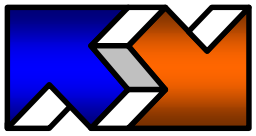
UNIVERSITY OF TORONTO

MIDDLEWARE SYSTEMS RESEARCH GROUP MSRG.ORG



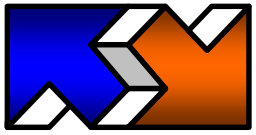
Securing Publish/Subscribe

M.A.Sc. Candidate: Javier Munster

Thesis Supervisor: Hans-Arno Jacobsen

Department of Electrical and Computer Engineering

University of Toronto



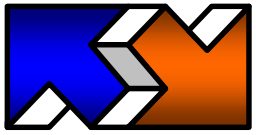
Agenda

- Where is pub/sub used?
- What is pub/sub?
- Why use pub/sub?
- Current state-of-the art summary
- HyShare

Pub/Sub in the Wild



Pub/Sub Applications



Security Perspective

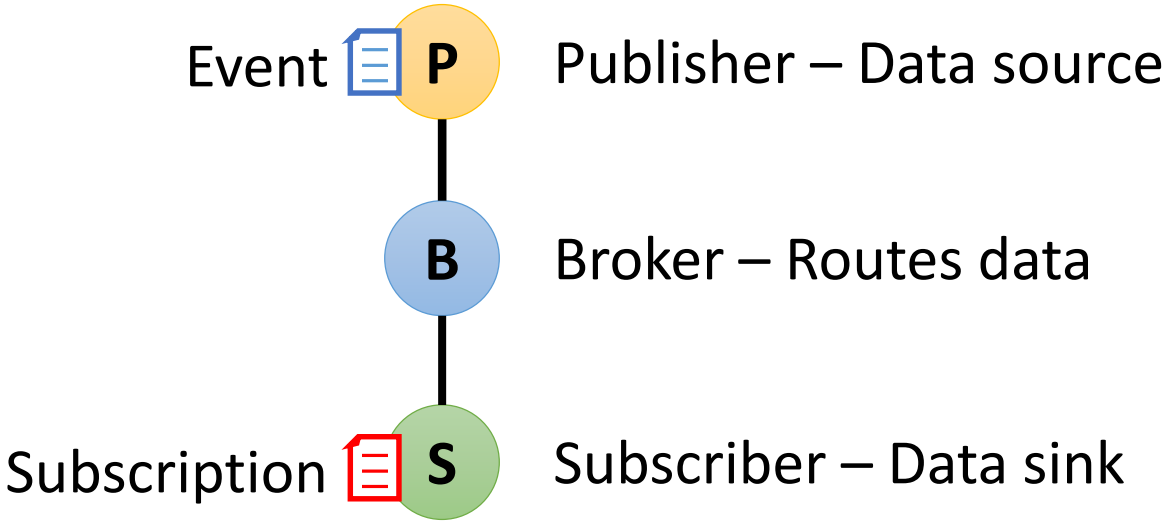
- **Confidentiality**
  - Sensitive data is being processed
- **Authorization**
  - Placing limits on who can do what within the system
- **Anonymity**
  - Sensitive data must be analyzed
  - Identity of data sources and sinks must be protected
- **Integrity**
  - Critical infrastructure must always be available
  - Attack resiliency

