# Fault-Tolerant and Secure Distributed Data Storage Using Random Linear Network Coding

Pouya Ostovari and Jie Wu

Computer & Information Sciences
Temple University

Center for Networked Computing
http://www.cnc.temple.edu

# Agenda

- Introduction
  - Distributed data storages
  - Network coding
- Setting
  - Problem formulation
- Fault-tolerant and secure data storage
- Evaluations
- Conclusions

# Introduction

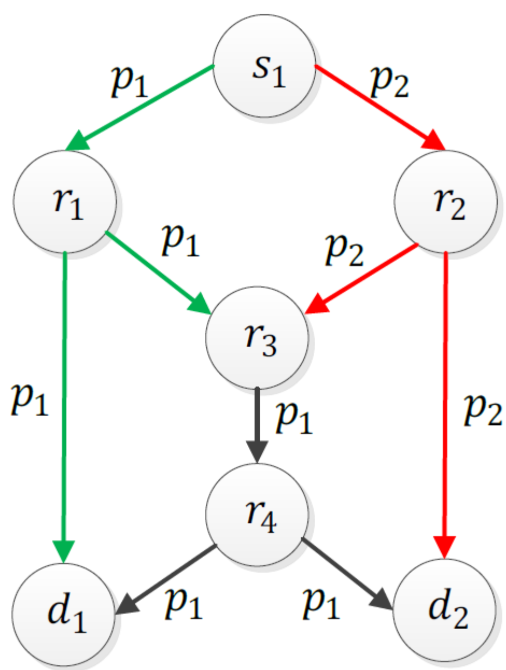
- Popularity of cloud storages
  - More convenient that local copies
    - Access from different devices
  - Fault-tolerant
    - Storing data on multiple storages
    - Different geographic location (data centers)
  - More secure
    - Encryption
    - Advanced security mechanisms
  - Different versions of files

# Fault tolerance

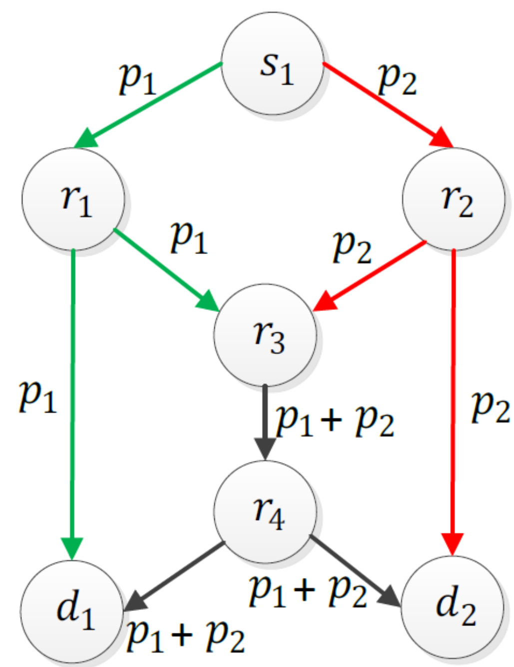
- Fault tolerance
  - Storing redundant data on multiple storages
  - More redundancy → higher level of protection against storage failure
  - More redundancy requires increases the cost
- Previous work
  - Finding amount of redundancy to achieve a given level of fault tolerance
  - Different coding methods
- Fault tolerance using network coding

# Network Coding in Wired Networks

- Single multicast session
  - Bottleneck problem (Ahlswede, 2000)




No coding




Coding

# Network Coding

- Random linear network coding
  - Linear combinations of the packets

$$\begin{cases} q_1 = \alpha_{1,1}p_1 + \alpha_{1,2}p_2 + \alpha_{1,3}p_3 \\ q_2 = \alpha_{2,1}p_1 + \alpha_{2,2}p_2 + \alpha_{2,3}p_3 \\ \vdots \\ q_\mathrm{n} = \alpha_{n,1}p_1 + \alpha_{n,2}p_2 + \alpha_{n,3}p_3 \end{cases}$$

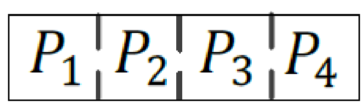
  - 3 linearly independent coded packets are sufficient for decoding
  - Gaussian elimination

