IEEE ICC®

GRAND-EDGE: A Universal, Jamming-resilient Algorithm with Error-and-Erasure Decoding

Furkan Ercan[†], Kevin Galligan[*], David Starobinski[†], Muriel Médard[§], Ken R. Duffy[*], Rabia Tugce Yazicigil[†]
[†]Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA
[§]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA
[*]Hamilton Institute, Maynooth University, Ireland

Presenter: Kevin Galligan

BOSTON UNIVERSITY

Maynooth University
National University of Ireland Maynooth

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

They all broadcast wireless signals at 2.4 GHz frequency regime.

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

They all broadcast wireless signals at 2.4 GHz frequency regime.

Modern wireless techniques ensure reliable communication through frequency hopping, cyclic prefixing, etc.

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

They all broadcast wireless signals at 2.4 GHz frequency regime.

Modern wireless techniques ensure reliable communication through frequency hopping, cyclic prefixing, etc.

On the other hand, powerful interference at a near frequency could disrupt communication severely.

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

They all broadcast wireless signals at 2.4 GHz frequency regime.

Modern wireless techniques ensure reliable communication through frequency hopping, cyclic prefixing, etc.

On the other hand, powerful interference at a near frequency could disrupt communication severely.

Even if channel anomalies are alerted, the transmitted data overpowered by interference is lost.

Communicating over Adversarial Conditions

- What do a Wi-Fi router, a Bluetooth speaker, and a leaky microwave oven have in common?

They all broadcast wireless signals at 2.4 GHz frequency regime.

Modern wireless techniques ensure reliable communication through frequency hopping, cyclic prefixing, etc.

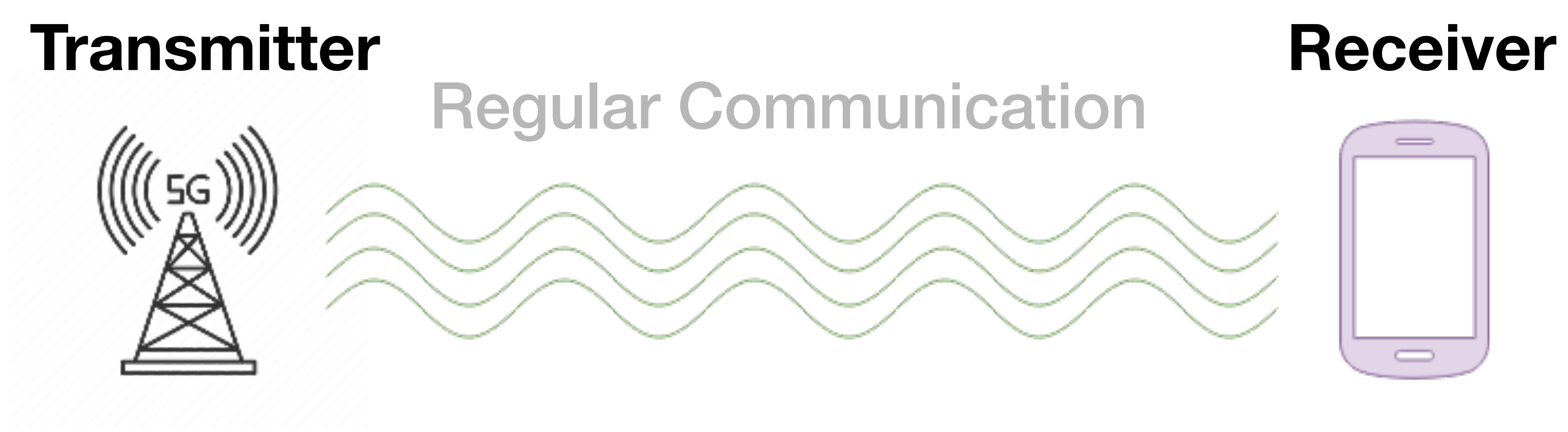
On the other hand, powerful interference at a near frequency could disrupt communication severely.

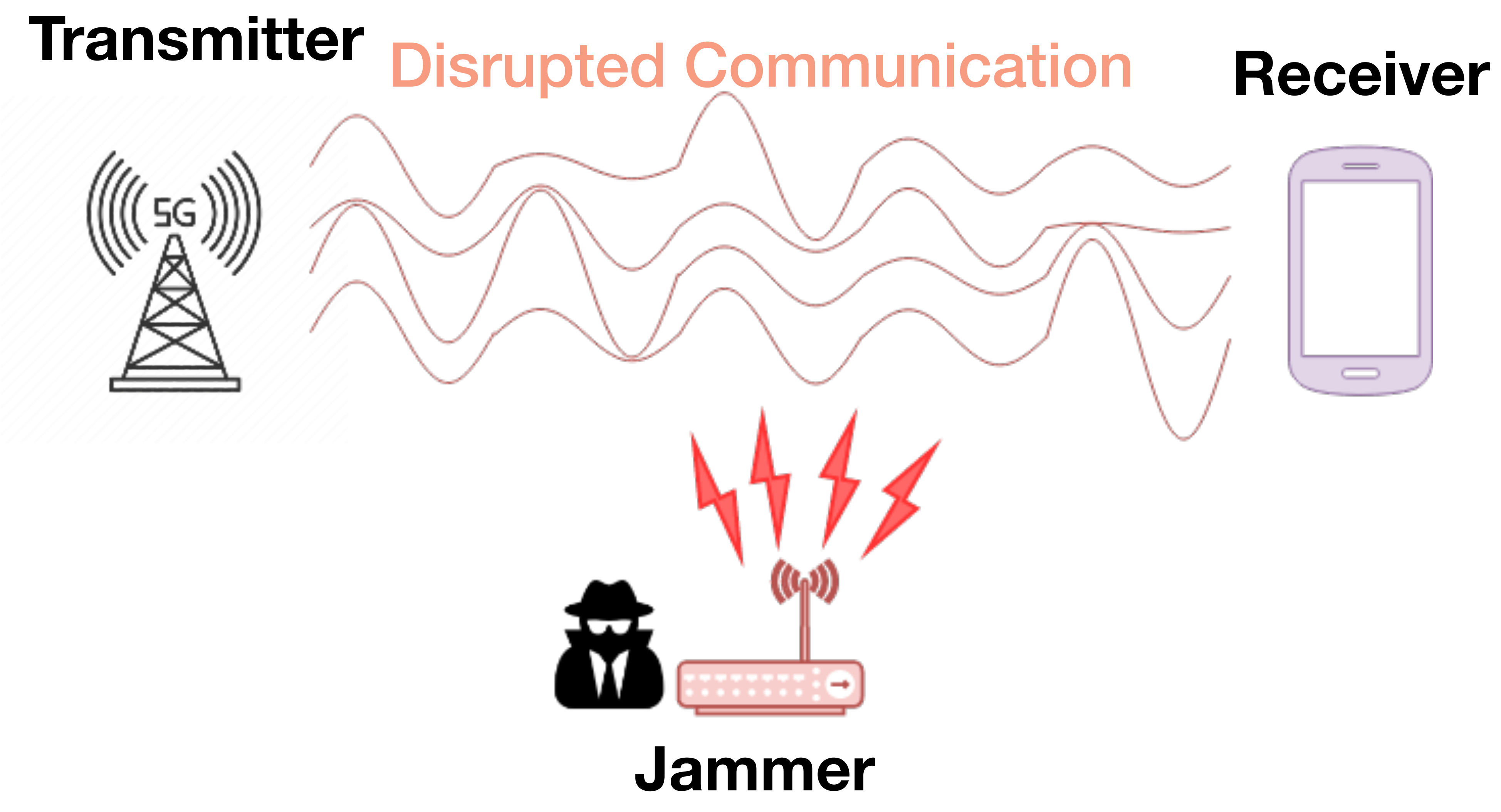
Even if channel anomalies are alerted, the transmitted data overpowered by interference is lost.

Developing resilience against powerful jamming can help retrieve an acceptable quality of service.

Motivation

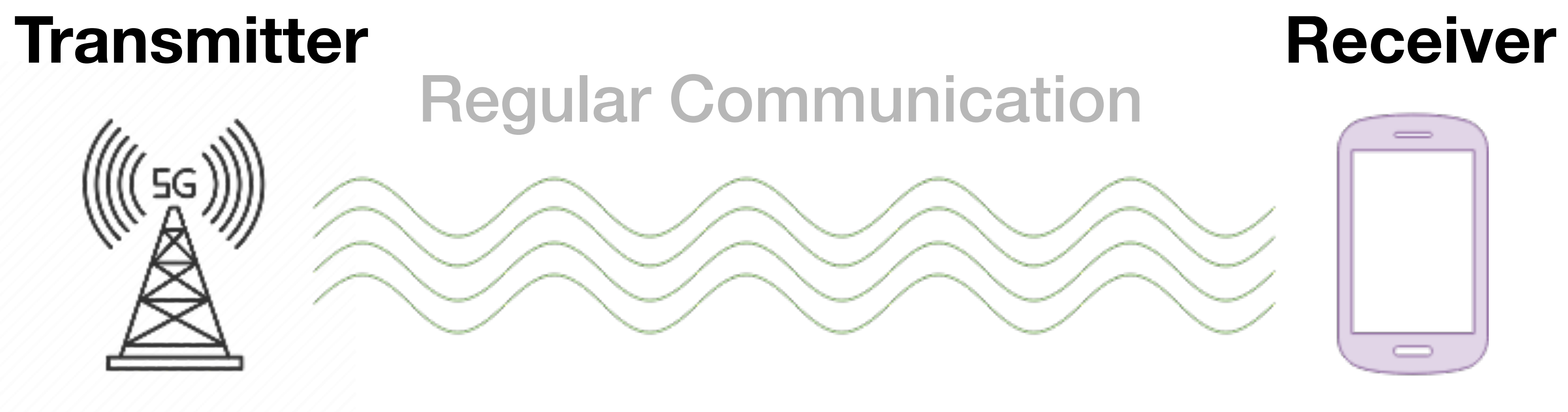
- Channel signal strength indicators can easily detect anomalies but cannot help recover transmitted data.
- This causes heavy retransmissions, added latency, power consumption, degradation of QoS, etc.



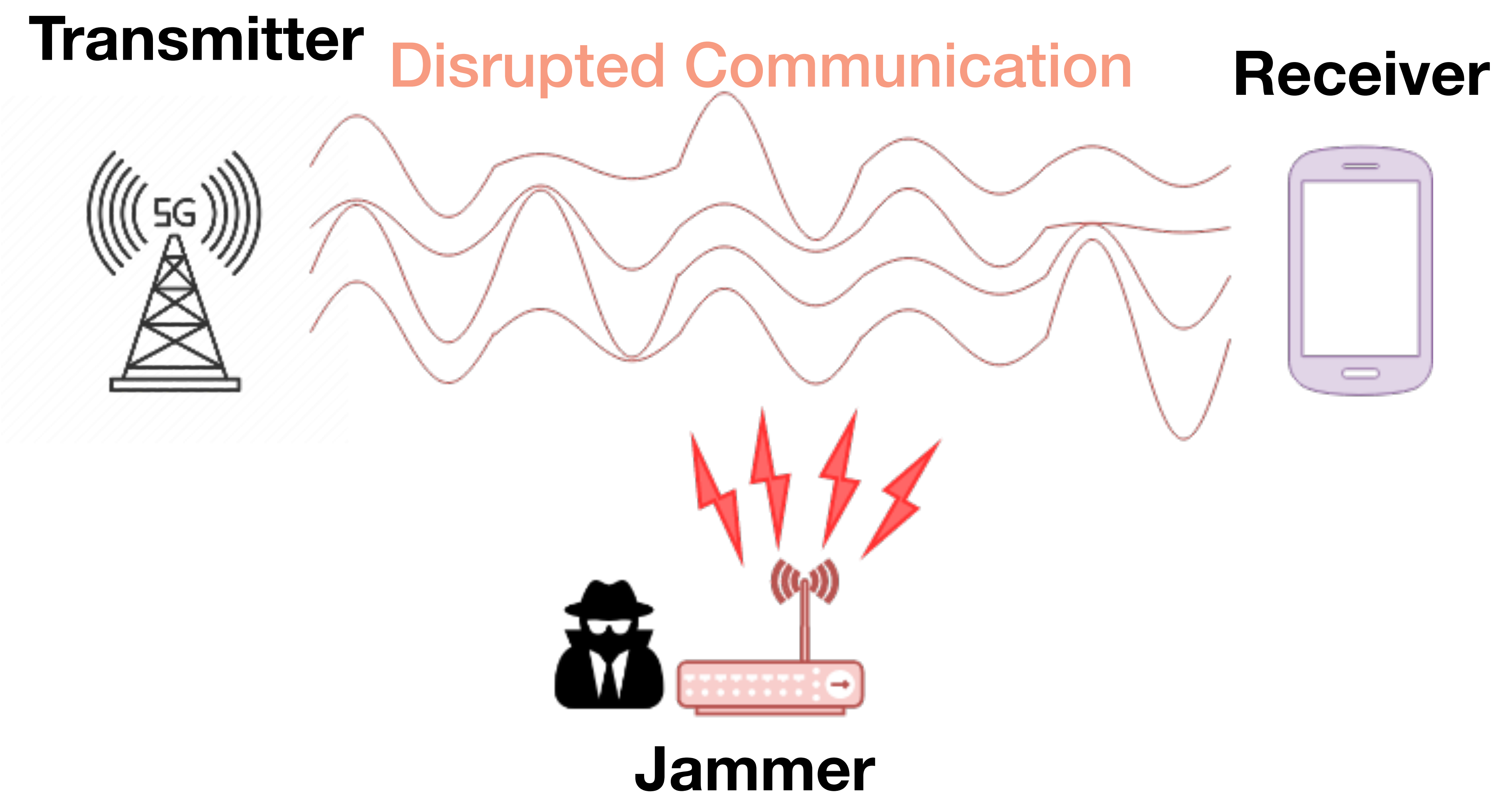


Motivation

- Channel signal strength indicators can easily detect anomalies but cannot help recover transmitted data.
- This causes heavy retransmissions, added latency, power consumption, degradation of QoS, etc.