Contributions of Thesis Work

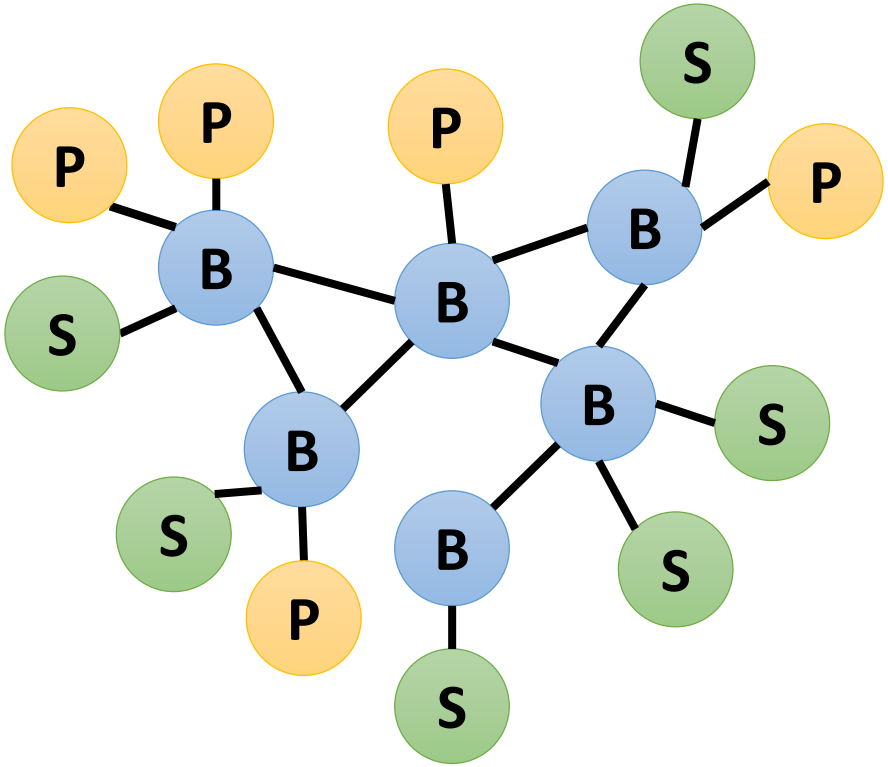
1. Categorization and analysis of the existing pub/sub security research[1]
2. Identification and analysis of the limitations of the existing approaches[1]
3. HyShare – A novel secret sharing solution used to ensure privacy[2]

1 – To be published in ACM CSUR
2 – Published in DEBS 2018

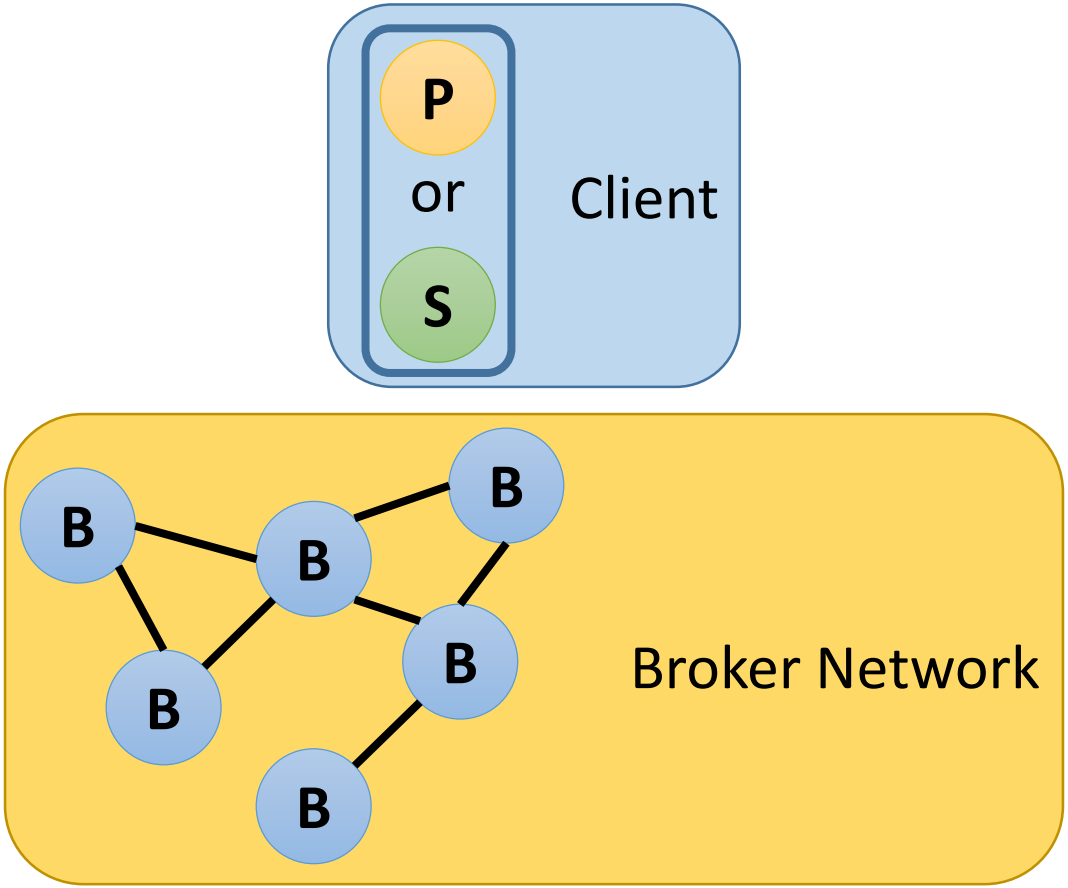
What is Publish/Subscribe?