# Network Coding

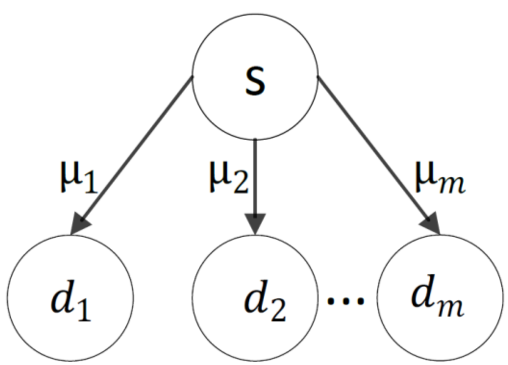
- Applications of network coding
  - Reliable transmissions
    - Wireless/wired networks
  - Throughput/capacity enhancement
    - Distributed storage systems
    - Content distribution
    - Layered multicast
  - Providing security

# Applications of Network Coding

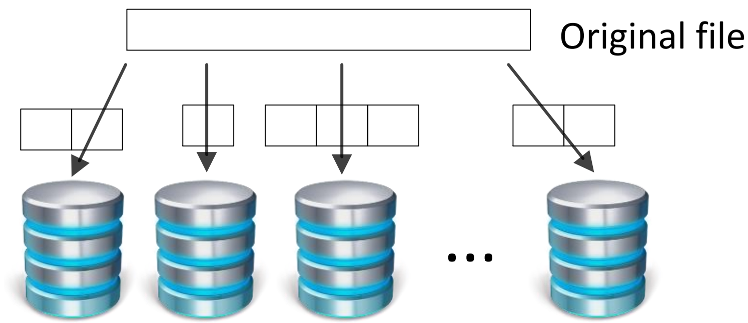
- Transmissions in wireless networks
- Intra-flow coding
  - Reliability




$$\sum_{i=1}^{4} \beta_i . P_i$$



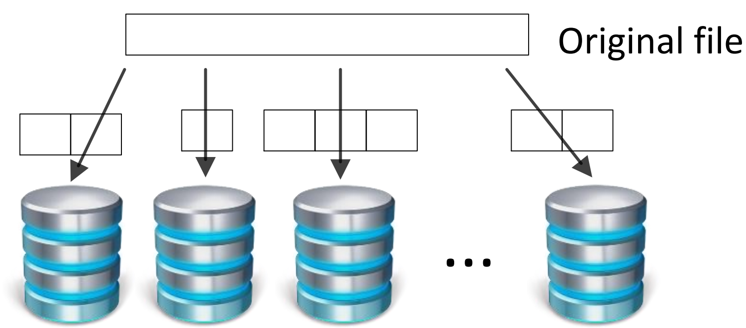
# System Setting




- Distributed data storage system
  - *n* storages
  - Each of these data storages might fail with probability $\epsilon_i$
    - Due to power limitation, hardware problems, high workload,…
  - Eavesdropper can access the *i*th data storage with probability $\gamma_i$
  - Storing a file: *m* packets

