

Sharing a secret in Pub/Sub

Javier Munster

Outline

- Why share secrets?
- Pub/Sub Overview
- Goal and Challenges
- Related Academic Solutions
- Techniques:
 1. (k, n) Threshold Value
 2. SGX

Why share secrets?

Encryption

- Symmetric
- Asymmetric (public/private keys)
- Homomorphic Cryptosystems
- ASPE
- Multiple Layer Commutative Encryption
- Attribute-Based
- Predicate
- Functional

Confidentiality

- Events contain sensitive data
- Health records, paid content

Access Control

- Control what each node can do
- Prevent attacks (DoS, etc)

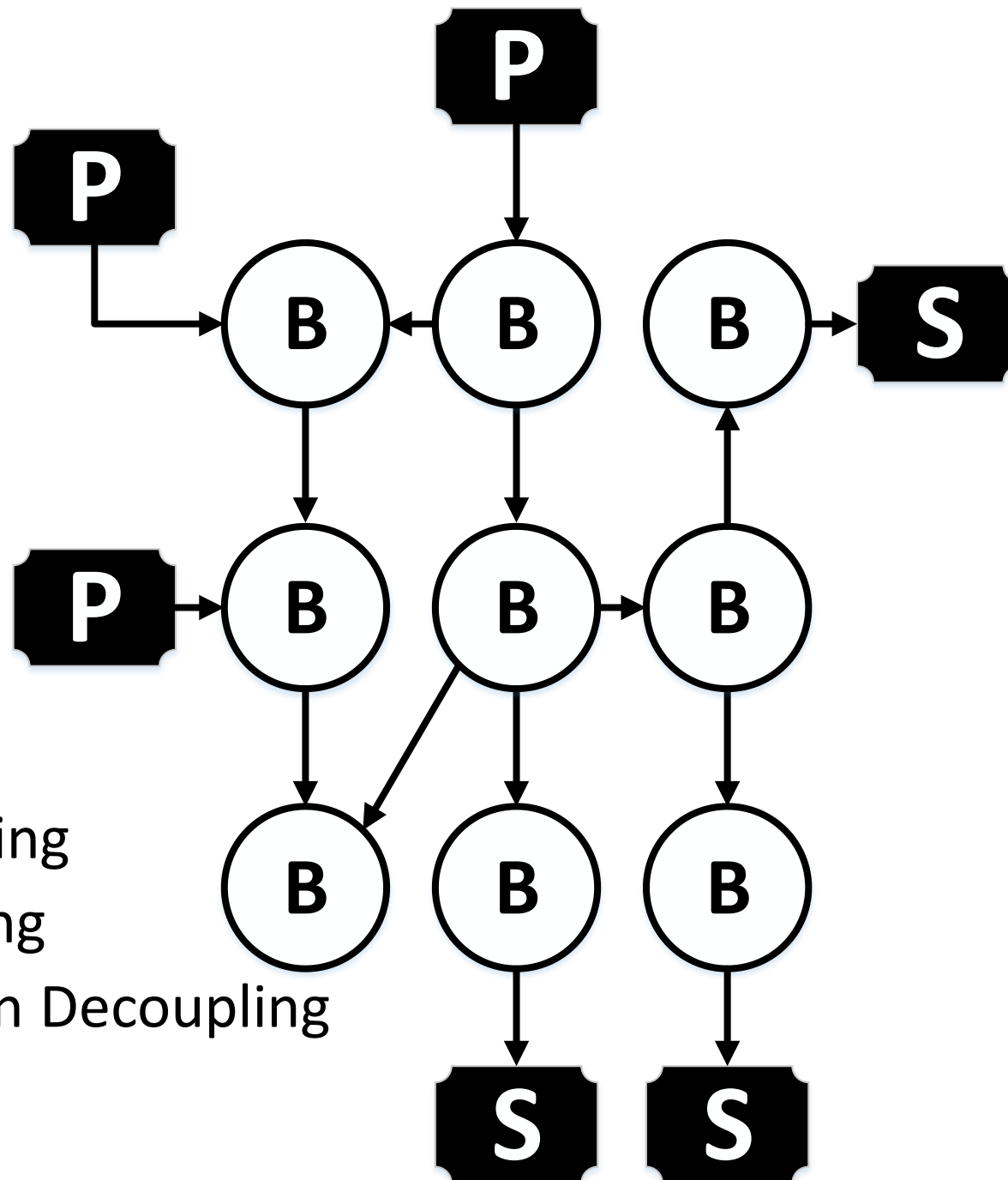
Anonymity

- Onion Routing

Overview

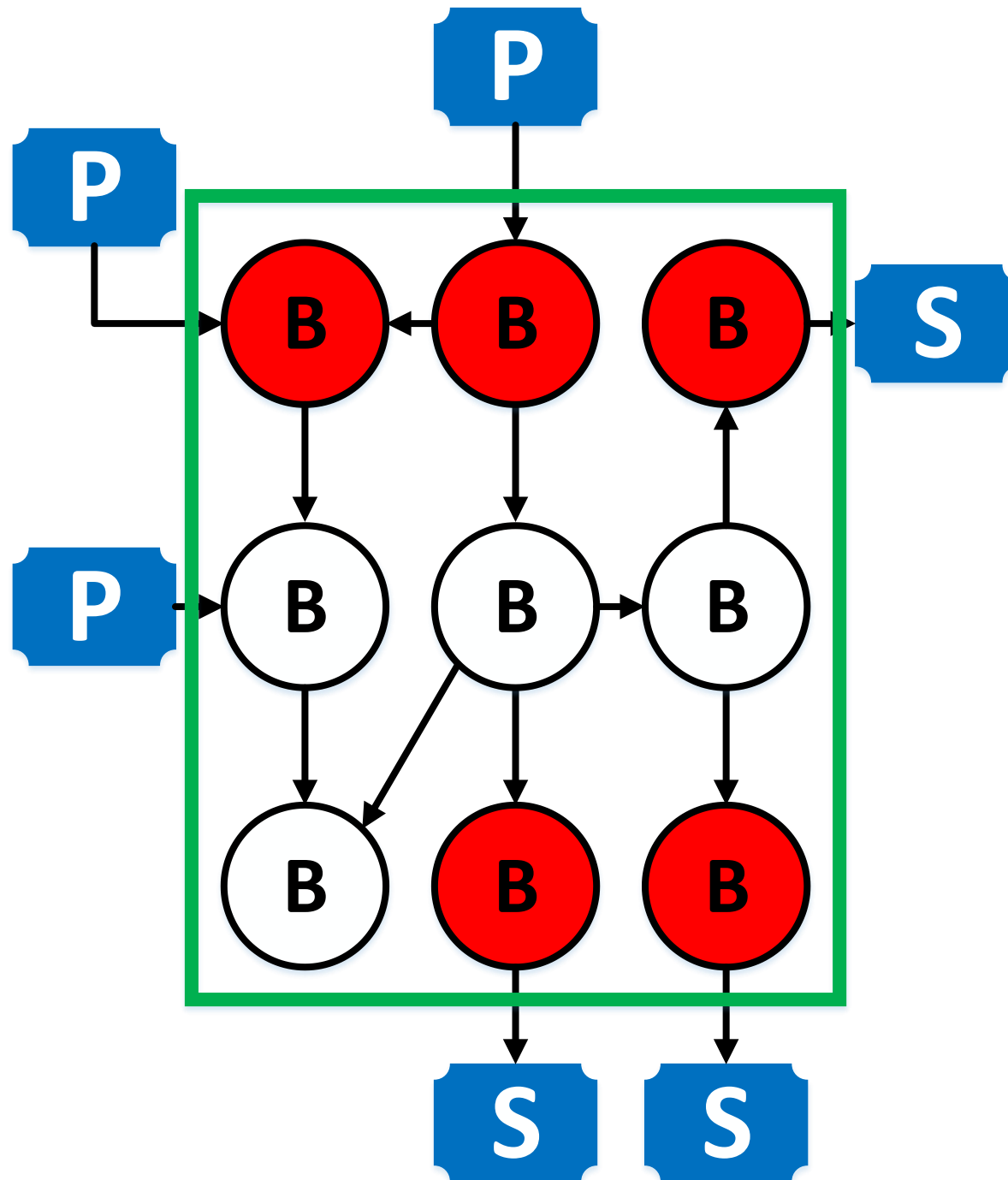
- Publishers
- Brokers
- Subscribers
- Events

- Space Decoupling
- Time Decoupling
- Synchronization Decoupling



Terms

- Broker Network
- Principals
- Edge Brokers



Goal

- **Disseminate a secret/key from a publisher to a subset of subscribers through an untrusted broker network**
- Minimize trust assumptions required
- Minimize number of components needed

Threat Models

1. Honest-but-curious brokers
 - Trusted to execute instructions correctly
 - Not trusted with the contents of the data
2. Colluding brokers
 - Brokers share knowledge

Orthogonal Problems

- Untrusted subscribers (digital copyright problem)
- Byzantine brokers (redundant routing paths)
- Send/receive omission failures, eavesdropping, dropped packets (solved by industry standard techniques: TLS, TCP)
- Scalable key management (key trees, etc)

Related Work

- Trusted out-of-band solutions that exist independent of the broker network
 - Trusted [de]centralized key manager grants keys upon valid request by a principal
 - Trusted certificate authority maintains public keys for all principals
- Pros:
 - Infrastructure to support this already exists (internet)
- Cons:
 - Points of failure vs number of nodes that must be trusted with key tradeoff



(k, n) Threshold Scheme – Value Splitting

- Idea: *How to Share a Secret* by Adi Shamir, 1979 (edited by Ron Rivest)
- Divide data D into n pieces D_1, \dots, D_n such that:
 - Knowledge of k or more D_i pieces makes D easily computable
 - Knowledge of any $k-1$ or fewer D_i pieces leaves D completely undetermined