Publish/Subscribe Example



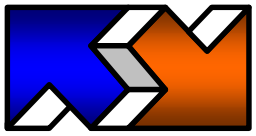
Pub/Sub Terminology



Subscription Anatomy

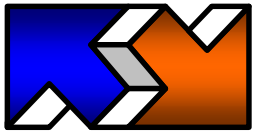
1. Topic-Based Pub/Sub:
   - All events belong to a topic
   - Subscriptions specify the topic subscriber is interested in
2. Content-Based Pub/Sub:
   - Events include attribute-value pairs
   - Subscriptions contain a subset of attributes subscribers are interested in

- Brokers must match event topic or attributes to subscriptions as part of routing events



Why Use Pub/Sub?

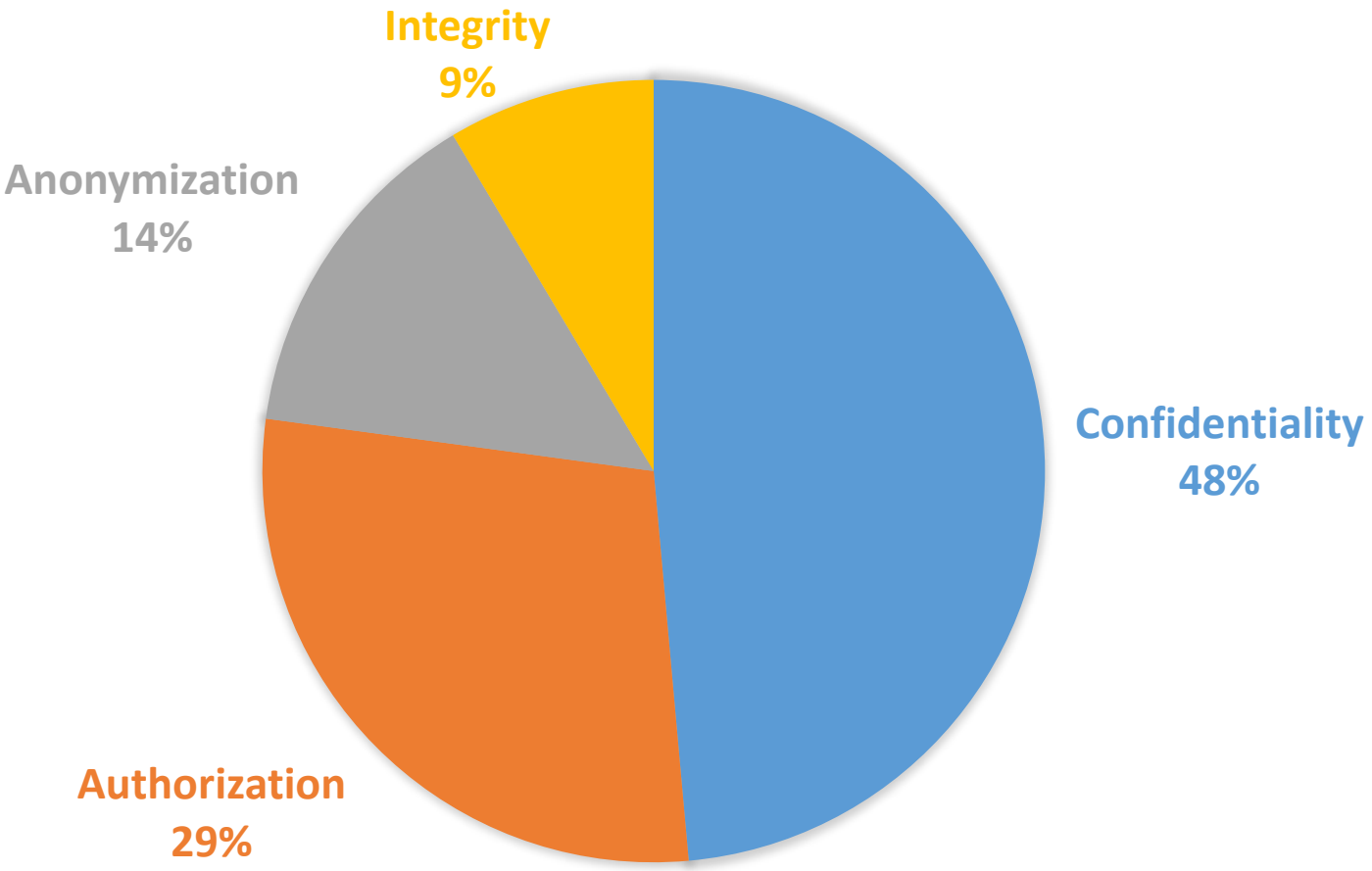
1. **Space Decoupling**
   - **Clients do not need to know of one another**
2. Time Decoupling
   - Sending an event is decoupled from receiving an event
3. Synchronization Decoupling
   - Clients do not block when sending or receiving events