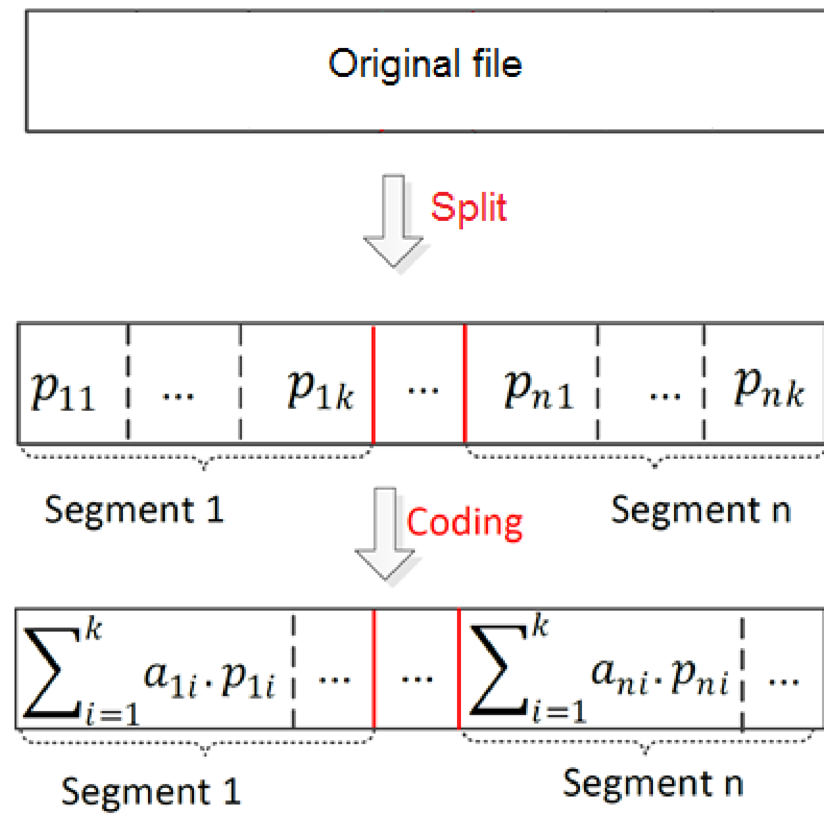
# System Setting

- Objective:
  - Providing fault-tolerance and security
  - Using random linear network coding