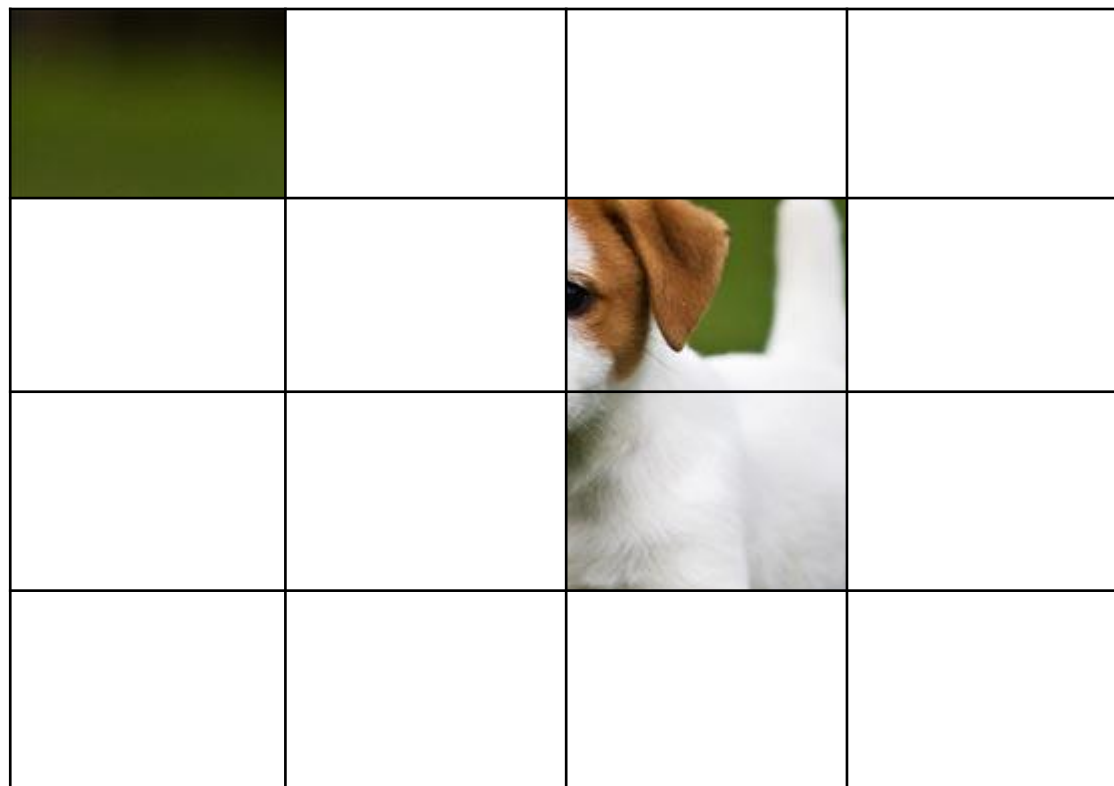
Value Splitting

Value Splitting

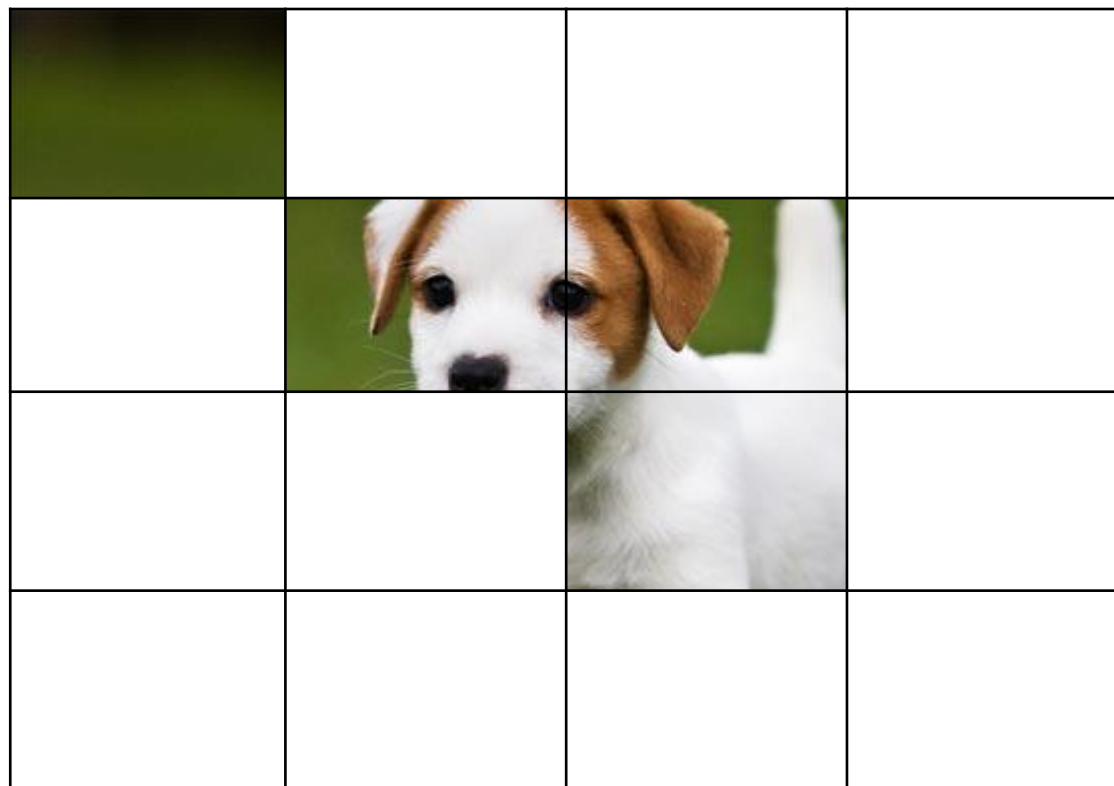
Value Splitting

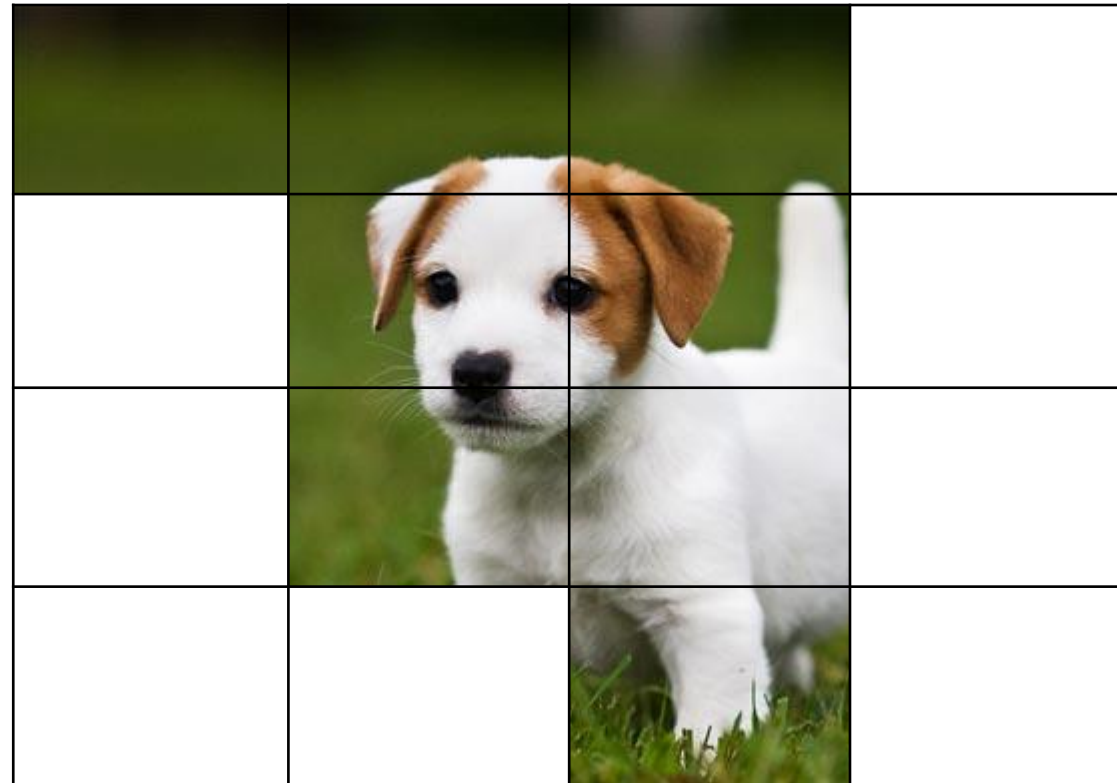
Value Splitting



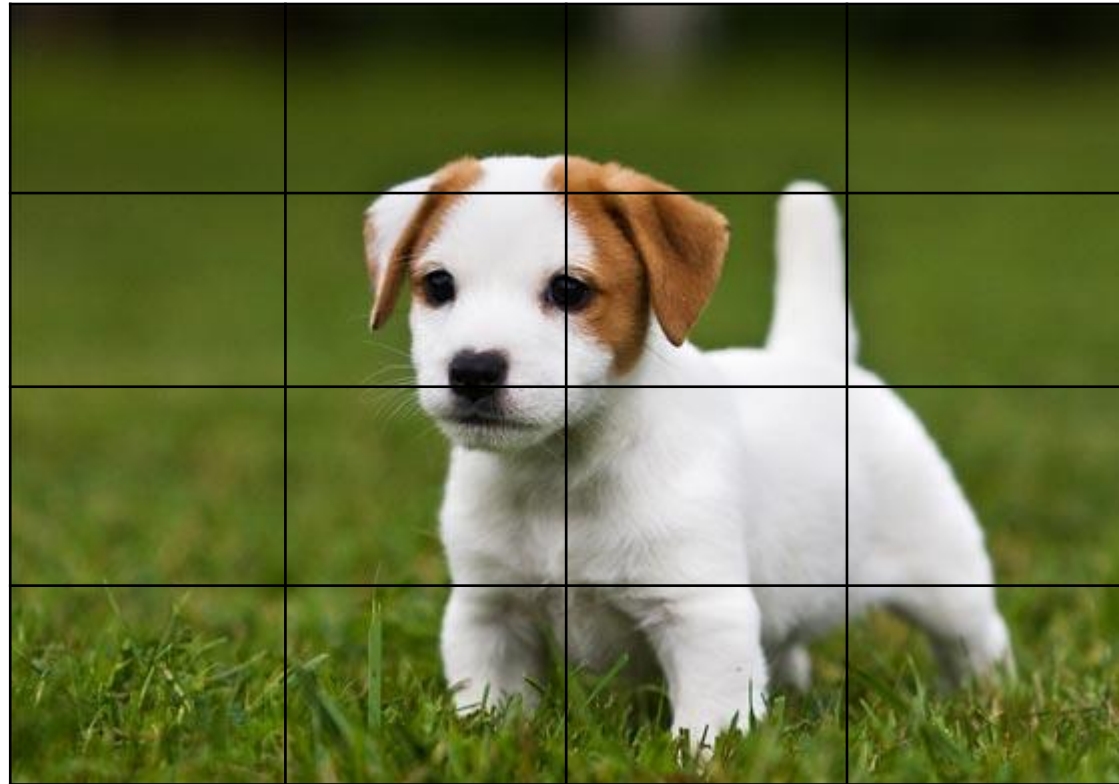
Value Splitting



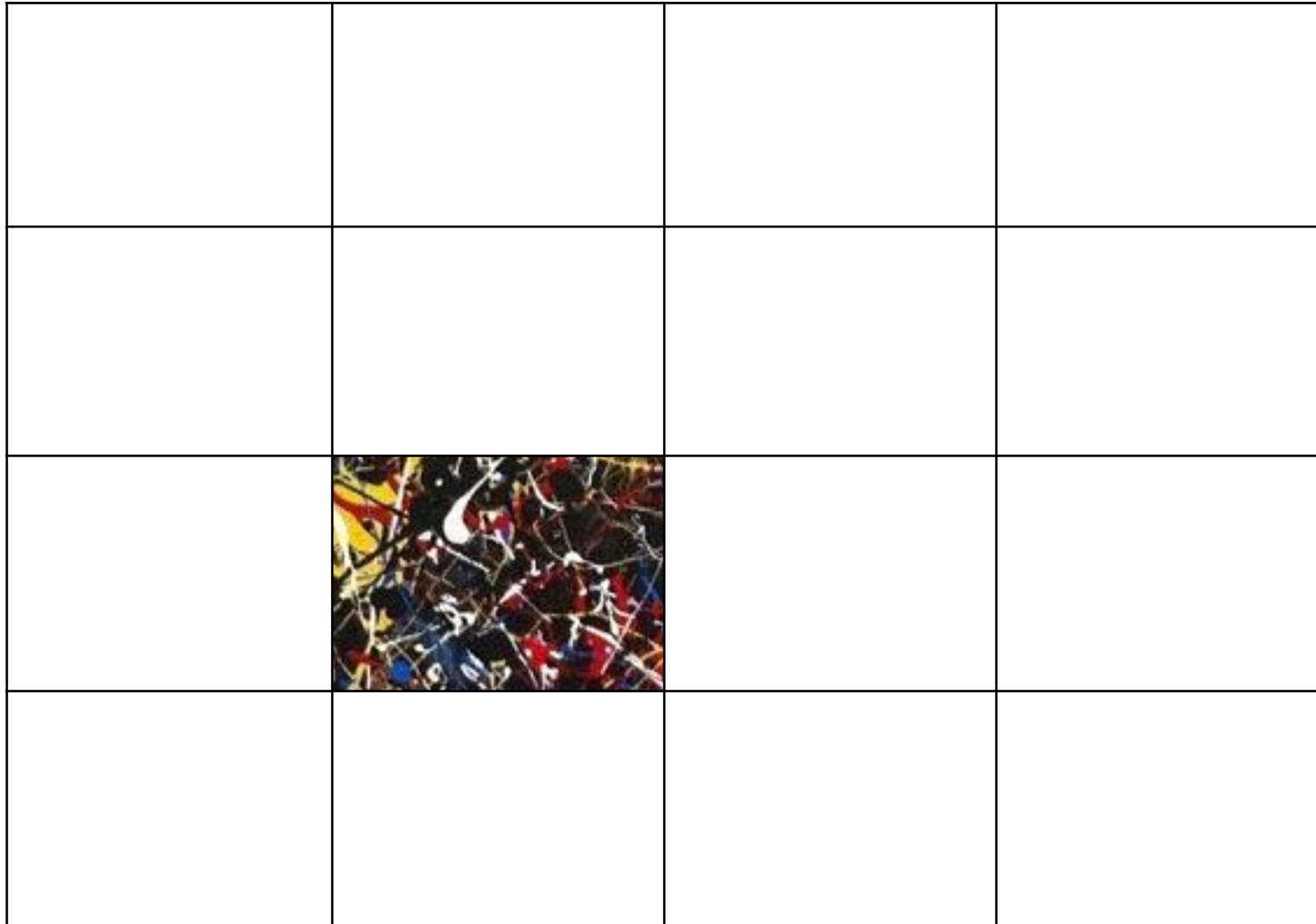
Value Splitting



Value Splitting