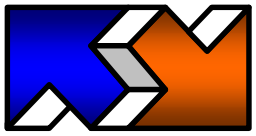


Research Categories

- Confidentiality
  - Hiding the content of events and subscriptions from others
- Authorization
  - Controlling which entities can do what actions within the system
- Anonymization
  - Hiding user identifying information from others
- Integrity
  - Attacks and mitigation strategies for pub/sub

Research Focus

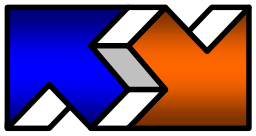




Confidentiality

- Hide sensitive event and subscription information from brokers and external entities
- Brokers still need to be able to filter events based on subscriptions
- Many existing techniques cannot be directly applied in content-based pub/sub
- Techniques:

  Symmetric-Key Encryption, Homomorphic Cryptosystems, ASPE, Functional Encryption, Multiple Layer Commutative Encryption, Oblivious Transfer, Secret Sharing

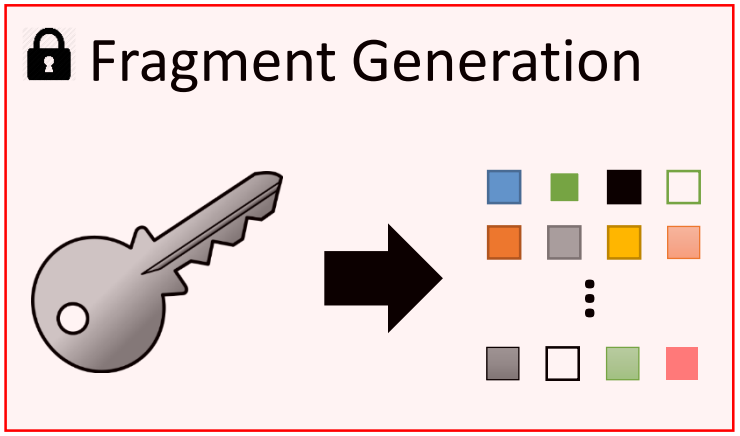


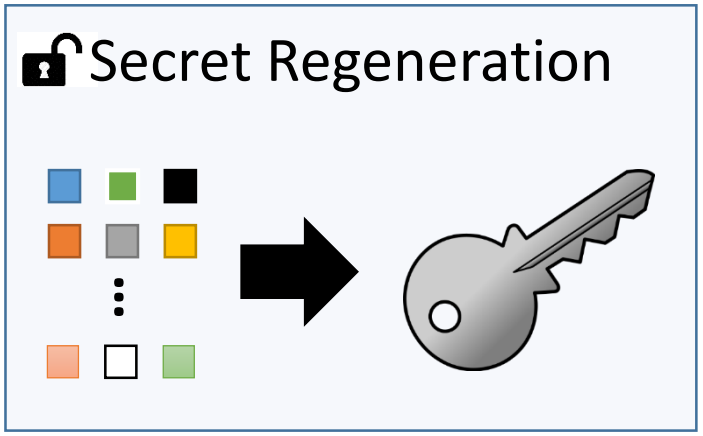
Confidentiality Limitations

- All solutions require the use of a service that is:
  1. Unilaterally trusted
  2. Universally available
  3. Out-of-band
- All solutions use the honest-but-curious broker threat model
- Secret sharing is the only exception but it requires a very large number of messages in order to share a secret

(k, n)-Secret Sharing Scheme

- Based on linear interpolation
- Introduces two operations:

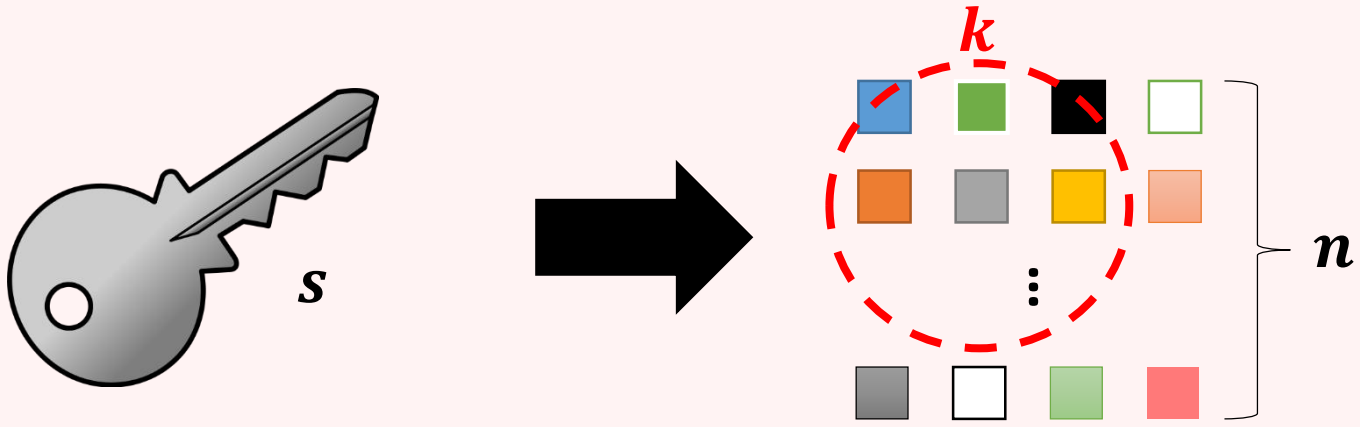




How to Share a Secret, Adi Shamir 1979

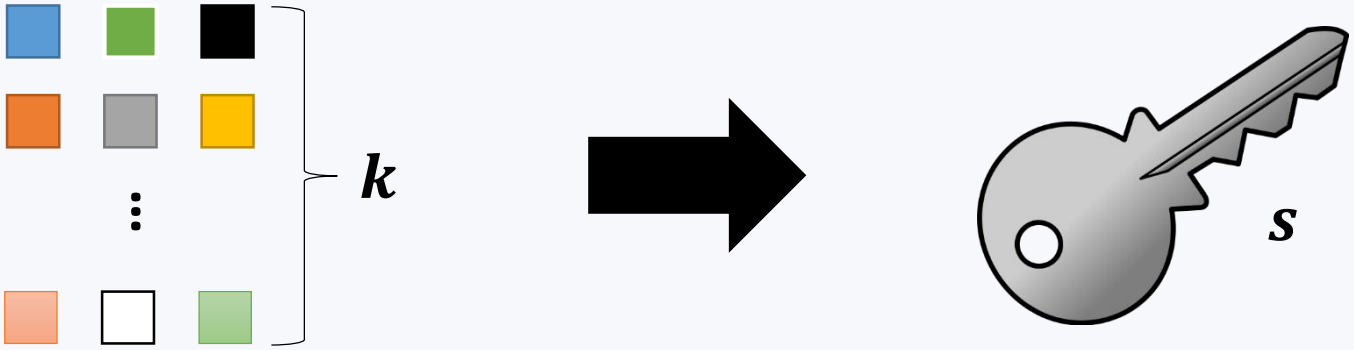
Fragment Generation

- Input: Secret s, input parameters k, n
- Output: n fragments
- At least k fragments necessary to regenerate s
- $n \geq k > 1$



Secret Regeneration

- Input: k fragments
- Output: Secret s
- Reverse operation to Fragment Generation



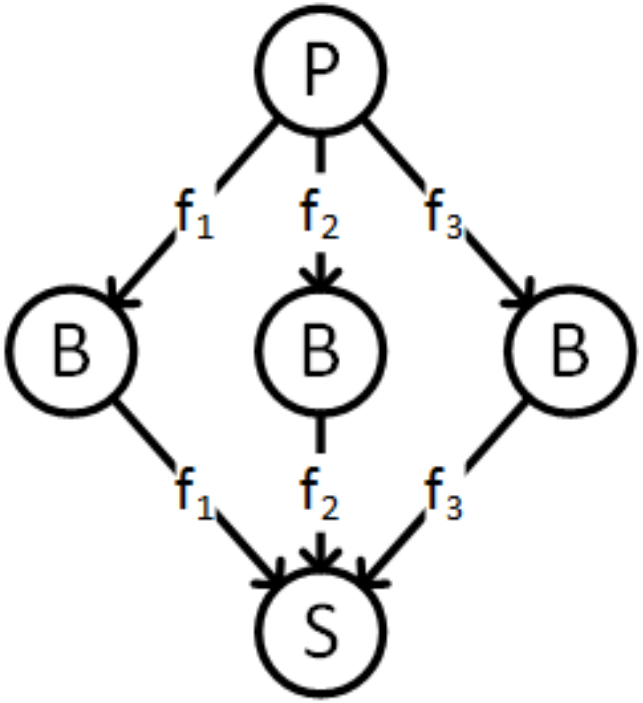
Secret Sharing Propagation Scheme (SSPS)

- Anyone with k fragments can regenerate secret s
- Secret is secure as long as no one ever ***sees*** more than $k - 1$ fragments
  - ➢No broker ever gets $k - 1$ fragments for a given secret
- In pub/sub, physically redundant delivery paths are used to disseminate fragments
  - Referred to as parallel paths
  - No single broker belongs to k parallel paths