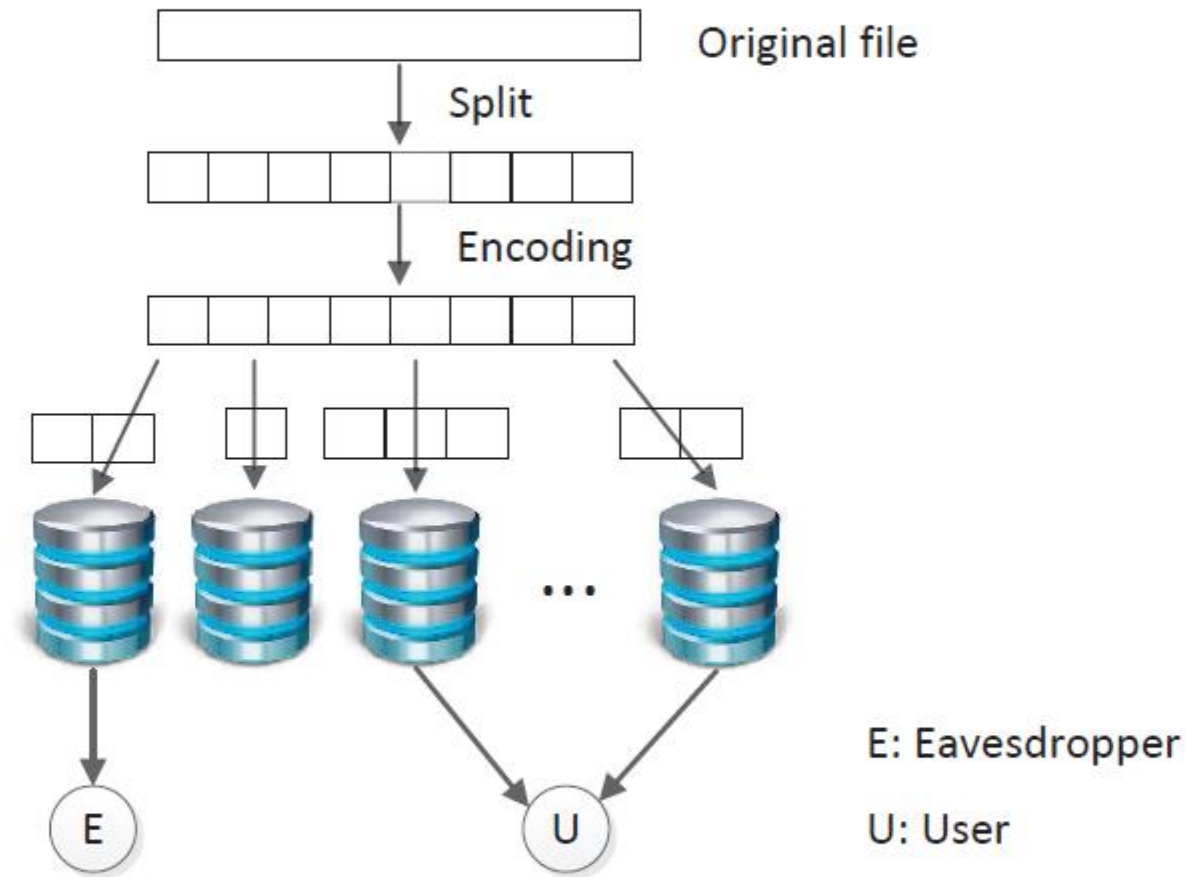
# Fault-Tolerant Data Storage using NC

- Splitting the original file to segments of the same size
- Performing NC among the packets of the same segment
- Storing *x* percent of each segment



# Fault-Tolerant Data Storage using NC

- Fault-tolerance vs. security



# Fault-Tolerance and Security

- Fault-tolerance
  - *m* linearly independent coded packets are sufficient for retrieving the original data
- Security
  - Eyedropper cannot decode the coded packets unless it has access to *m* linearly independent packets
- Challenge

More stored coded packets

More robust against failures

More vulnerable against eavesdropping

- Trade-off between security and fault tolerance

# Problem Formulation

- Case 1: We define the objective function as a function of fault tolerance and security.

$$\min\ U = \sum_{R_j \in R} \alpha_1 q_j y_j - \alpha_2 p_j y_j$$

$$s.t \quad y_j \in \{0,1\} \quad \forall\ R_j \in R$$

$$p_j = \prod_{s_k \in R_j} (1 - \epsilon_k) \prod_{s_i \notin R_j} \epsilon_i$$

$$q_j = \prod_{s_k \in R_j} \gamma_k \prod_{s_i \notin R_j} (1 - \gamma_i)$$

# Problem Formulation

- Case 2: we fix the fault tolerance into a specific threshold, and set it as a constraint of the optimization.
- We then minimize the eavesdropping probability.

$$\min \ U = \sum_{R_j \in R} q_j y_j$$

$$s.t \quad \sum_{R_j \in R} p_j y_j \geq t_2$$

$$y_j \in \{0, 1\} \quad \forall \ R_j \in R$$

# Problem Formulation

- Case 3: This is the opposite of Case 2.
- We define an eavesdropping probability threshold and set it as a constraint.
- We maximize the fault tolerance.

$$\max \ U = \sum_{R_j \in R} p_j y_j$$

$$s.t \quad \sum_{R_j \in R} q_j y_j \leq t_1$$

$$y_j \in \{0, 1\} \quad \forall \ R_j \in R$$

# Relaxation to Linear Programming

- Case 1: $$\min \ U = \sum_{R_j \in R} \alpha_1 q_j z_j - \alpha_2 p_j z_j$$

$$s.t \quad z_j = \sum_{i:s_i \in R_j} x_i \quad \forall \ R_j \in R$$

$$z_j, x_i \in (0,1) \quad \forall \ R_j \in R, s_i \in S$$

$$\min \ U = \sum_{s_i \in S} \alpha_1 \gamma_i x_i - \alpha_2 (1 - \epsilon_j) x_i$$

$$s.t \quad x_i \in (0,1) \quad \forall s_i \in S$$

# Relaxation to Linear Programming

- Case 2:

$$\min \; U = \sum_{R_j \in R} q_j z_j$$

$$s.t \quad \sum_{R_j \in R} p_j z_j \geq t_2$$

$$z_j = \sum_{i : s_i \in R_j} x_i \quad \forall \; R_j \in R$$

$$z_j, x_i \in (0, 1) \quad \forall \; R_j \in R, s_i \in S$$

# Relaxation to Linear Programming

- Case 3:

$$\min\ U = \sum_{R_j \in R} p_j z_j$$

$$s.t \quad \sum_{R_j \in R} q_j z_j \geq t_1$$

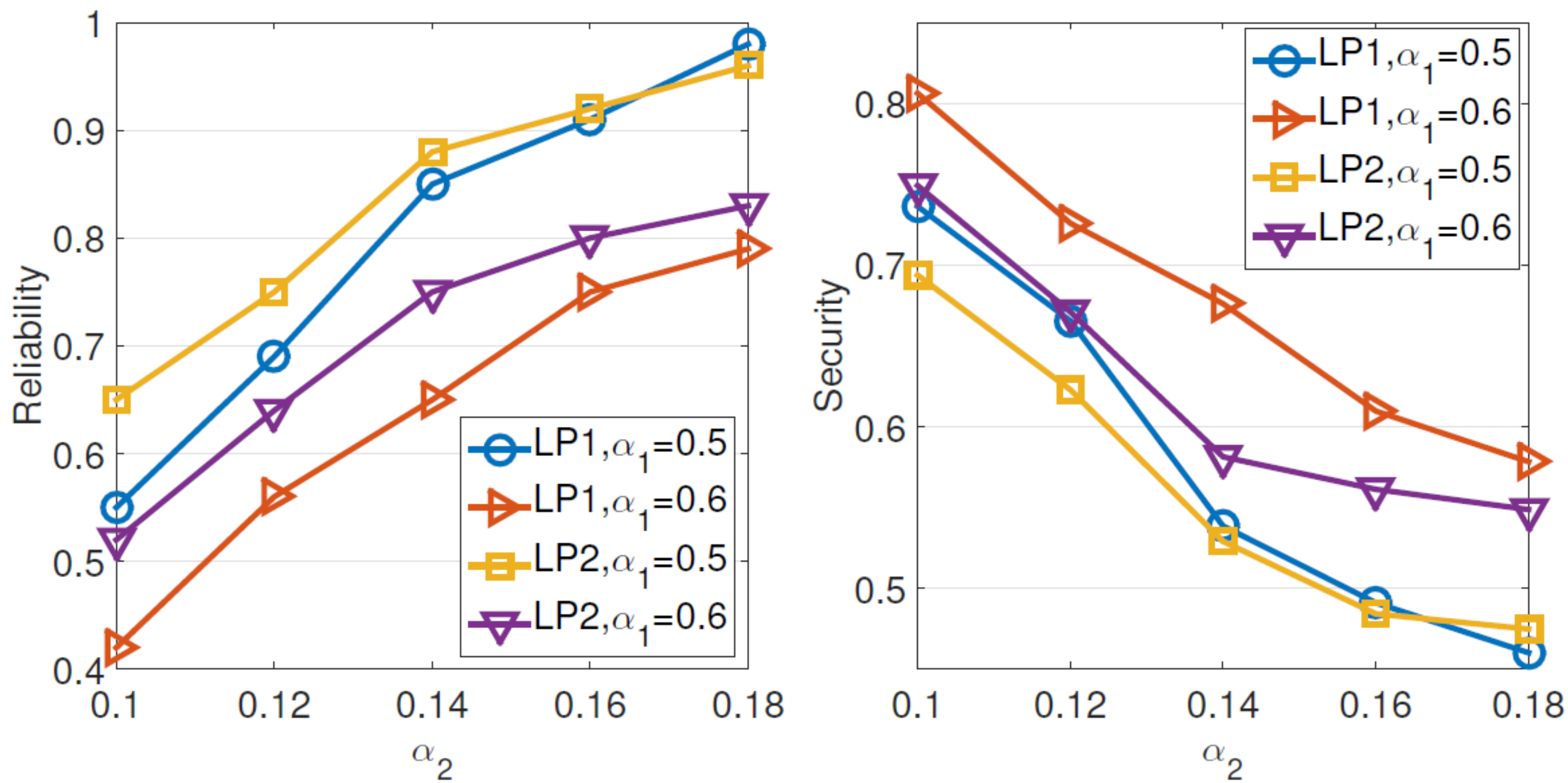
$$z_j = \sum_{i:s_i \in R_j} x_i \quad \forall\ R_j \in R$$