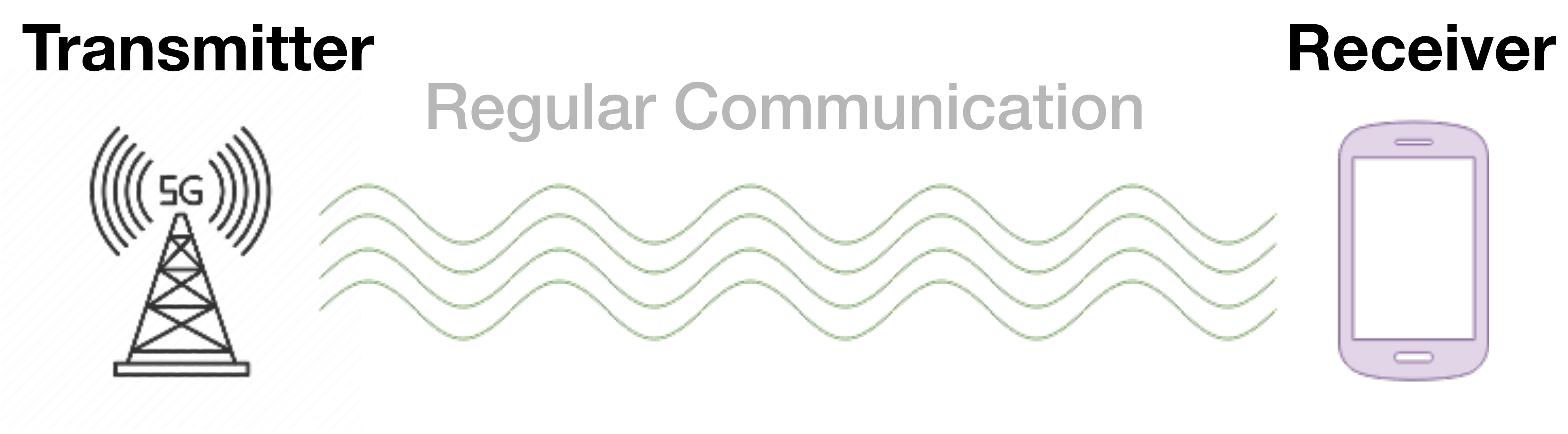


- Our purpose is to add resilience against powerful jamming events.



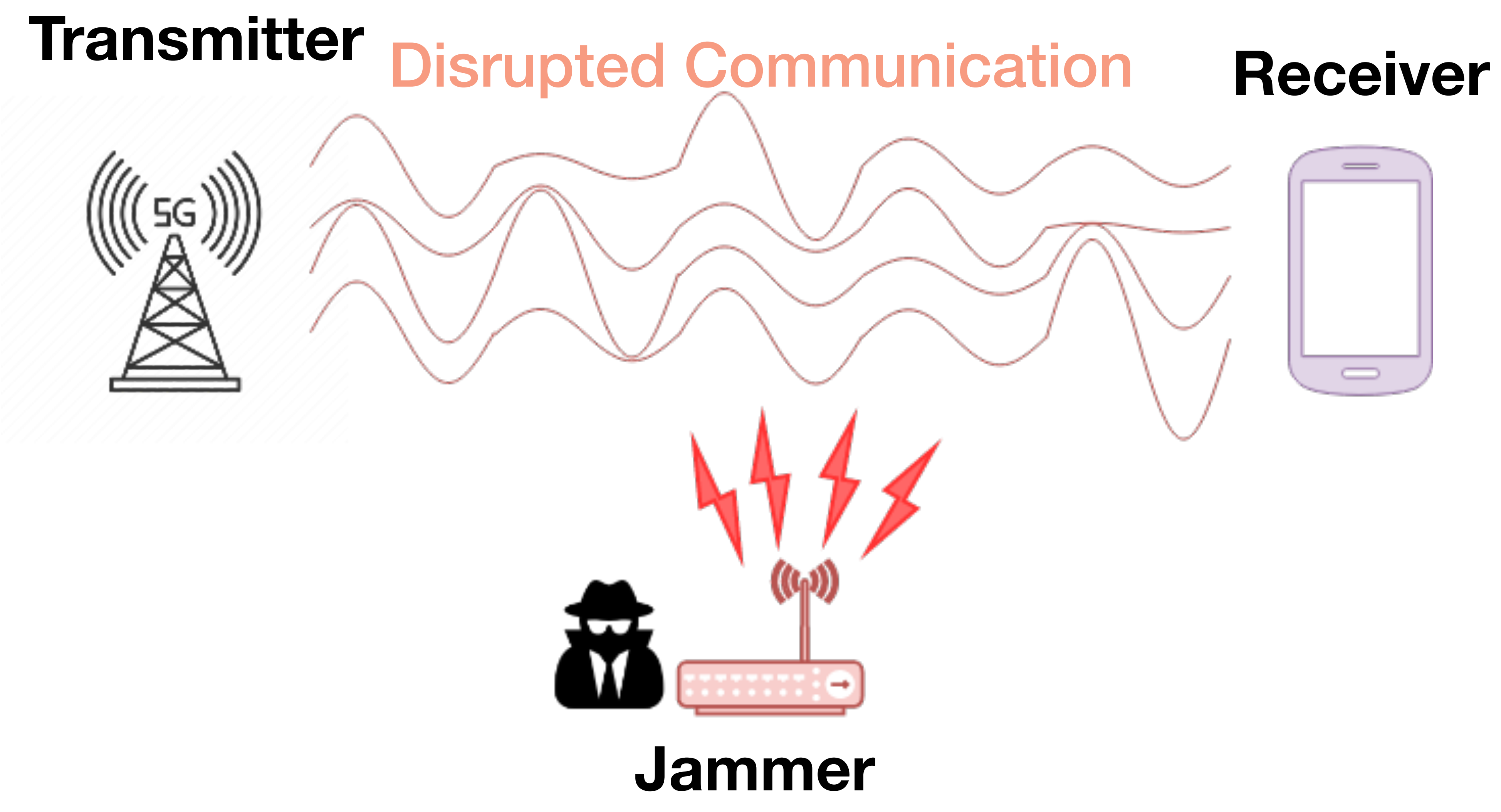
Motivation

- Channel signal strength indicators can easily detect anomalies but cannot help recover transmitted data.
- This causes heavy retransmissions, added latency, power consumption, degradation of QoS, etc.



- Our purpose is to add resilience against powerful jamming events.

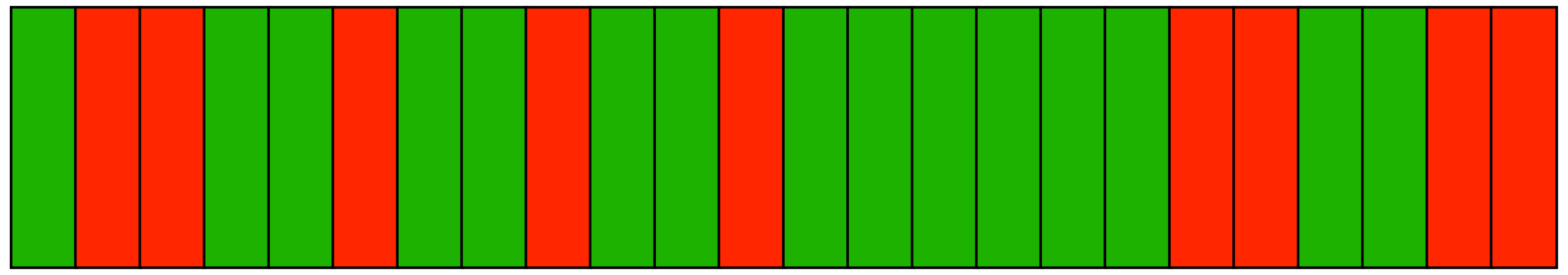
- We aim to develop an error correction algorithm with data recovery capabilities under powerful adversarial conditions.
- Our proposal is in conjunction with the Guessing Random Additive Noise Decoding (GRAND) algorithm, which can work with any codebook.



Overview of This Work

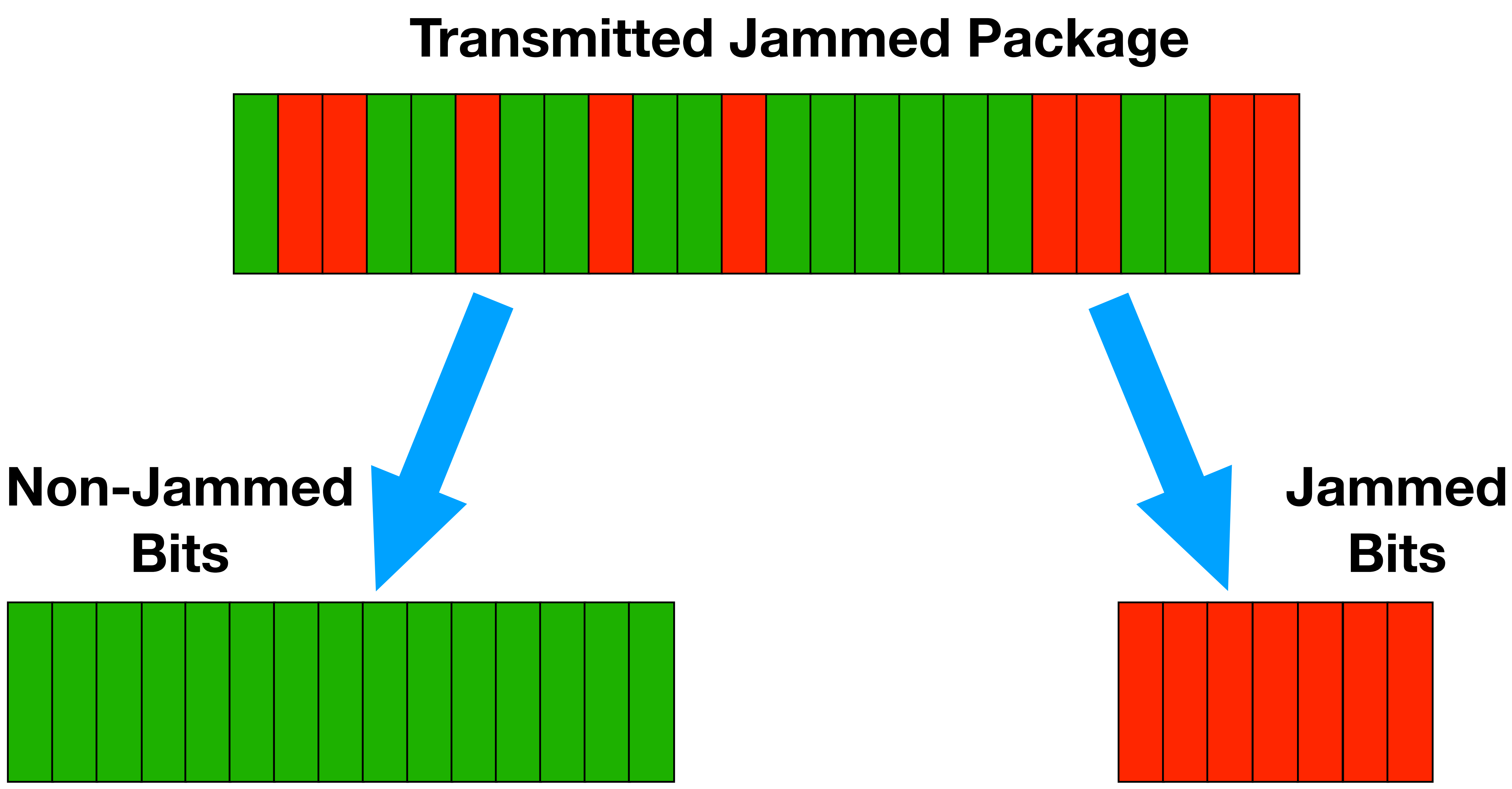
❖ We propose a jamming-resilient algorithm based on GRAND:

Transmitted Jammed Package



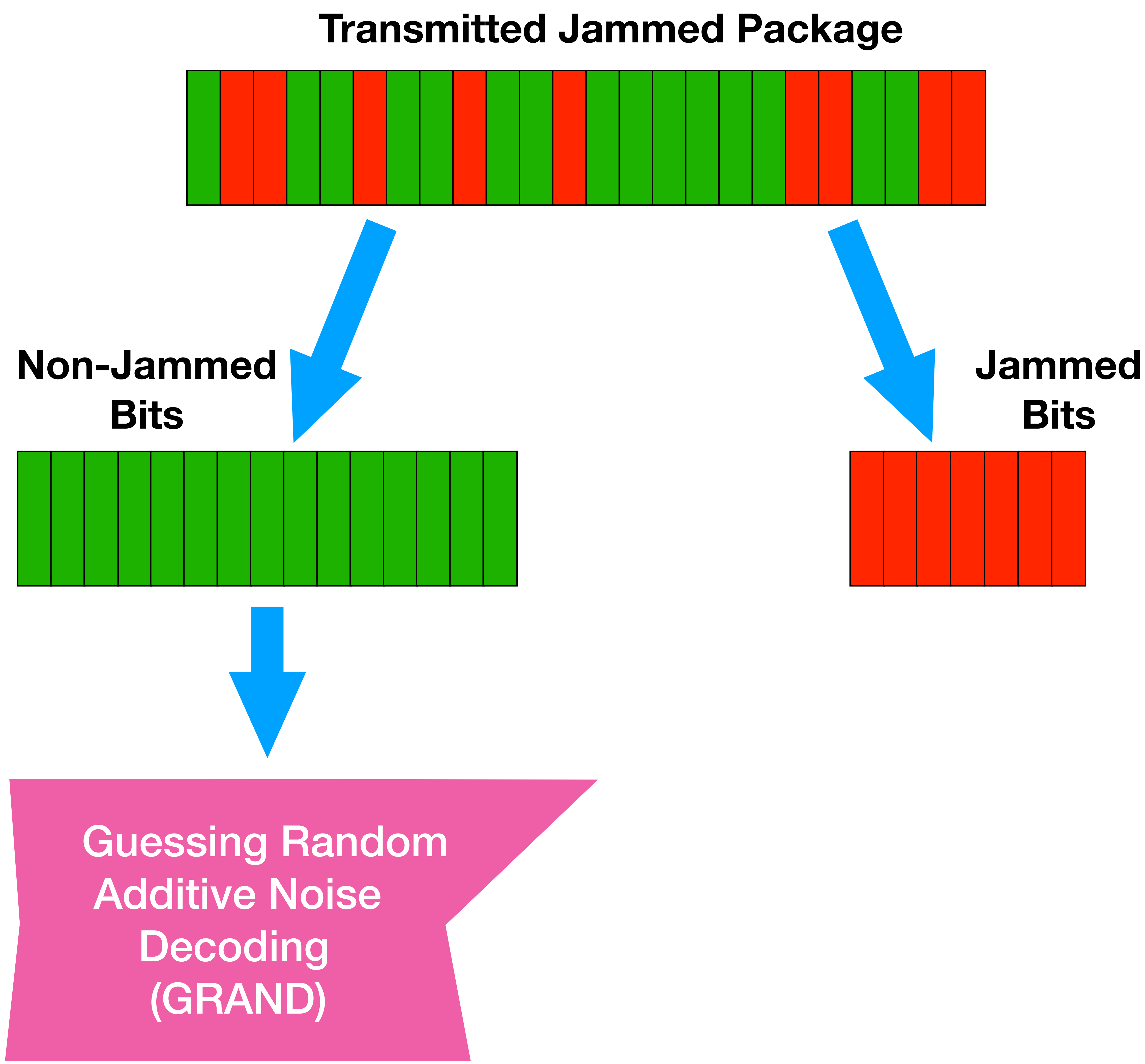
Overview of This Work

- We propose a jamming-resilient algorithm based on GRAND:
  - First, the jammed bits are identified and separated from the rest of the transmission.



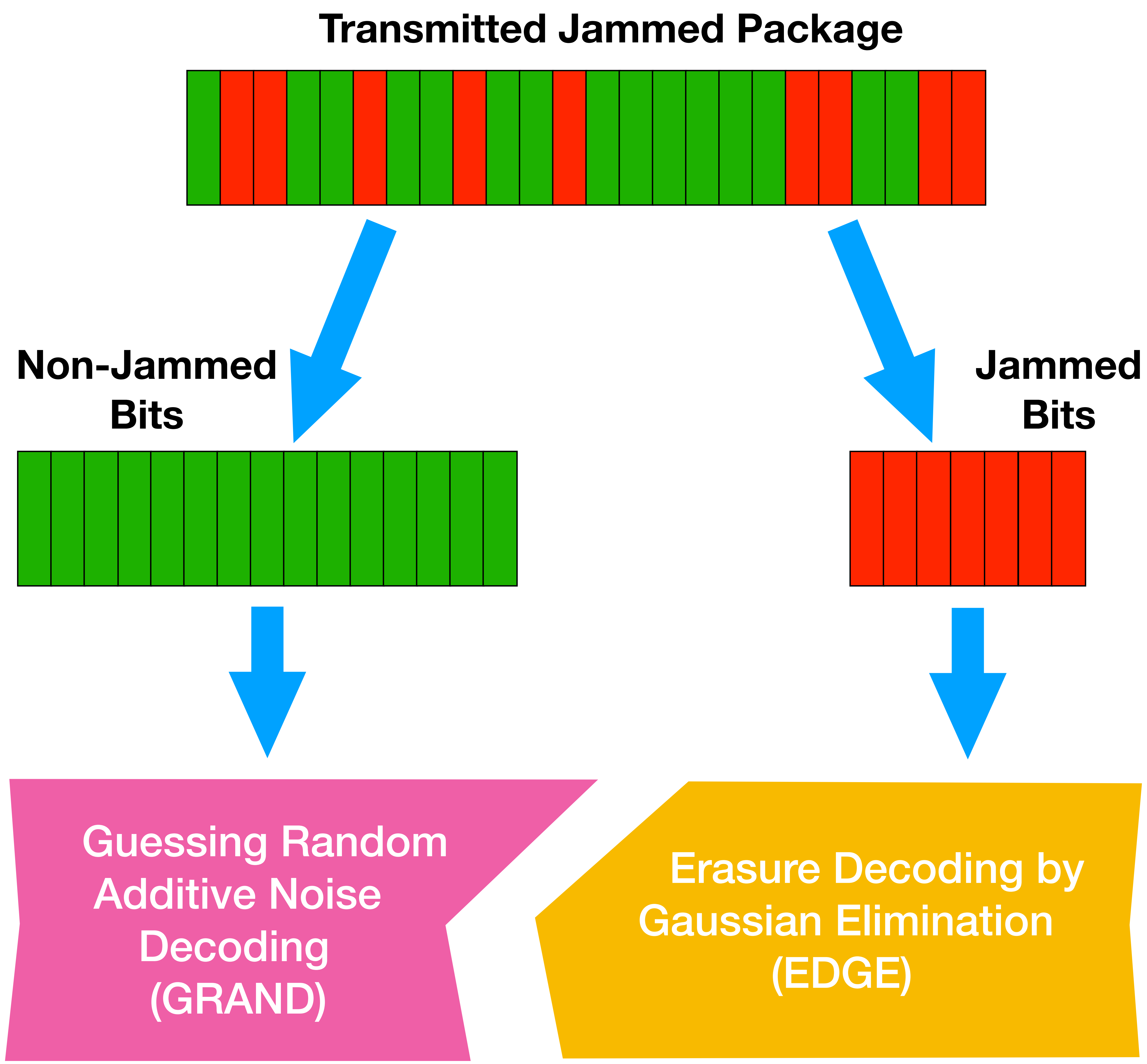
Overview of This Work

- We propose a jamming-resilient algorithm based on GRAND:
  - First, the jammed bits are identified and separated from the rest of the transmission.
  - Next, error correction is performed over the non-jammed parts.



Overview of This Work

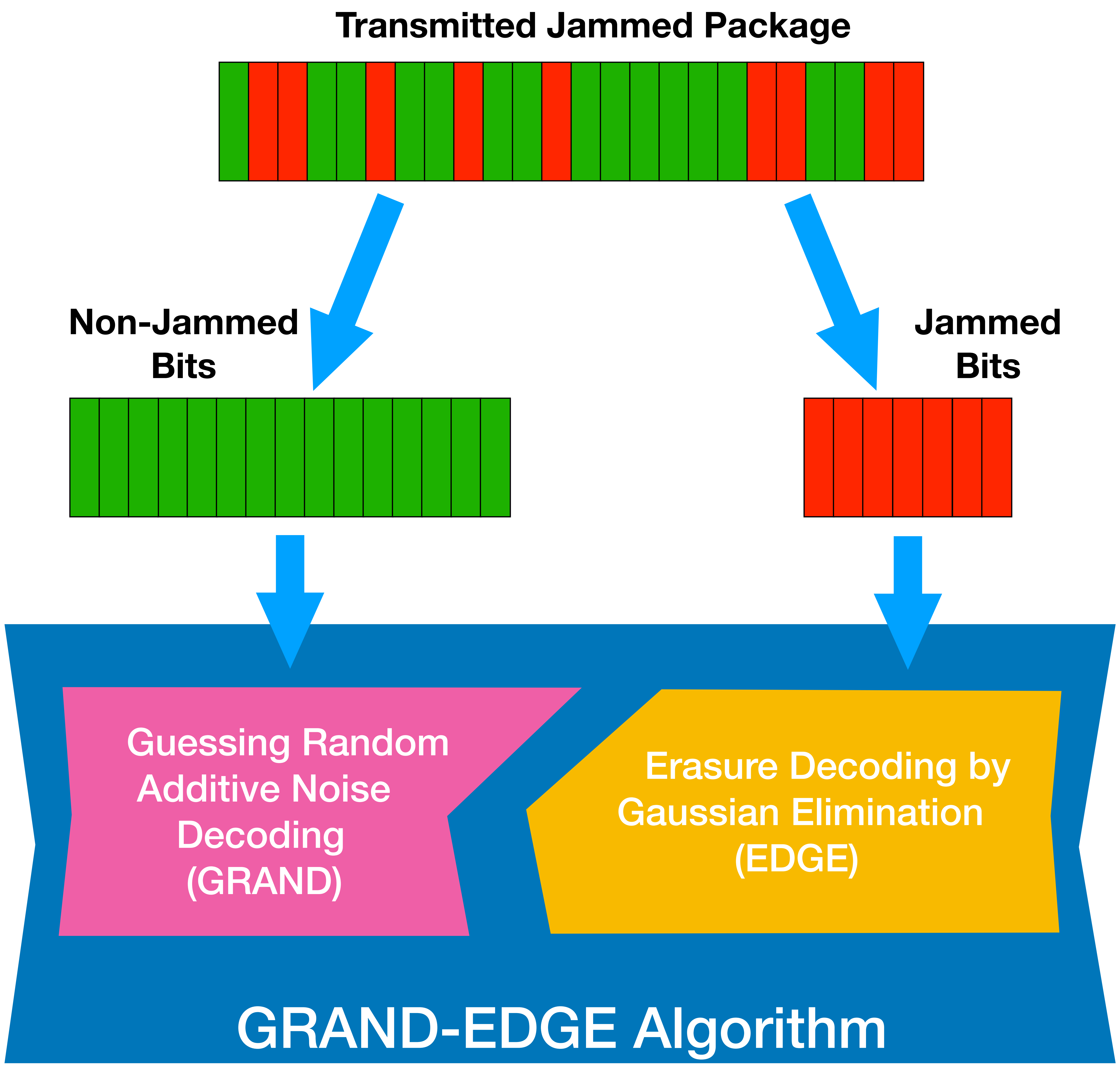
- We propose a jamming-resilient algorithm based on GRAND:
  - First, the jammed bits are identified and separated from the rest of the transmission.
  - Next, error correction is performed over the non-jammed parts.
  - Finally, values of jammed bits are estimated through Gaussian Elimination.



Overview of This Work

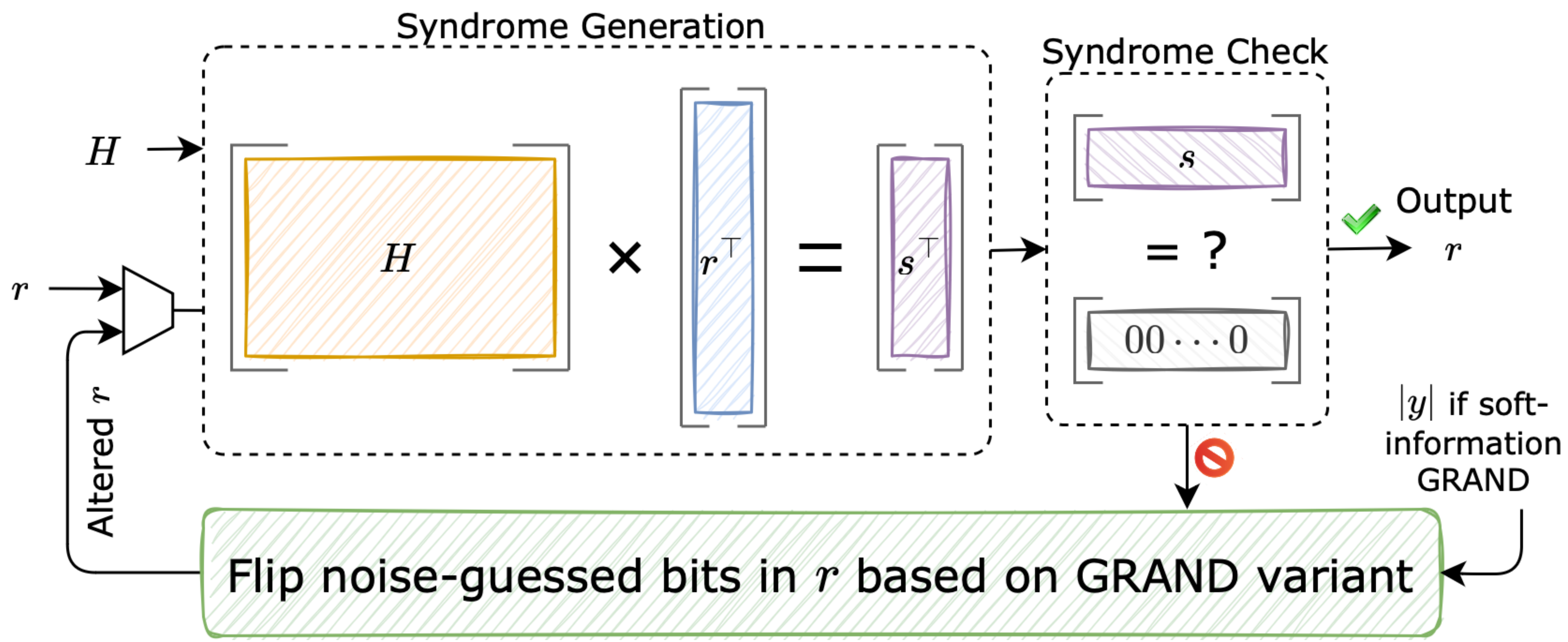
- We propose a jamming-resilient algorithm based on GRAND:
  - First, the jammed bits are identified and separated from the rest of the transmission.
  - Next, error correction is performed over the non-jammed parts.
  - Finally, values of jammed bits are estimated through Gaussian Elimination.