Value Splitting



Value Splitting

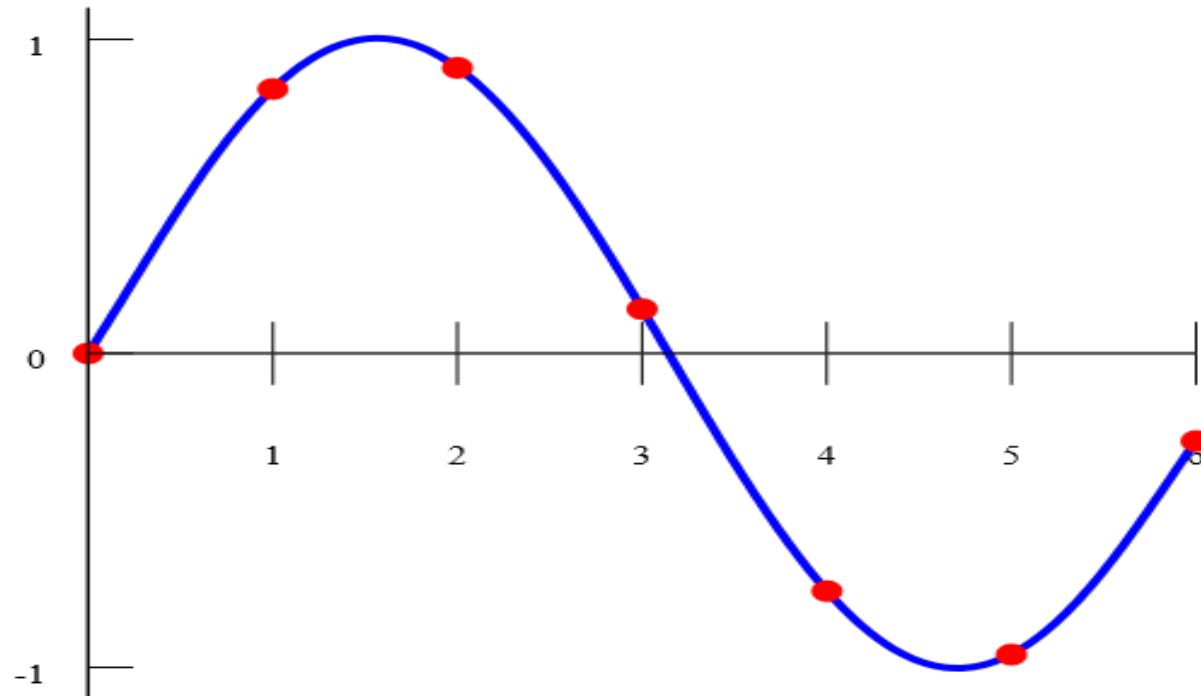


Value Splitting



(k, n) Threshold Scheme

- Based on polynomial interpolation
 - Given k unique points
 - There exists a unique polynomial of degree at most $(k-1)$ that goes through all points



(k, n) Threshold Scheme

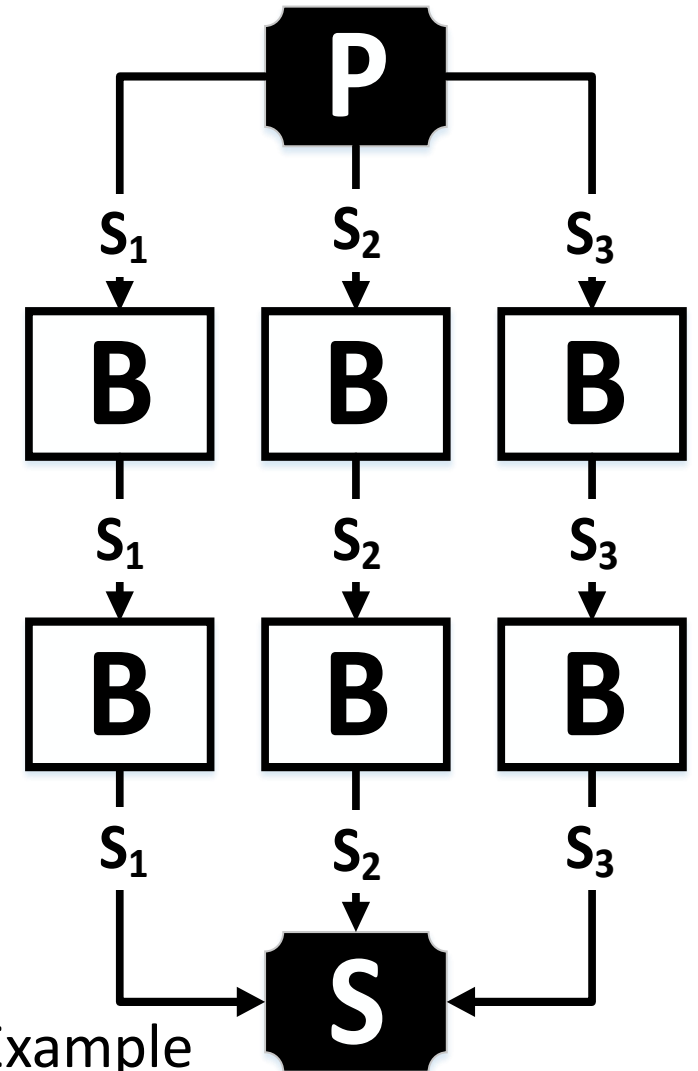
- Splitting D:
 - Pick a random $(k-1)$ degree polynomial
 - $q(x) = a_0 + a_1x + \dots + a_{k-1}x^{k-1}$
 - Set $a_0 = D = q(0)$
 - Output: $D_1 = q(1), \dots, D_i = q(i), \dots, D_n = q(n)$
- Regenerating D:
 - Retrieve a_i values through polynomial interpolation ($O(n \lg(n))$ operation)
 - Regenerate q function
 - Output: $q(0)$

