pollution attack: can’t reconstruct signature string

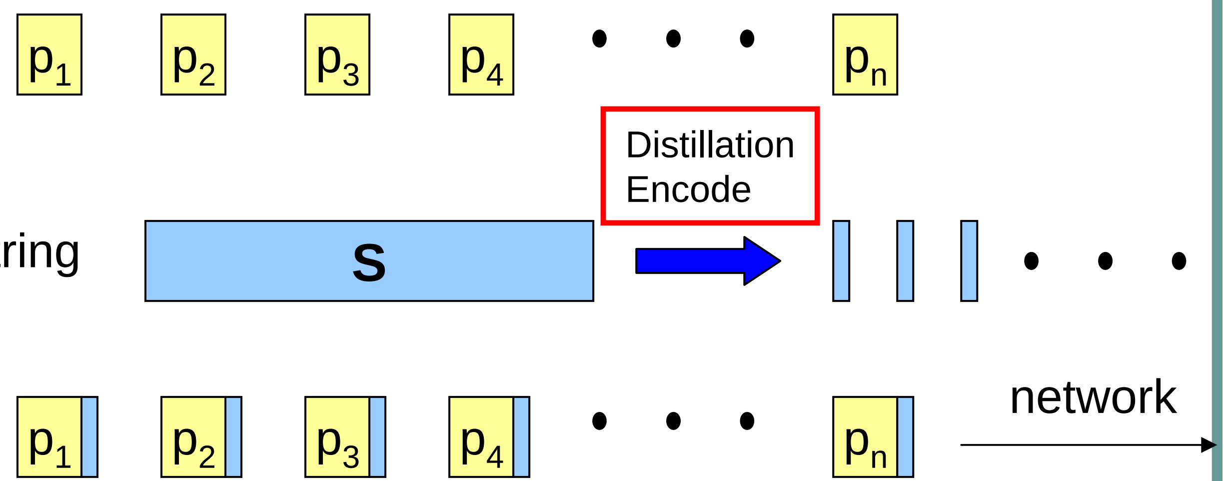
Distillation codes fix SAIDA

Sender:

Use distillation codes instead of erasure codes

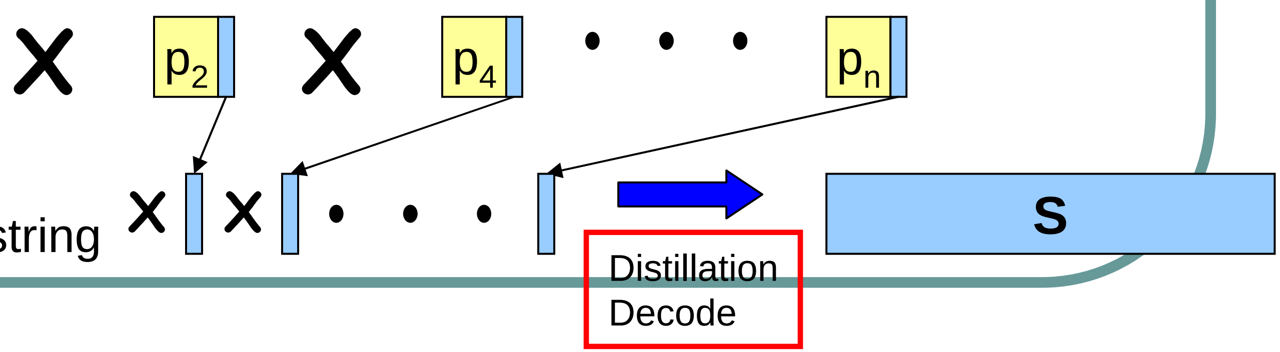
1) Break stream into
   blocks of n packets

2) Compute a signature string
   and distillation encode



Receiver:

network



1) Distillation decode sig. string
2) Authenticate packets

Analysis

- Packet authenticity: block signature
- Computational DoS resistance: distillation codes
- Tolerant to loss: distillation codes