The new algorithm is called the GRAND-EDGE algorithm.



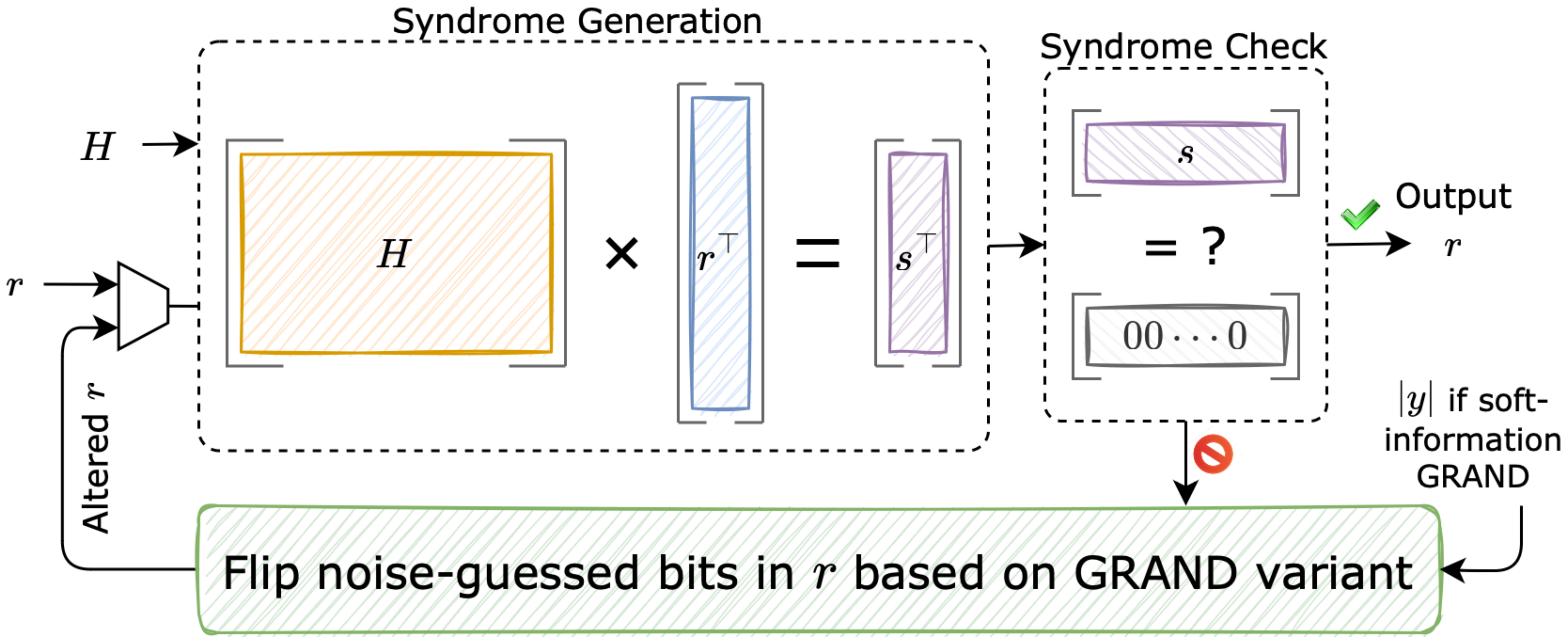
The GRAND Algorithm Family

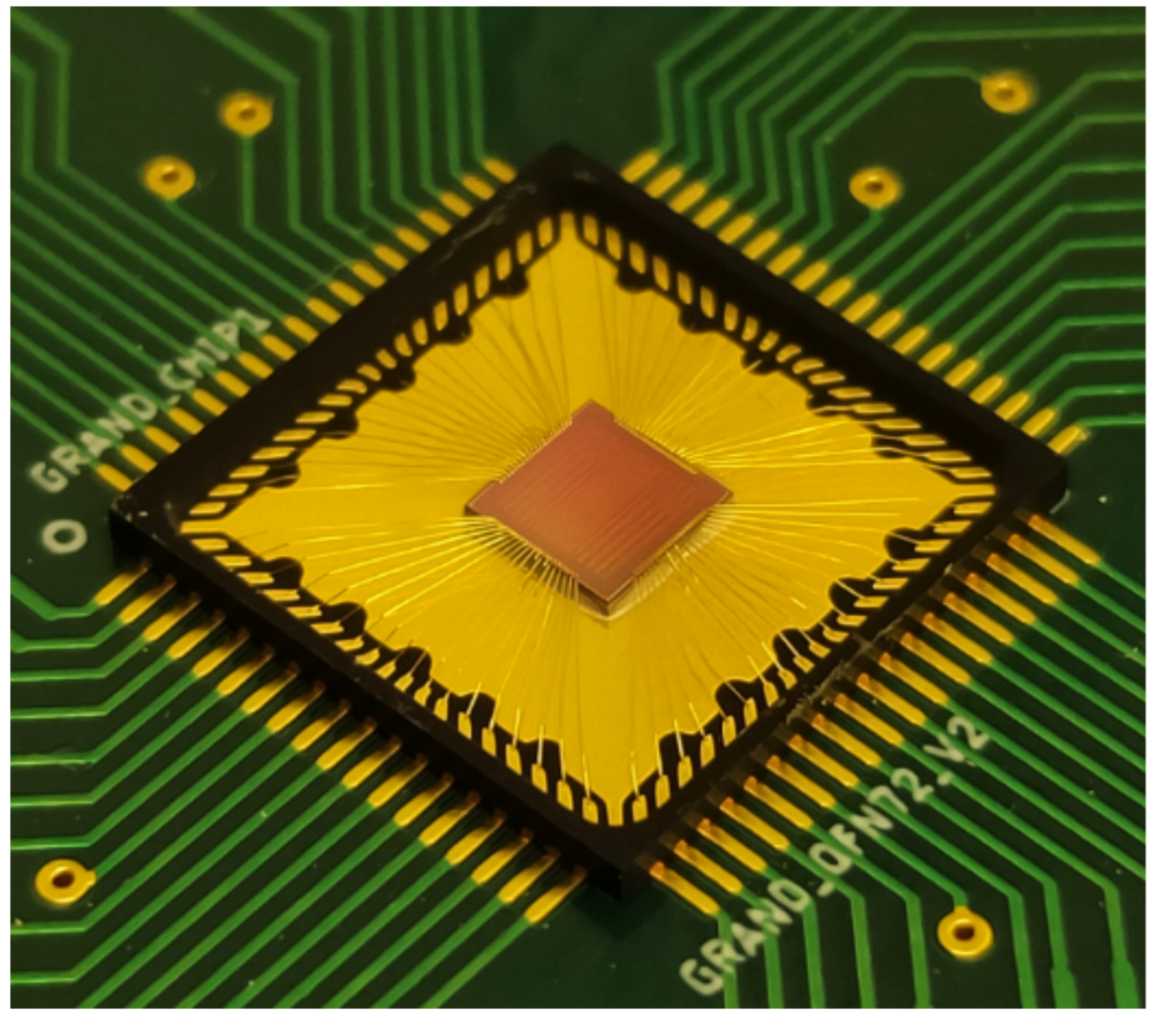
- All GRAND algorithms are based on:
- A syndrome check,
- On a failed syndrome, guessing the noise pattern based on an agenda.
- The ‘agenda’ determines the variant of GRAND.



The GRAND Algorithm Family

- All GRAND algorithms are based on:
- A syndrome check,
- On a failed syndrome, guessing the noise pattern based on an agenda.
- The 'agenda' determines the variant of GRAND.

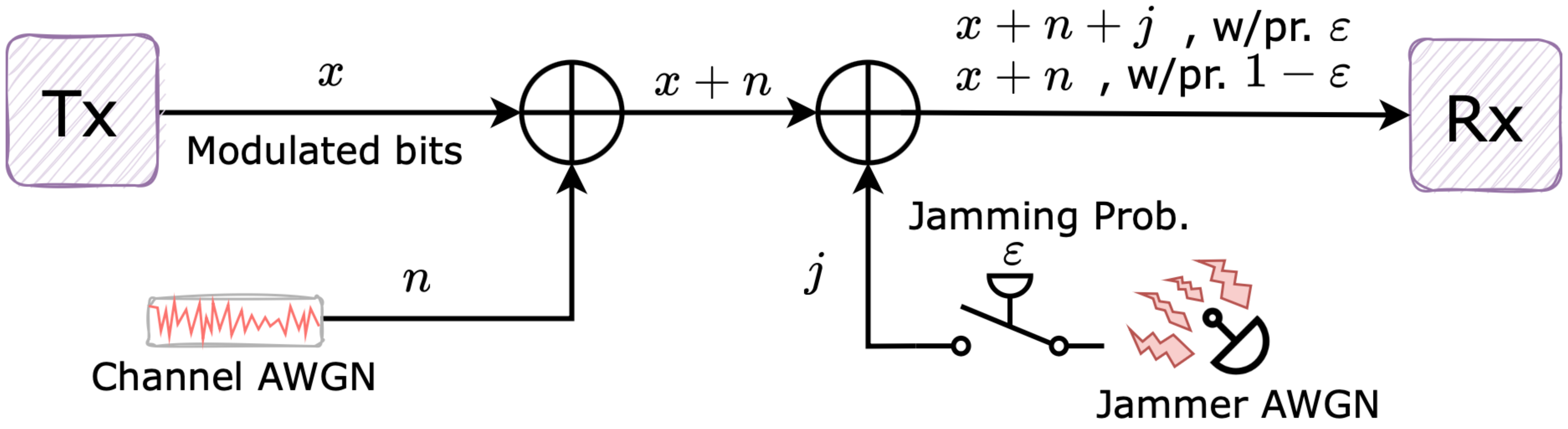




- In this work, we consider:
  - Hard-information GRAND, and
  - Soft-information GRAND (ORBGRAND).

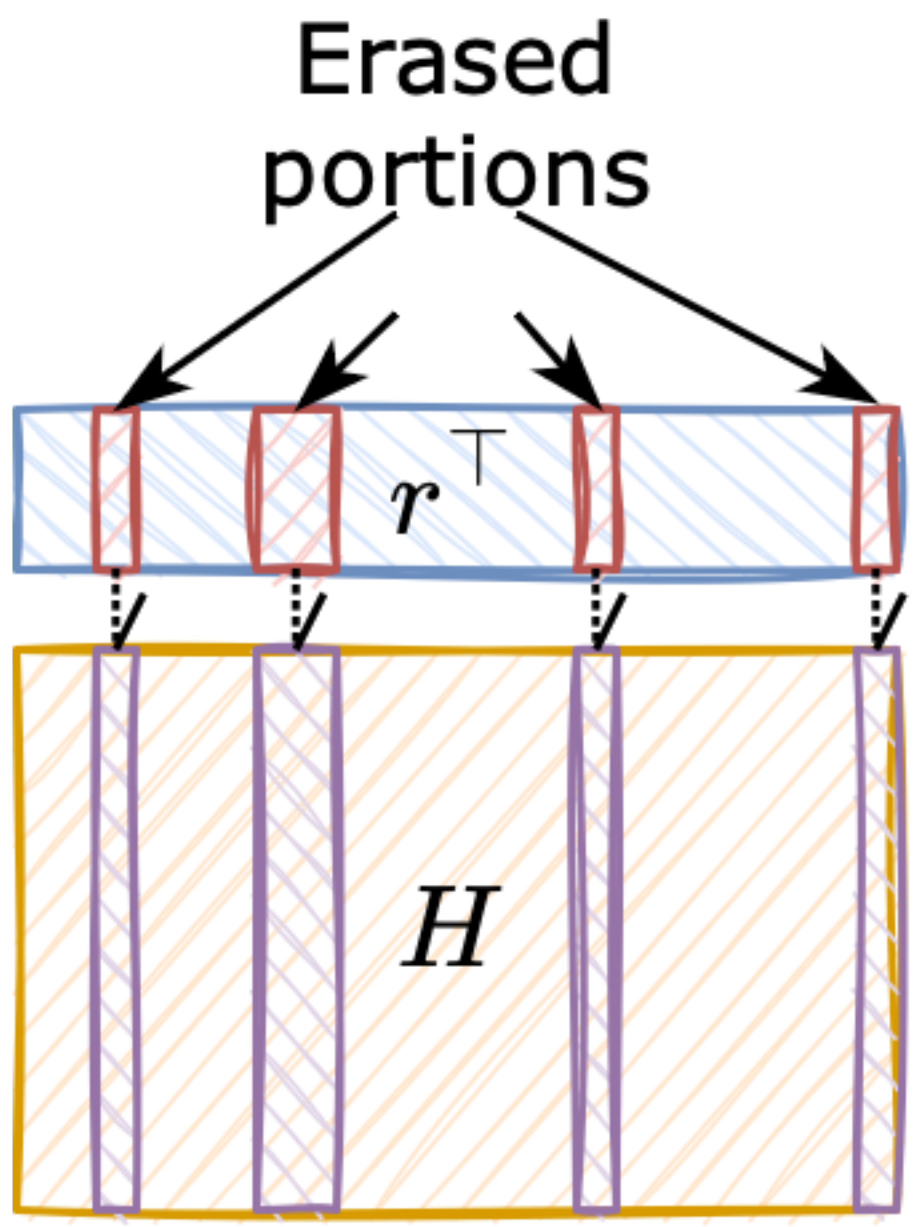
Adversary Channel Model

- We consider an AWGN channel model that is randomly disrupted by a powerful jammer.
- The jammer instance j, which is activated by a probability ϵ, may be added to the transmitted signal.
- The jammer can be modeled as AWGN but with far greater variance.
- If the signal is far stronger than typical, it is considered jammed and its value is not trusted.