$$z_j, x_i \in (0,1) \quad \forall\ R_j \in R, s_i \in S$$

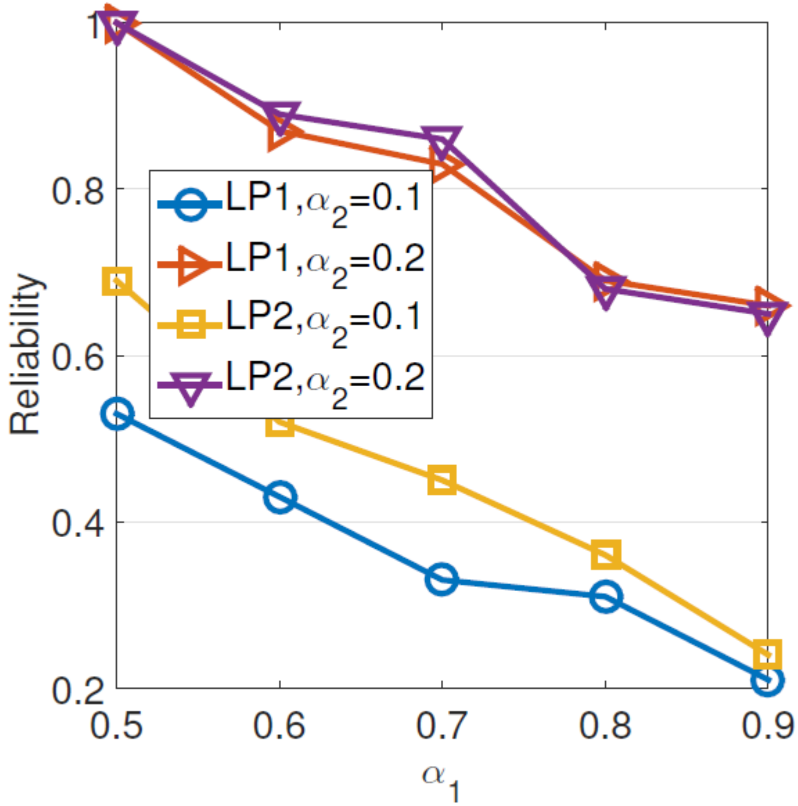
# Evaluations

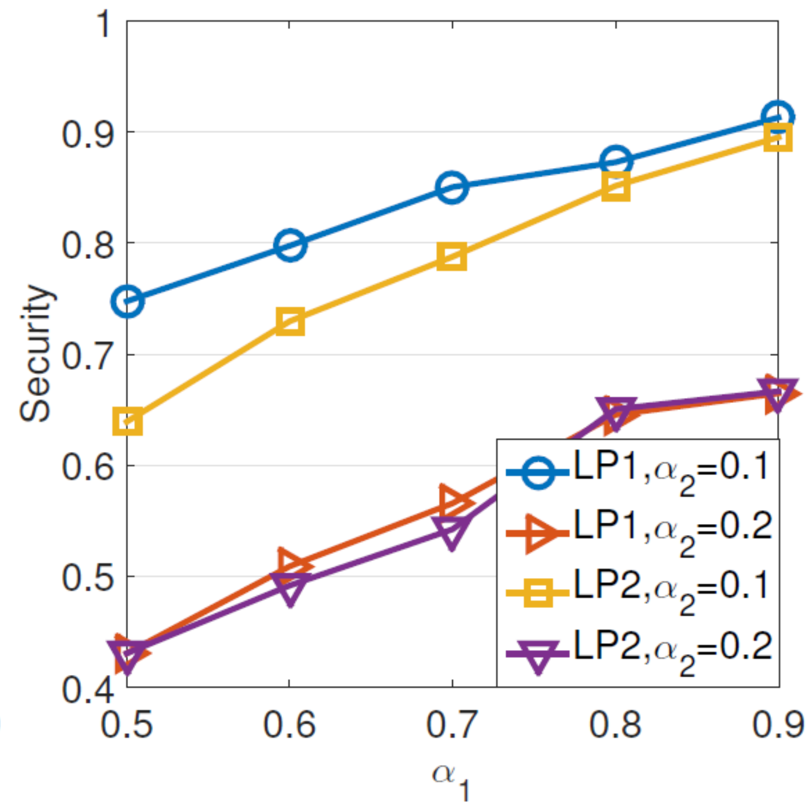
- Simulator in Matlab environment
- We use Linprog tool of Matlab to find the solution of the optimizations
- 100 simulation runs