    ⇔ no single broker can regenerate secret *s*

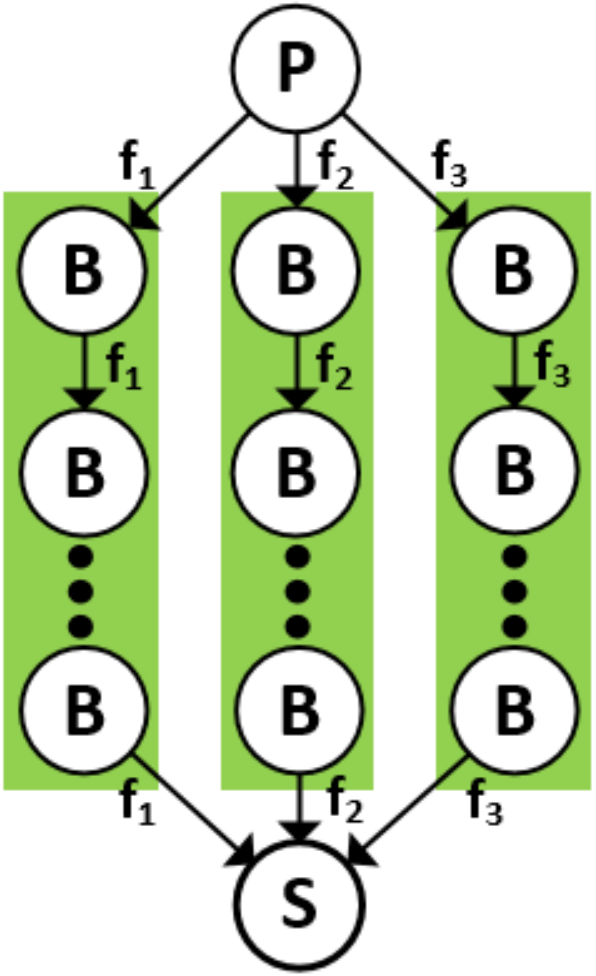
SSPS Example 1

- $n = 3$
- $k = 2$ or 3
- Up to $k - 1$ brokers may collude



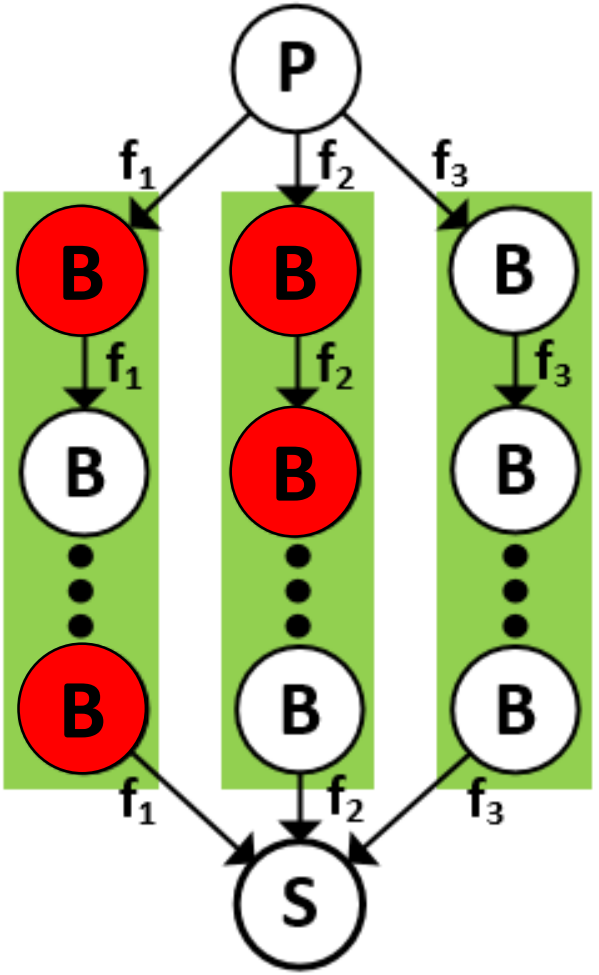
SSPS Example 2 – Multiple Hops

- Multiple hops between source and sink
- Parallel paths are highlighted
- Parallel paths are of equal length for simplicity