(k, n) Threshold Scheme in Pub/Sub

- MSRG alumni Young Yoon, Assistant Professor at Hongik University
 - *Reliable and Confidential Messaging on Publish/Subscribe Broker Overlays*
- Unpublished work, please keep confidential

(k, n) Threshold Scheme in Pub/Sub

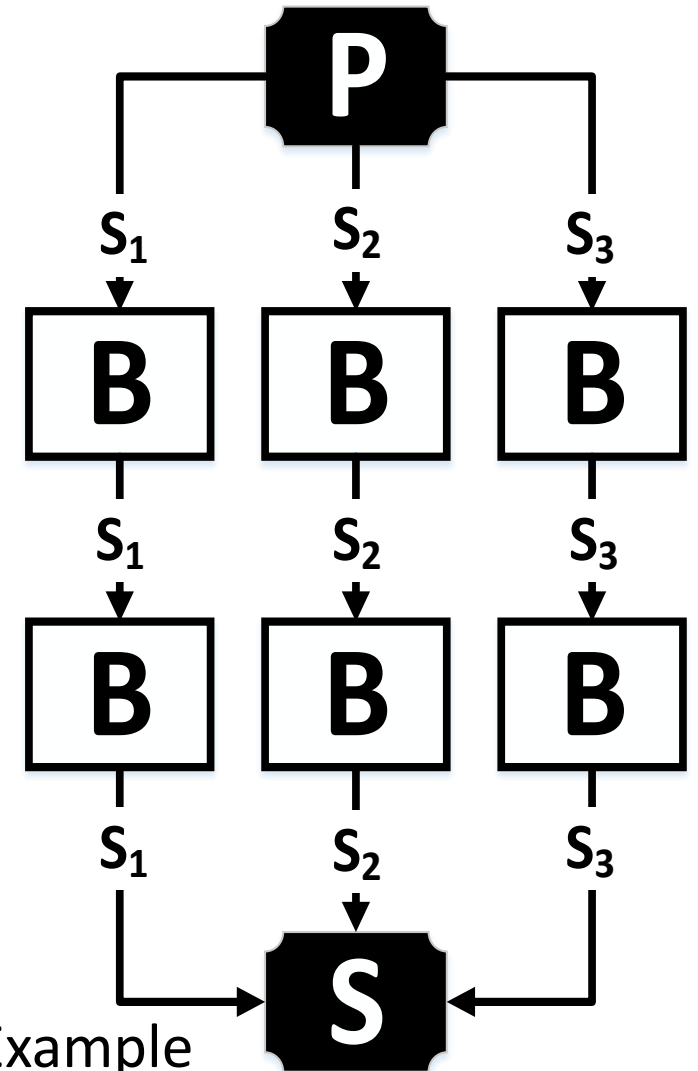
- Publisher splits secret S into pieces and sends it to subscribers through broker network
- Subscribers receive pieces and regenerate S
- Brokers are none the wiser



(3, 3) Threshold Scheme Example

(k, n) Threshold Scheme in Pub/Sub

- Multiple unique routing paths must exist where no node is in more than one path
- Tolerates up to $(k-1)$ leaky/colluding brokers in the worst case
 - Up to $(k-1)$ of the unique paths may be compromised