# Evaluations

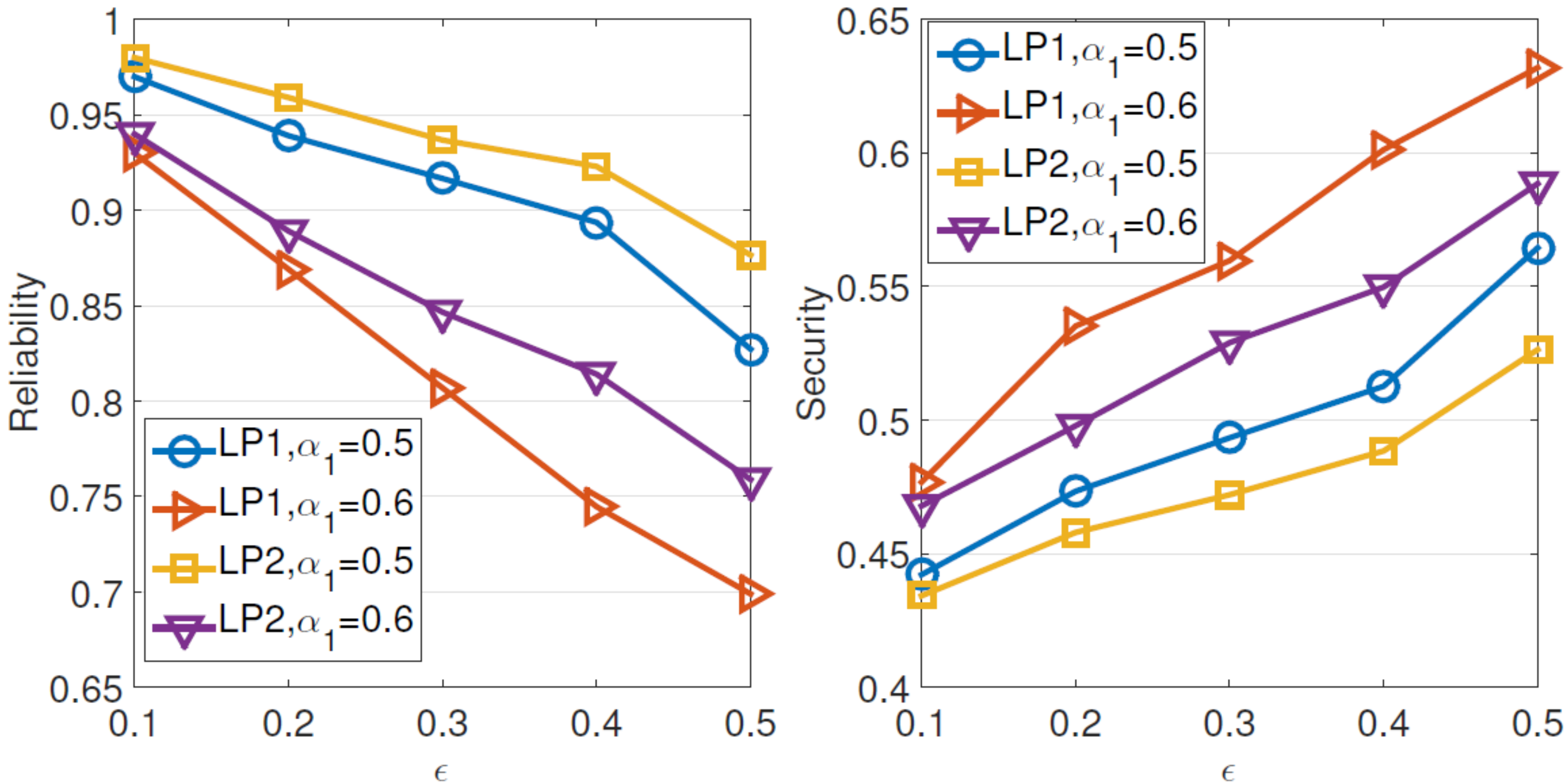


# Evaluations



# Evaluations



# Conclusion

- Fault-tolerance using network coding
  - Storing redundant data on multiple storages
- Security using network coding
  - Preventing eavesdropper to receive sufficient coded packets
- Trade-off between fault-tolerance and security

Thank you