$$y = \begin{cases} x+n+j & \text{with probability } \epsilon; \\ x+n & \text{otherwise.} \end{cases}$$

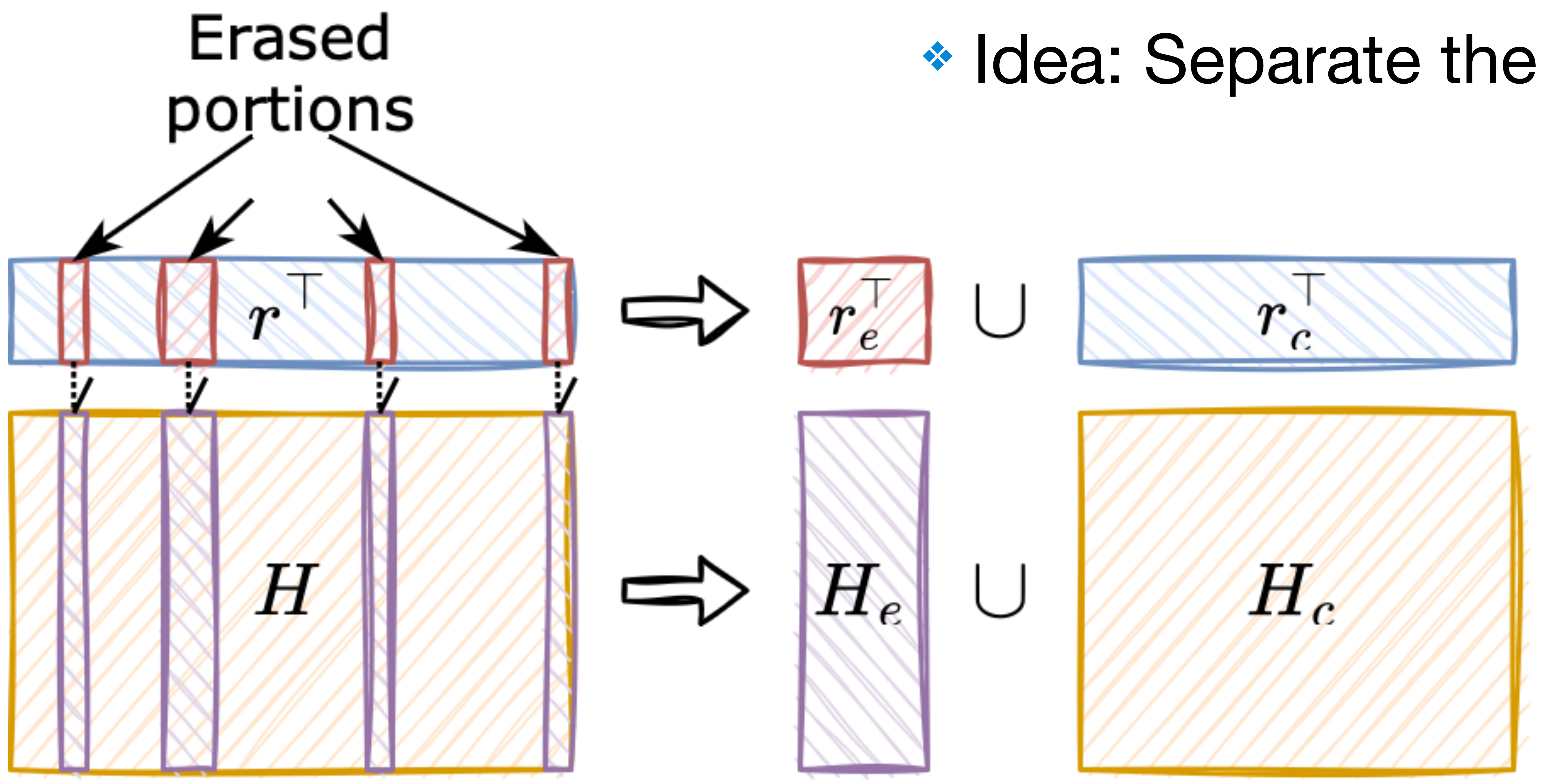
The EDGE Subroutine



- Idea: Separate the jammed bits from the unjammed ones!

- Step 1: Align the received codeword (r) with the parity check matrix (H).

The EDGE Subroutine

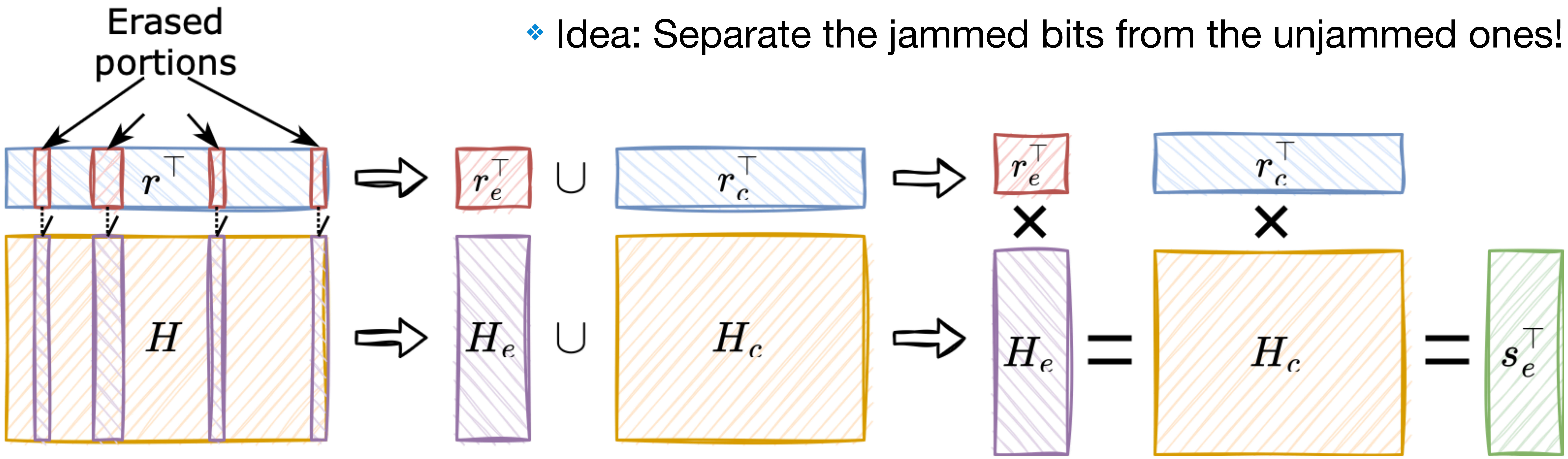


- Idea: Separate the jammed bits from the unjammed ones!

- Step 1: Align the received codeword (r) with the parity check matrix (H).
- Step 2: Separate erased (jammed) columns from the rest in both r and H.

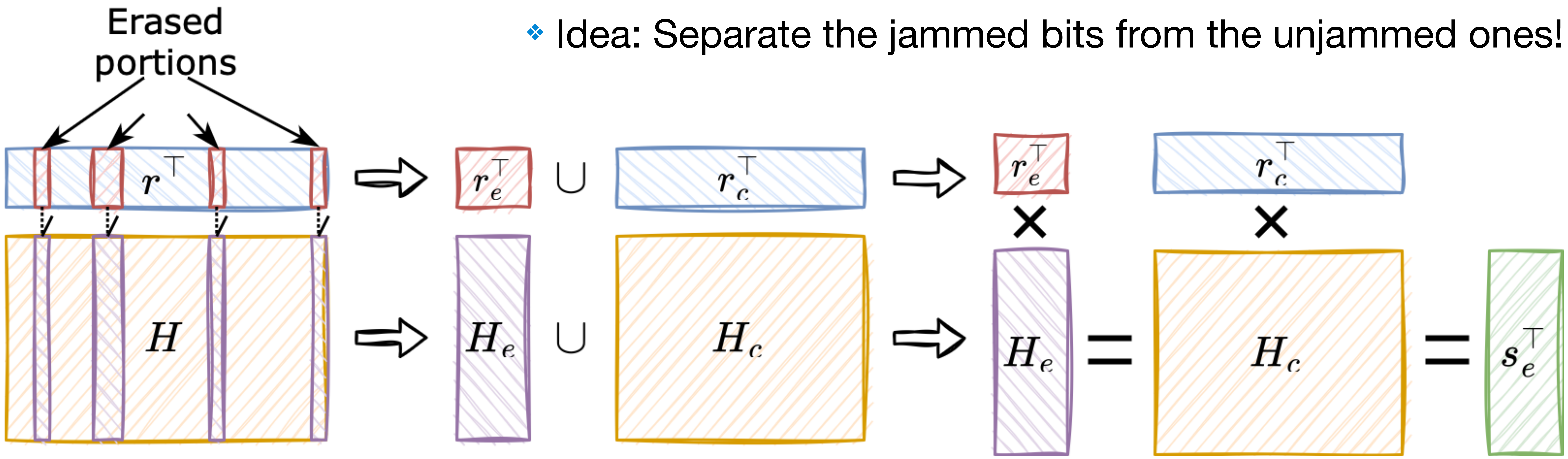
The EDGE Subroutine

- Idea: Separate the jammed bits from the unjammed ones!



- Step 1: Align the received codeword (r) with the parity check matrix (H).
- Step 2: Separate erased (jammed) columns from the rest in both r and H.
- Step 3: Create erasure syndrome by multiplying the received part.

The EDGE Subroutine



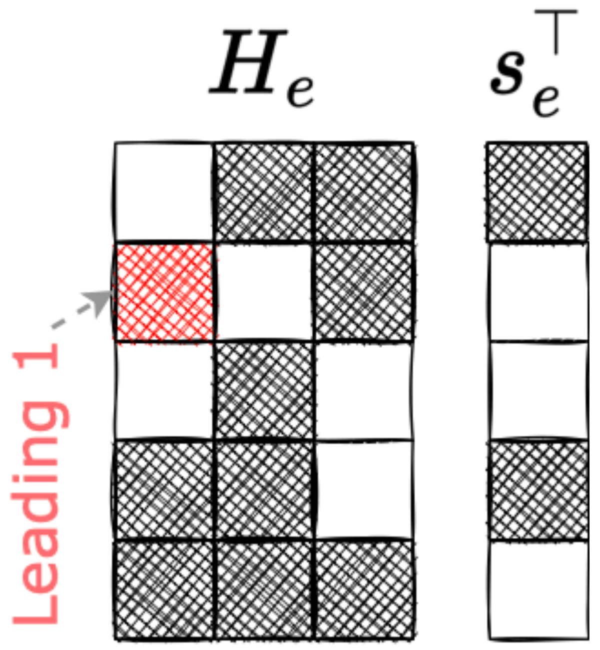
- Idea: Separate the jammed bits from the unjammed ones!

- Step 1: Align the received codeword (r) with the parity check matrix (H).
- Step 2: Separate erased (jammed) columns from the rest in both r and H.
- Step 3: Create erasure syndrome by multiplying the received part.
- Step 4: Solve the system of linear equations for the erased received sequence.

$$\mathbf{H_e} \cdot \boxed{\mathbf{r_e}^\top} = \mathbf{s_e}^\top .$$

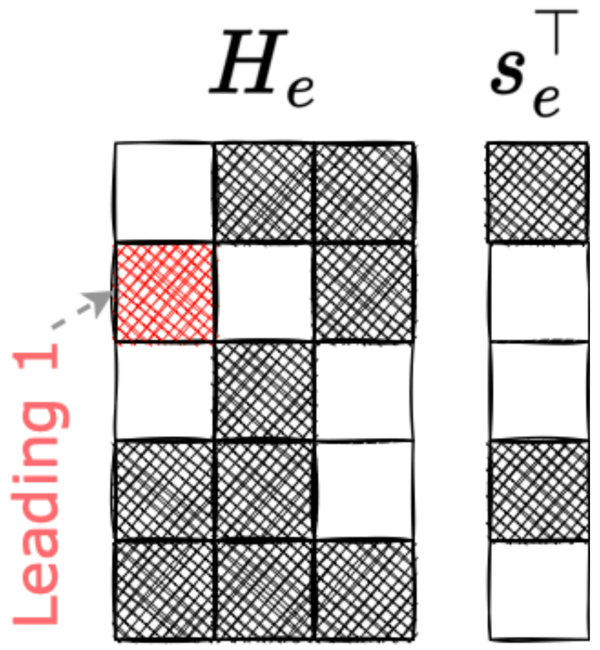
The Gaussian Elimination Process

- Consider the following example:



The Gaussian Elimination Process

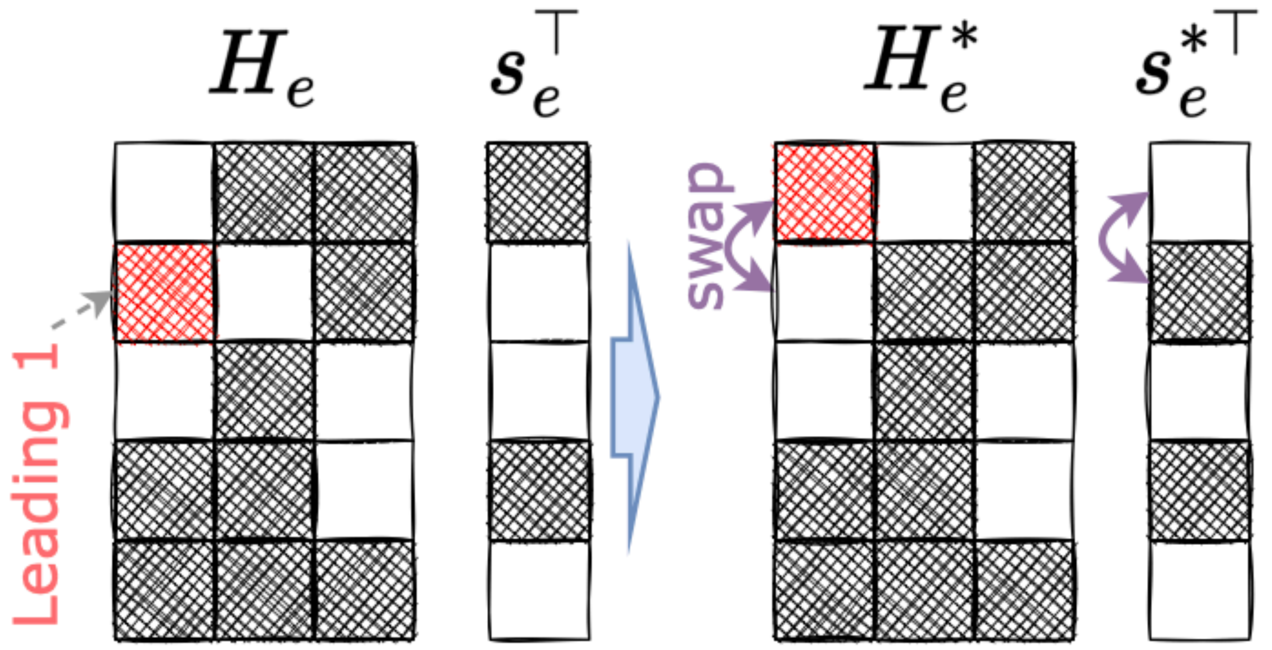
- Consider the following example:



- First, a leading 1 for each column in H matrix is identified.

The Gaussian Elimination Process

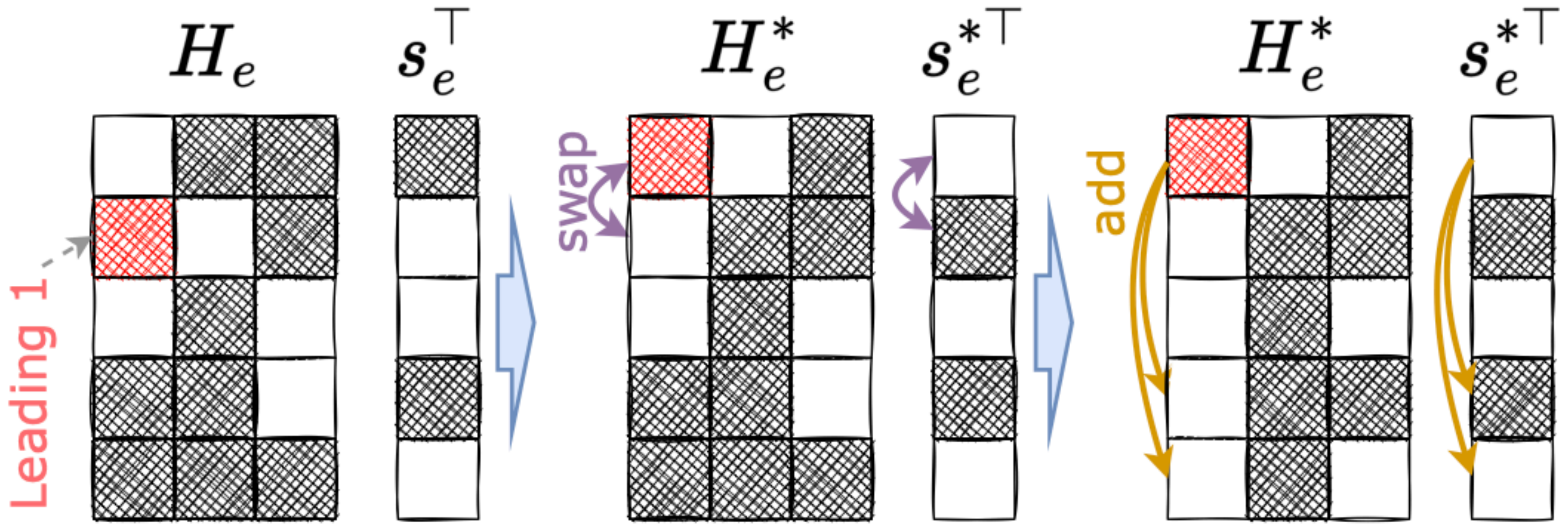
- Consider the following example:



- First, a leading 1 for each column in H matrix is identified.
- The identified row with leading 1 is swapped to place the 1 at the diagonal.
    - The same operation is performed over the syndrome vector s_e.

The Gaussian Elimination Process

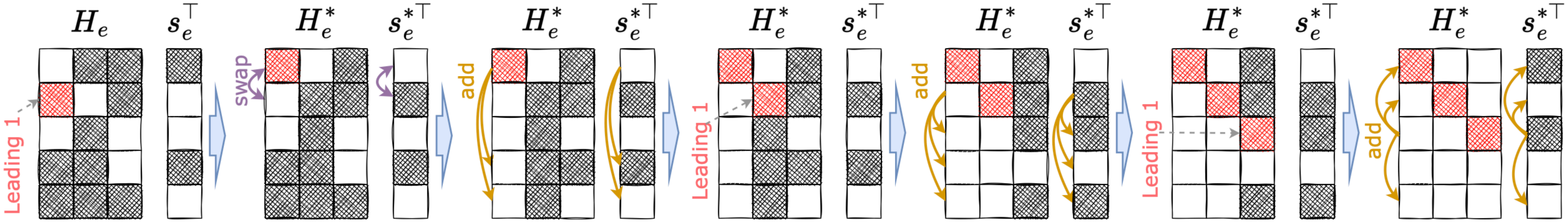
- Consider the following example:



- First, a leading 1 for each column in H matrix is identified.
- The identified row with leading 1 is swapped to place the 1 at the diagonal.
    - The same operation is performed over the syndrome vector s_e.
- Add (XOR) operations is carried out row-wise to clear out any remaining 1s for that column.
    - The same operation is performed over the syndrome vector s_e.

The Gaussian Elimination Process

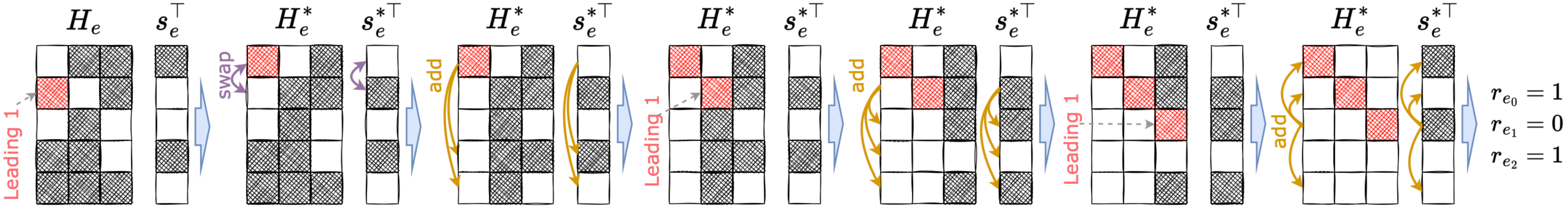
- Consider the following example:



- First, a leading 1 for each column in H matrix is identified.
- The identified row with leading 1 is swapped to place the 1 at the diagonal.
  - The same operation is performed over the syndrome vector s_e.
- Add (XOR) operations is carried out row-wise to clear out any remaining 1s for that column.
  - The same operation is performed over the syndrome vector s_e.
- Process is repeated until all columns are cleared.

The Gaussian Elimination Process

- Consider the following example:



- First, a leading 1 for each column in H matrix is identified.
- The identified row with leading 1 is swapped to place the 1 at the diagonal.
    - The same operation is performed over the syndrome vector s_e.
- Add (XOR) operations is carried out row-wise to clear out any remaining 1s for that column.
    - The same operation is performed over the syndrome vector s_e.
- Process is repeated until all columns are cleared.
- Resulting modified syndrome vector is the solution key for erased sequences.

Reducing the Computational Complexity of Gaussian Elimination

- In practice, GE is costly with computational complexity $O(n^3)$.
- To reduce its impact over the iterations, GE can be performed only once per frame.

Reducing the Computational Complexity of Gaussian Elimination

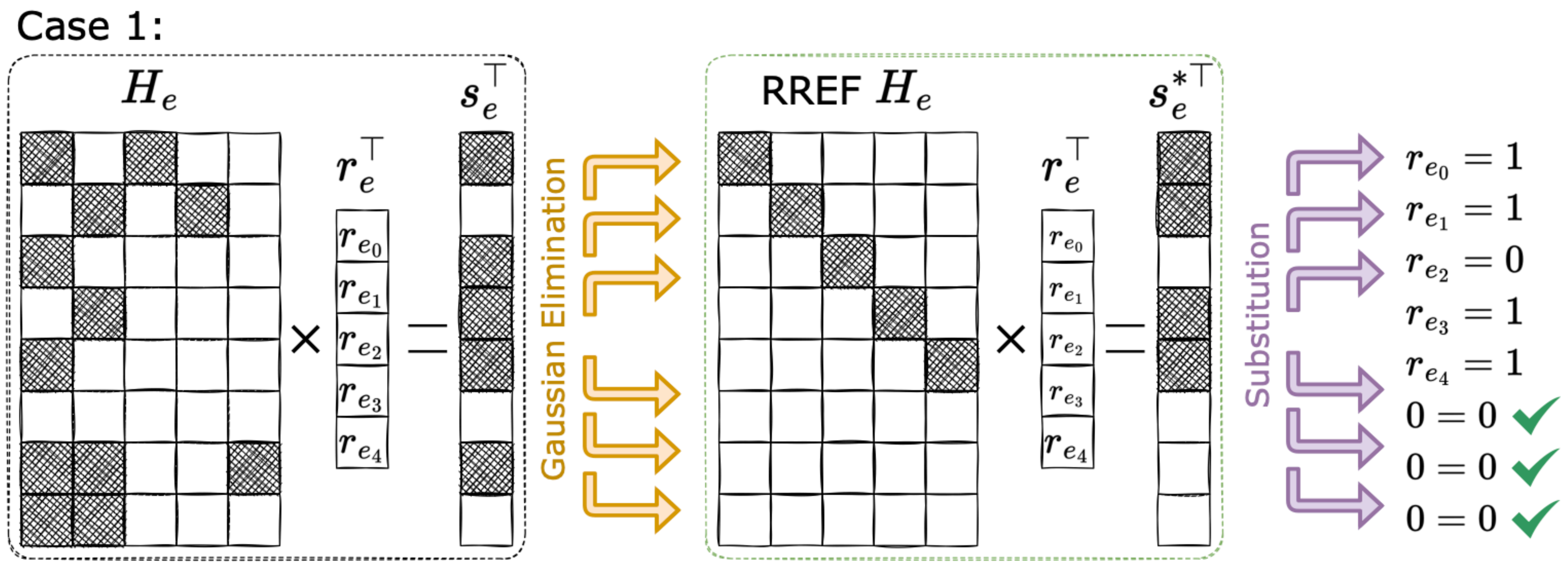
- In practice, GE is costly with computational complexity $O(n^3)$.
- To reduce its impact over the iterations, GE can be performed only once per frame.
- Add/swap operations of the EDGE can be stored in a matrix E.