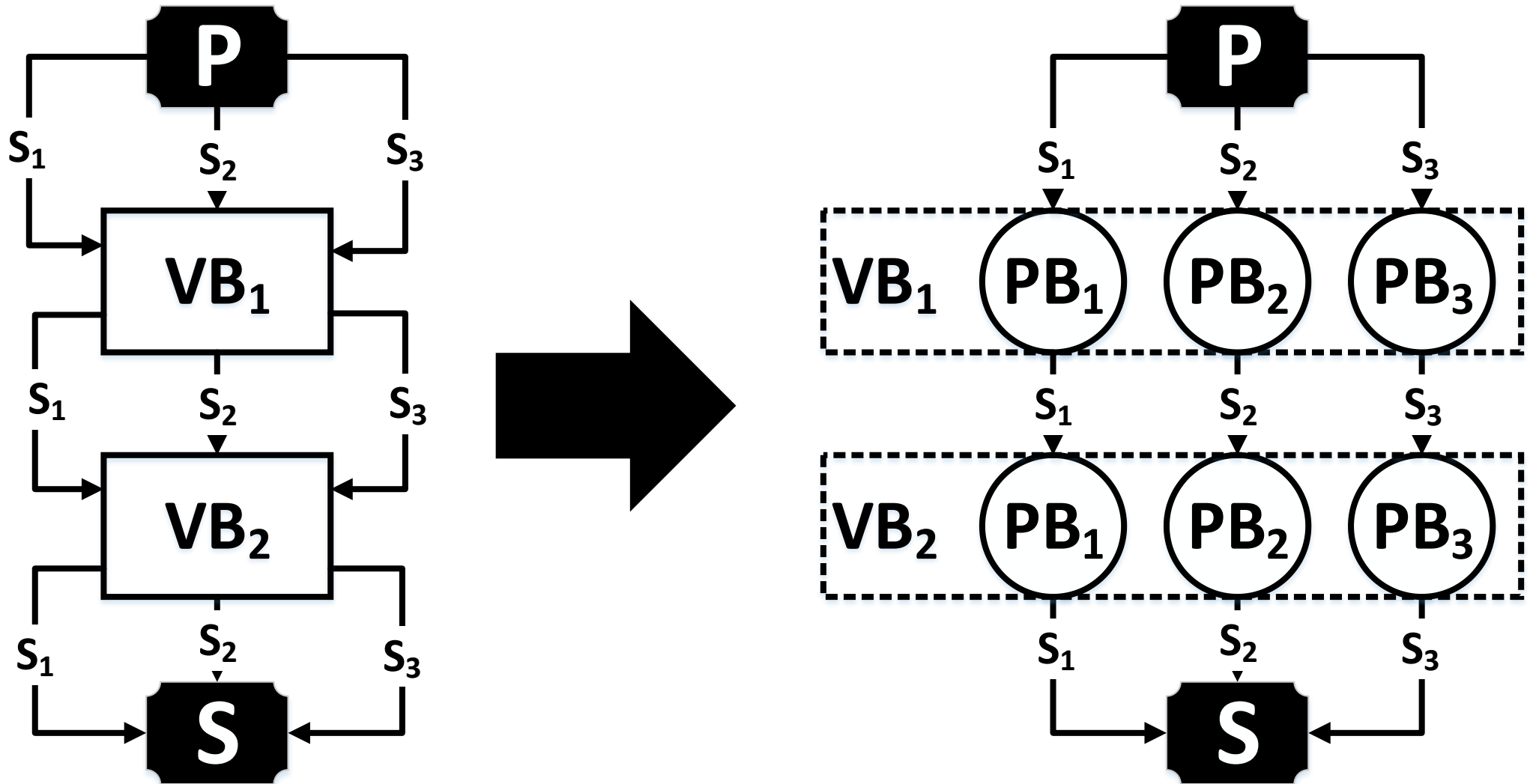


$(3, 3)$ Threshold Scheme Example

(k, n) Threshold Scheme in Pub/Sub

- Routing solutions:
 1. Find these paths in the broker network
 - Consult a topology manager
 - Construct broker network with multiple path requirement
 - Create routing paths on demand as part of key sharing operation
 2. Leverage broker replications within system