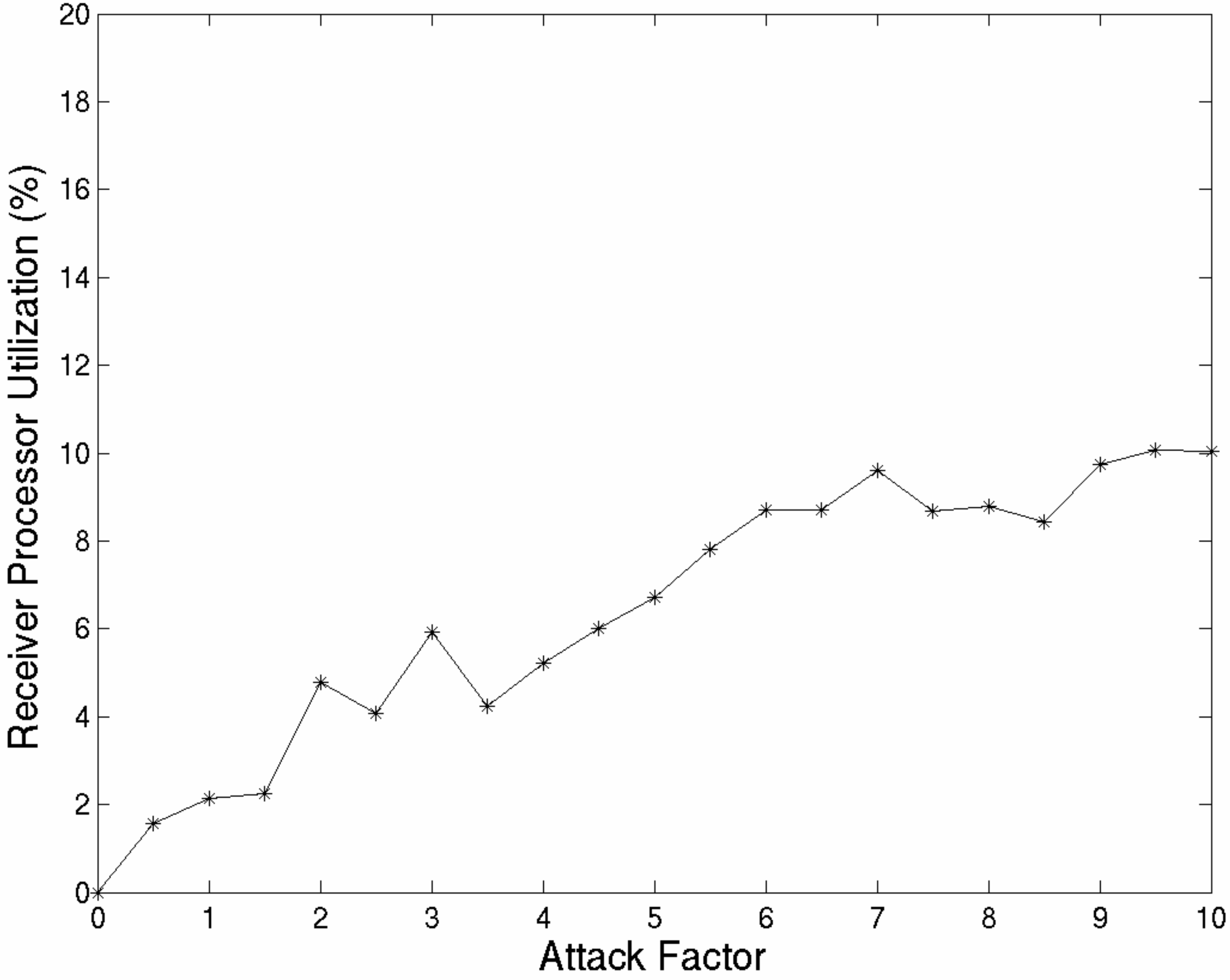
- See paper for proofs of security
- Also applicable to Pannetrat-Molva
- Overhead:
  - (n,t) = (128, 64)
  - RSA-1024 signature

→ SAIDA overhead: 22 bytes
Distillation code overhead: 43 bytes
Total Overhead: 65 bytes per packet

Performance Experiment

- Transmitted 4 Mb/s stream
  (128,64) erasure code
  RSA-1024 signatures
- Receiver: 2.4 GHz Pentium 4, 1GB RAM
- Worst case attack
- Varied attack factor f = 0..10

Performance



Conclusion

- Distillation codes extend erasure codes for malicious environments
- Distillation codes protect SAIDA and Pannetrat-Molva against DoS
- Applicable to other multicast and distributed storage protocols

Question