Reducing the Computational Complexity of Gaussian Elimination

- In practice, GE is costly with computational complexity $O(n^3)$.
- To reduce its impact over the iterations, GE can be performed only once per frame.
- Add/swap operations of the EDGE can be stored in a matrix E.
- This way, the final erasure syndrome can be directly obtained by

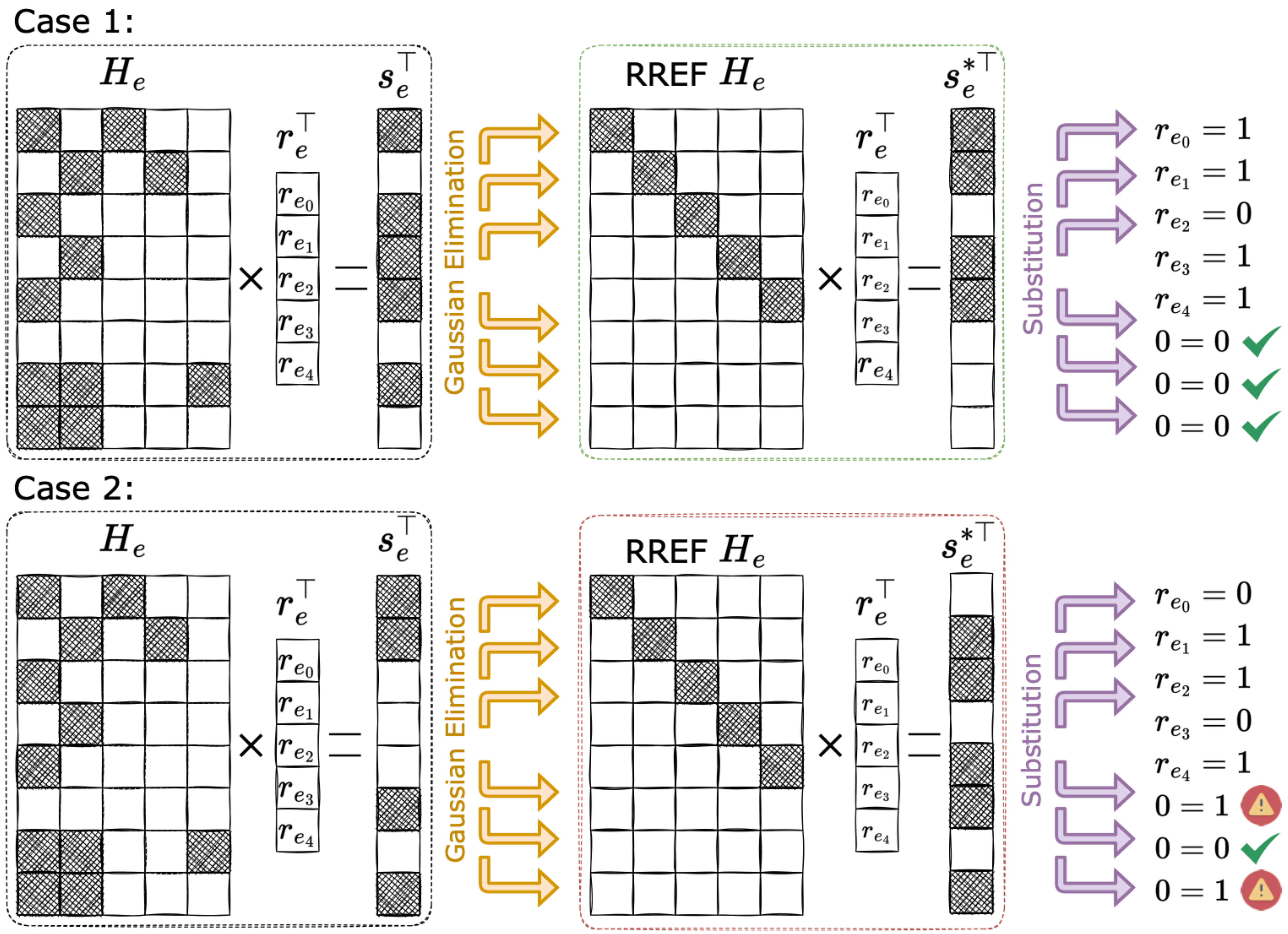
$$\mathbf{s}_{\mathbf{e}}^{*\top} = \mathbf{E} \cdot \mathbf{s}_{\mathbf{e}}$$

The GRAND-EDGE Algorithm Family



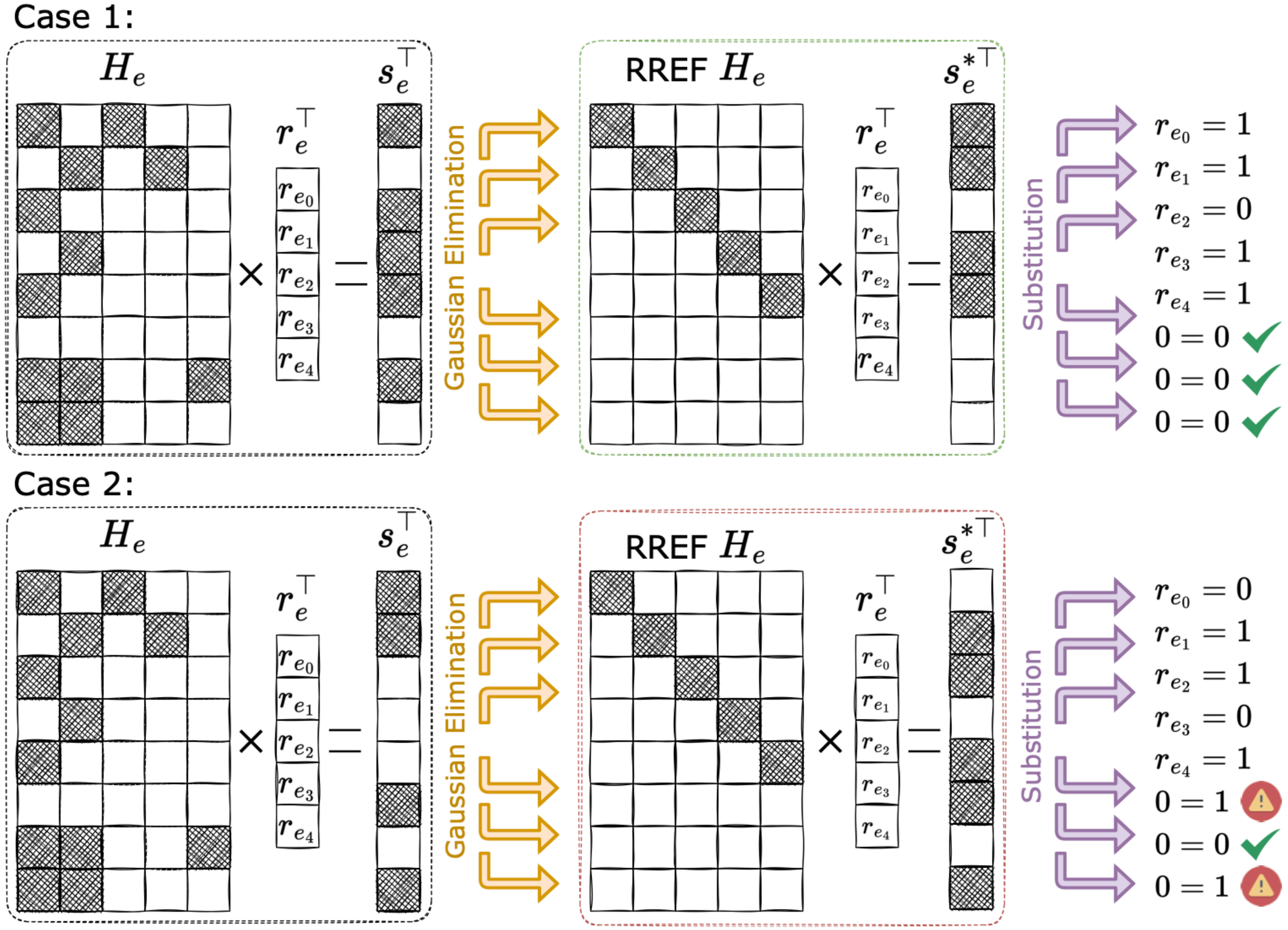
- Idea: Replace syndrome computation with EDGE subroutine.
- GE is acceptable only when no residue at the bottom matrix remains.
- Case 1: No residue, decoding acceptable, codeword is restored.

The GRAND-EDGE Algorithm Family



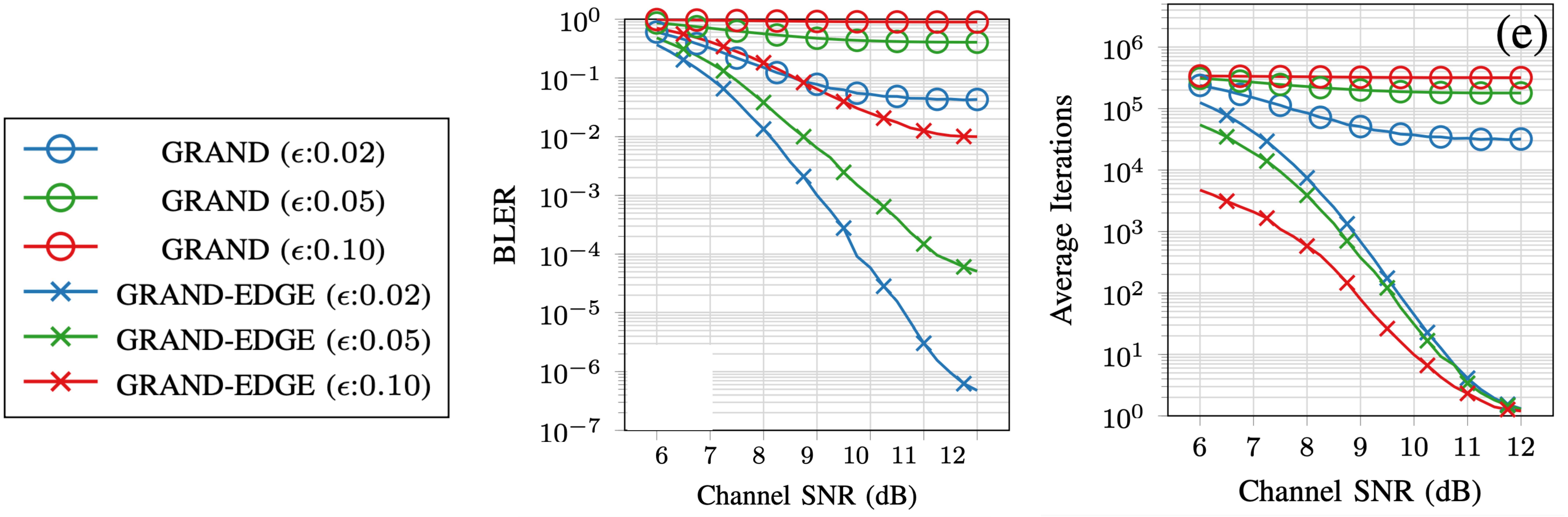
- Idea: Replace syndrome computation with EDGE subroutine.
- GE is acceptable only when no residue at the bottom matrix remains.
- Case 1: No residue, decoding acceptable, codeword is restored.
- Case 2: Residue remains due to channel errors, perform GRAND and try again.

The GRAND-EDGE Algorithm Family

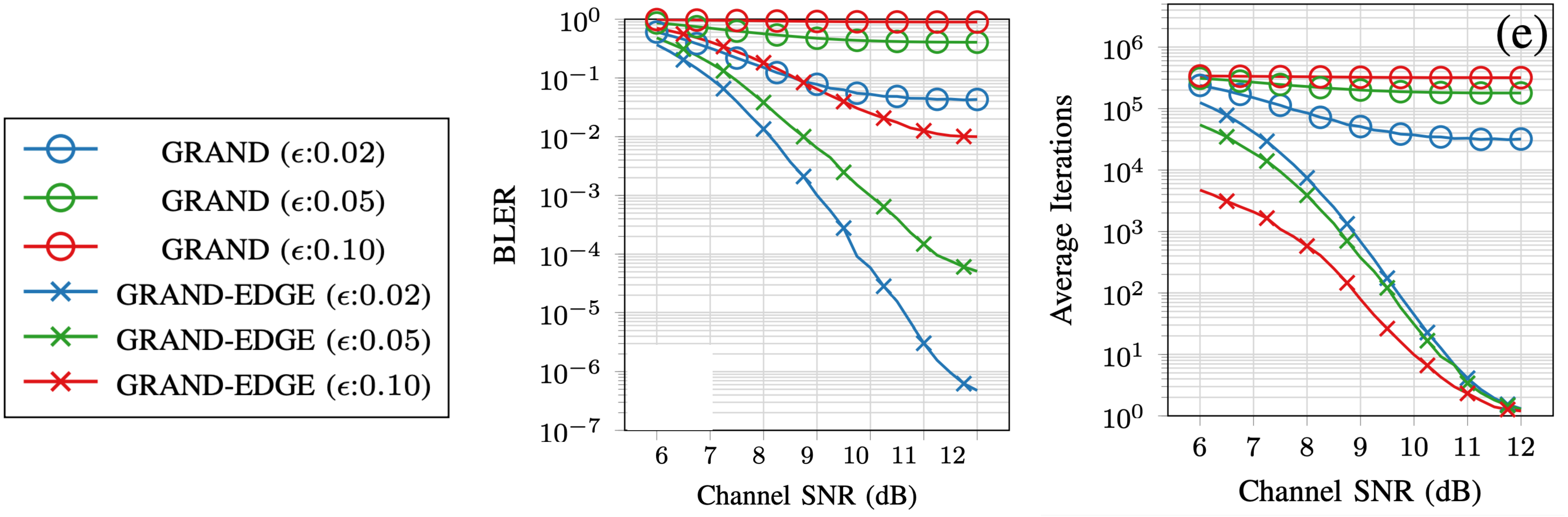


- Idea: Replace syndrome computation with EDGE subroutine.
- GE is acceptable only when no residue at the bottom matrix remains.
- Case 1: No residue, decoding acceptable, codeword is restored.
- Case 2: Residue remains due to channel errors, perform GRAND and try again.
- Keep iterating until EDGE passes or a maximum number of iterations is reached.