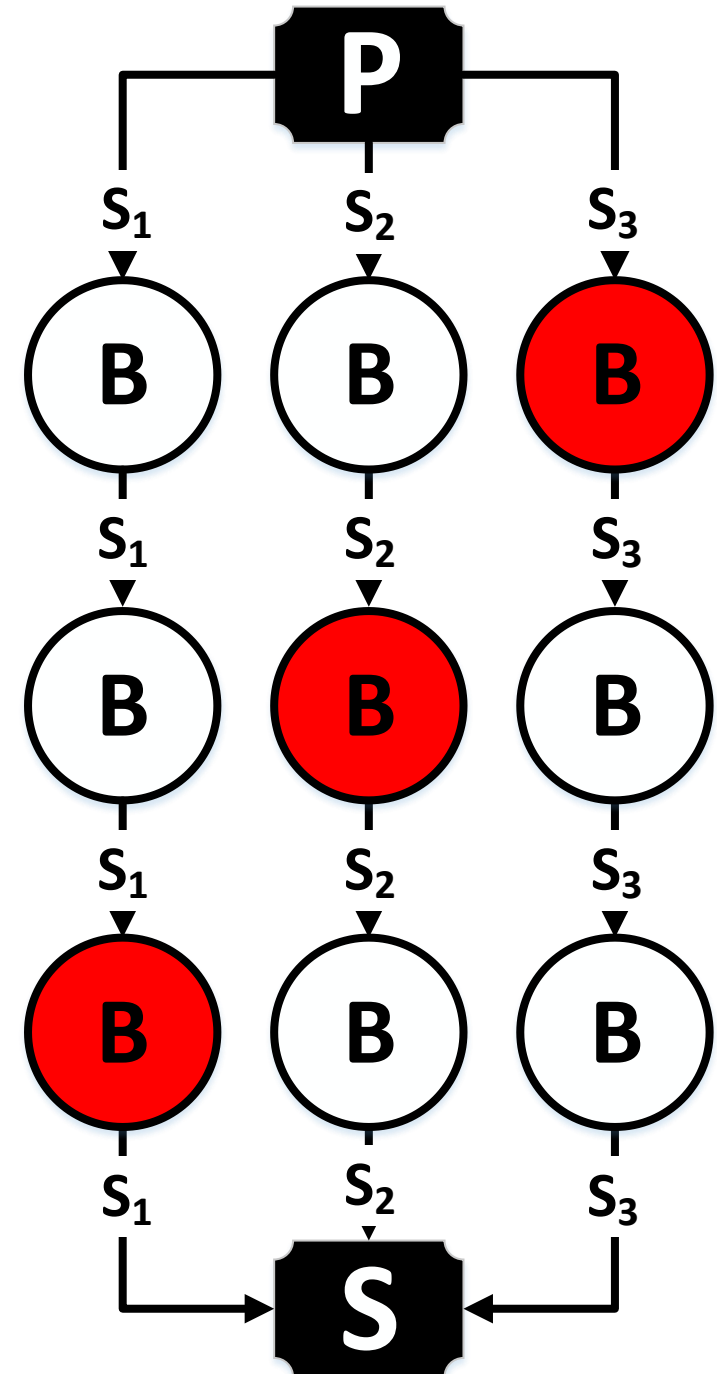
Leveraging Replication



(3, 3) Threshold Scheme

Further securing our secret

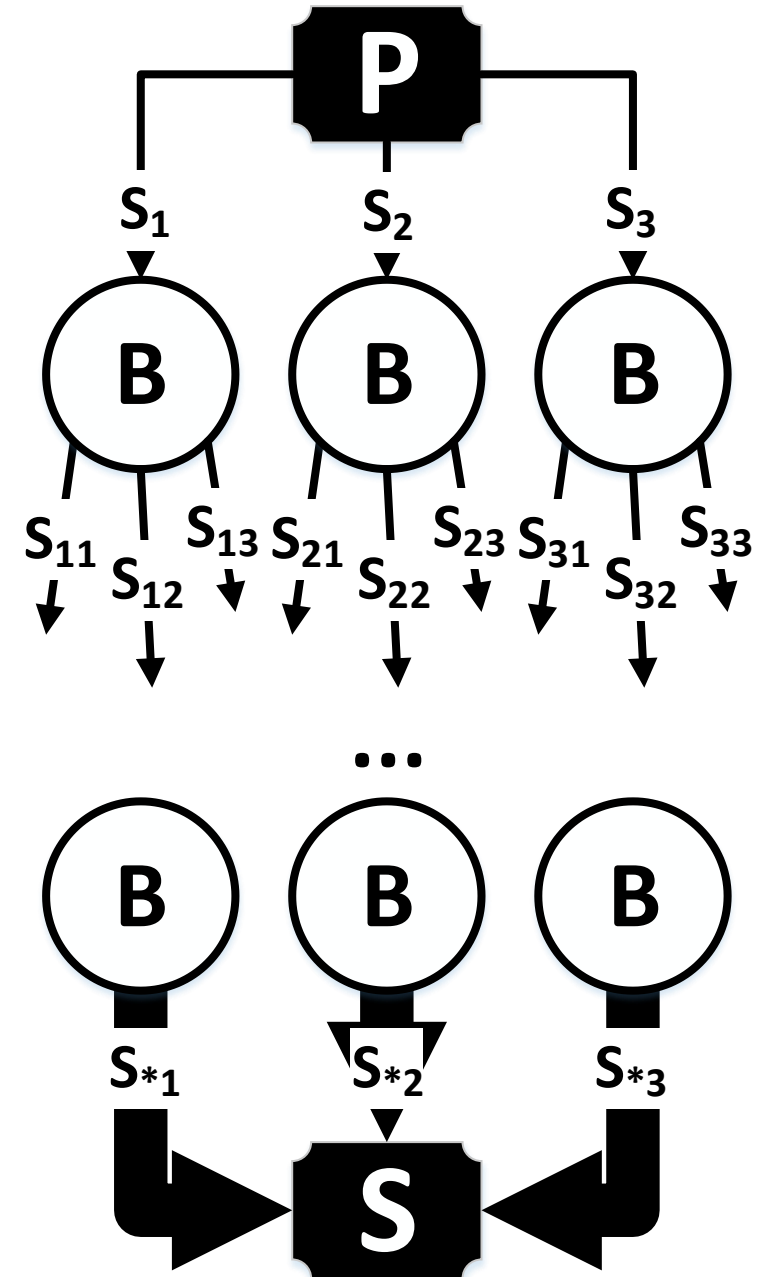
- Previous systems cannot tolerate a compromised broker in each unique path



Further securing our secret

- Apply the threshold scheme at each level
- Proof by contradiction (probably)
- Gain: each level must be fully compromised to leak secret
- Cost: each level exponentially increases number of splits
- Subscriber receives PB^L values

P = # of paths, B = # of brokers per level,
 L = # of levels



SGX

SGX

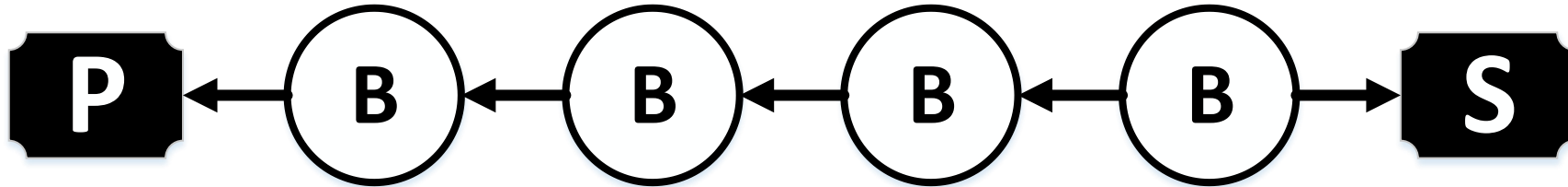
- Specific hardware can create trusted execution environments (**enclaves**)
- Data within enclave is encrypted to anything outside that enclave
- The code being executed is NOT private
- **Attestation**: magic used by an enclave to prove to other entities that itself is indeed a valid enclave (uses a trusted service)
- During attestation, a secure channel is established between enclave and other entity
- Can SGX be used to share a secret in pub/sub?

SGX

- Lets us trust enclaves within brokers
 - Assuming Intel can be trusted
- Outside the enclave, secret is encrypted
- Inside enclave, secret is decrypted
- Secure channel lets us transmit secret

Naïve Solution

- Daisy chain attestations to ensure end-to-end confidentiality when delivering key

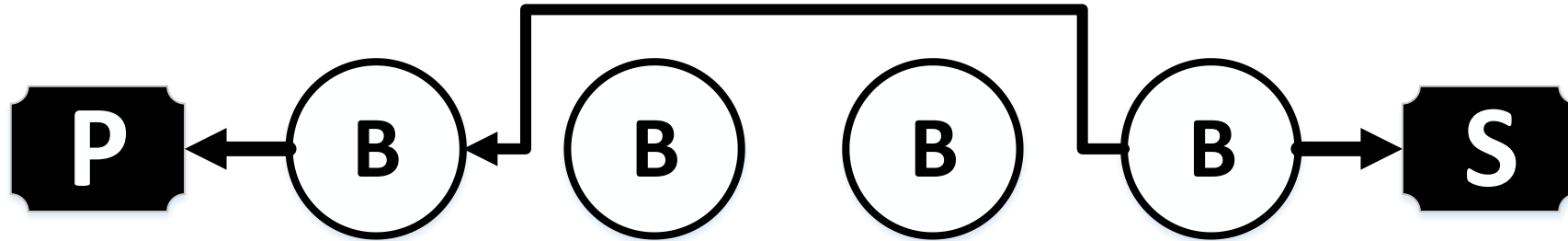


of attestations = Principals + B - 1

Optimizations on Naïve Solution

- Daisy chain brokers once when bringing up the network
- Same number of attestations, but requires only one attestation per newly connected principal

Open Solution (WIP)



- Only edge brokers attest
 - Intermediate brokers simply forward message
- # of attestations = Principals + (Edge Brokers - 1)