Performance Assessment: GRAND-EDGE



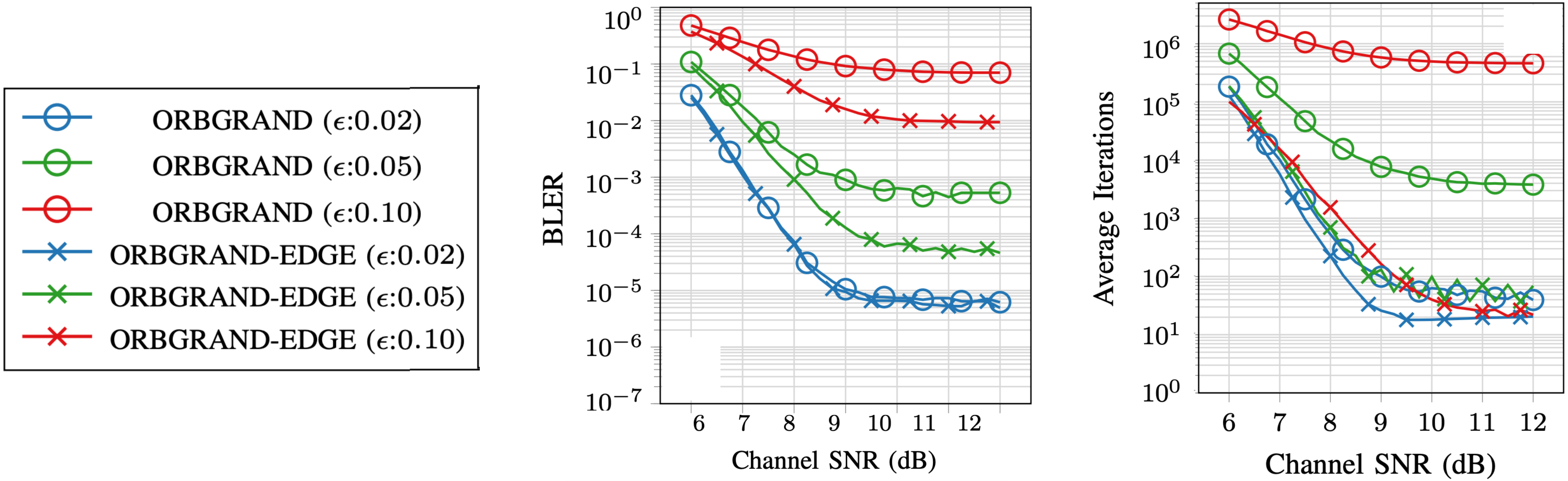
Performance Assessment: GRAND-EDGE



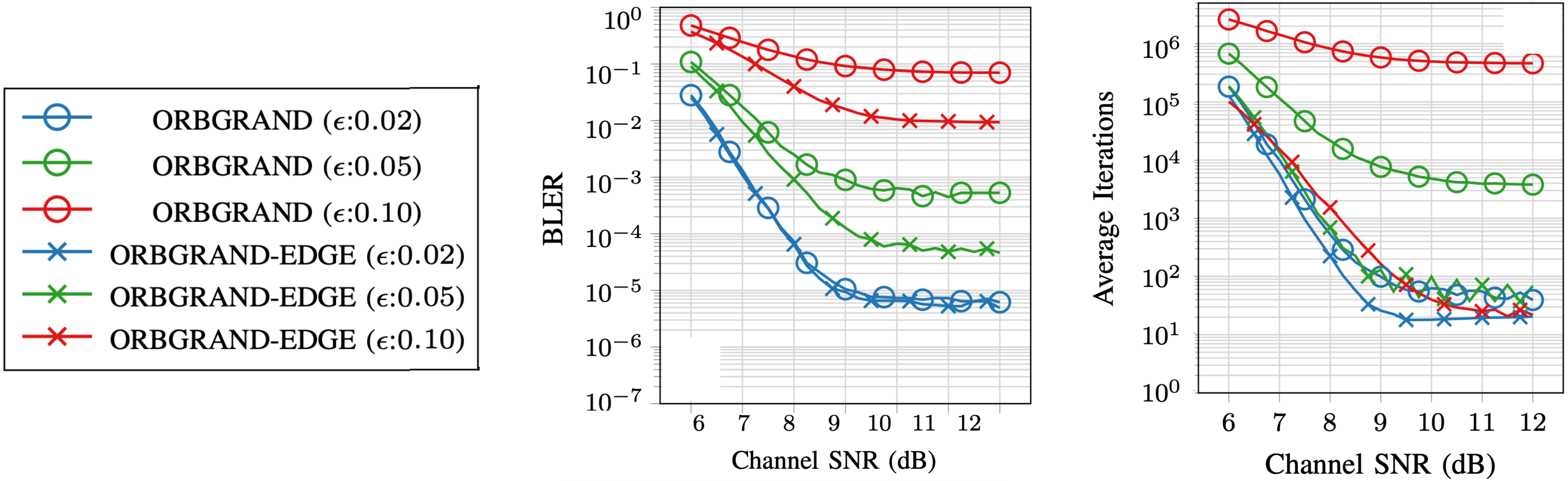
- Performance evaluation was carried out using Random Linear Codes.
- GRAND-EDGE and ORBGRAND-EDGE are created and simulated.
- Block Error Rate improvement of five orders of magnitude.
- Average number of iterations improves more than five orders of magnitude.

F. Ercan et al.

Performance Assessment: ORBGRAND-EDGE



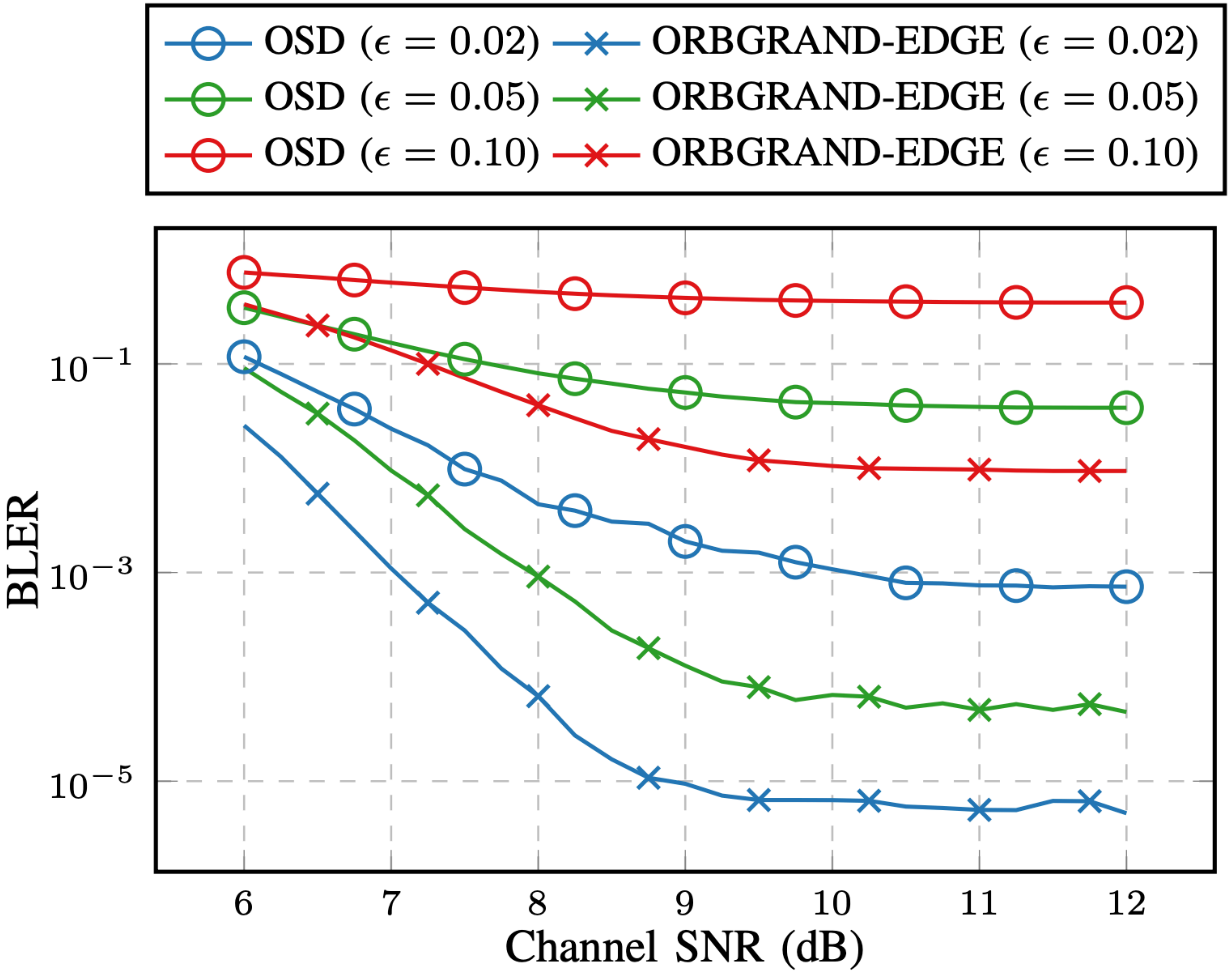
Performance Assessment: ORBGRAND-EDGE



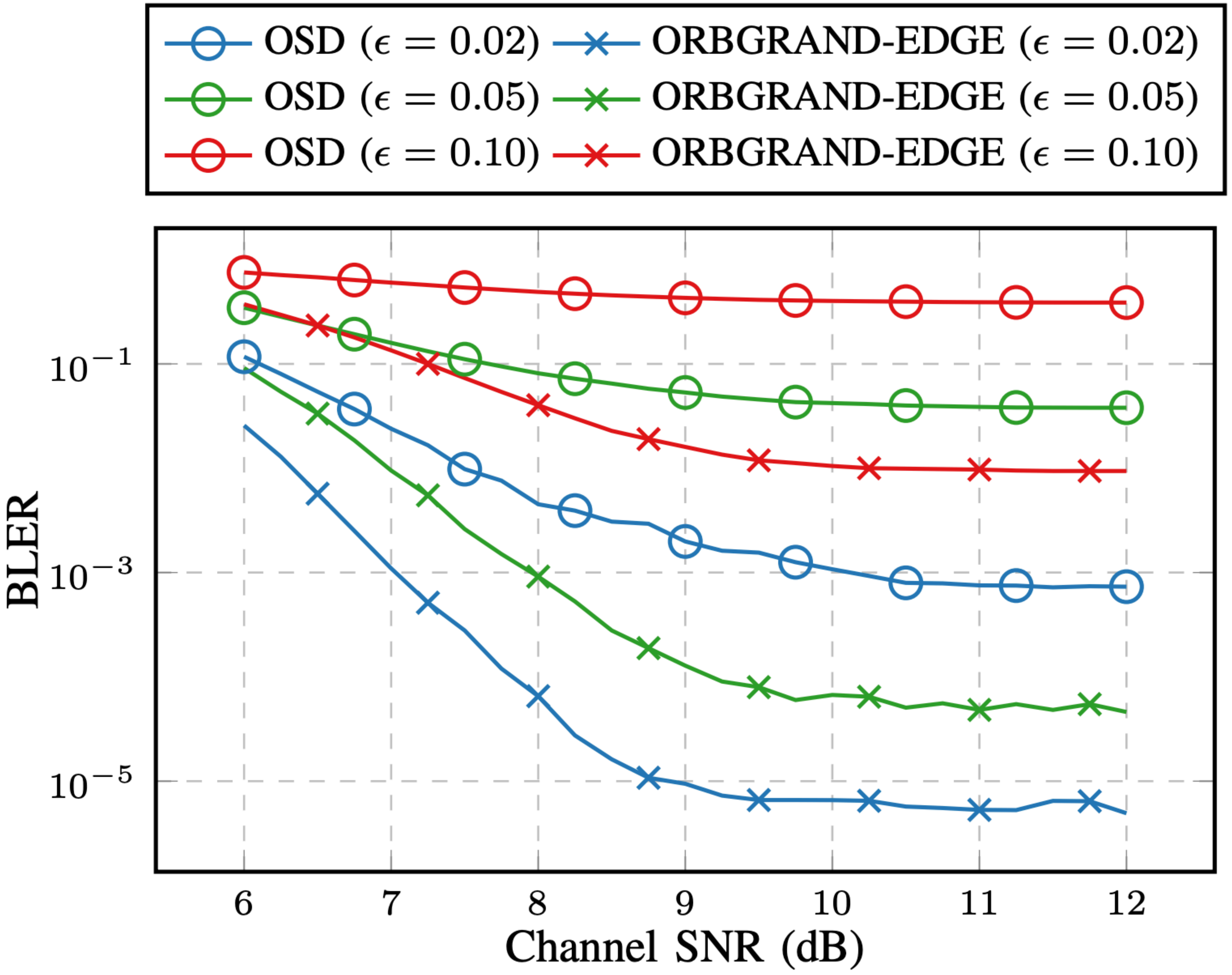
- BLER performance: Up to one order of magnitude improvement
- Average number of iterations: Up to one order of magnitude improvement.
- ORBGRAND scopes flipping bits in a limited way which in turn limits the performance improvement.
- The EDGE algorithm can be extended to any other variant of GRAND.

F. Ercan et al.

Performance Assessment: GRAND-EDGE vs. OSD



Performance Assessment: GRAND-EDGE vs. OSD



- RLC[128,105] code is used.
- OSD performs GE over k columns, whereas GRAND-EDGE does so over $N-k$ columns (far less for rates of interest).
- OSD requires a new GE for each iteration, whereas GRAND-EDGE requires only one per iteration.
- GRAND-EDGE is shown to have up to 3 orders of magnitude better BLER than that of OSD.

Conclusion

- We introduced an adversarial model whereby a jammer randomly destroys data by overpowering the channel.

Conclusion

- We introduced an adversarial model whereby a jammer randomly destroys data by overpowering the channel.
- We showed that the syndrome calculation block can be generalized into an erasure decoding, using Gaussian Elimination.
- The yielding algorithm is called the GRAND-EDGE.

Conclusion

- We introduced an adversarial model whereby a jammer randomly destroys data by overpowering the channel.
- We showed that the syndrome calculation block can be generalized into an erasure decoding, using Gaussian Elimination.
- The yielding algorithm is called the GRAND-EDGE.
- Any variant of GRAND can be used towards the proposed enhancement.
- Simulation results with both hard- and soft-information variants demonstrate substantial gains in error performance and computational complexity under adversarial constraints.

Thank you!