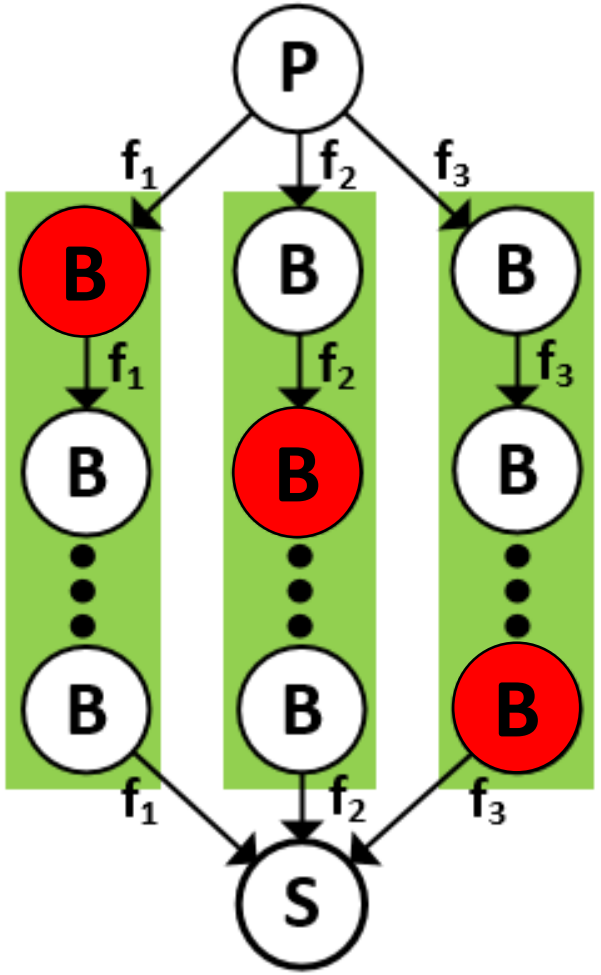
SSPS Example 2 – Collusion Tolerance

- Up to $k - 1$ parallel paths may contain colluding brokers



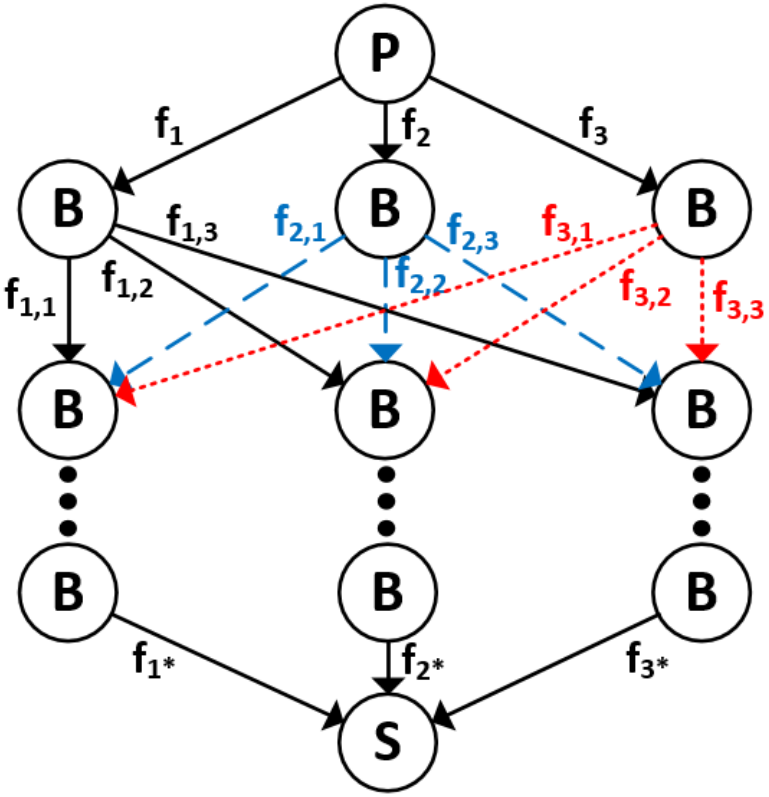
SSPS Example 2 – Compromised Topology

- Collusion between single broker in each path defeats scheme



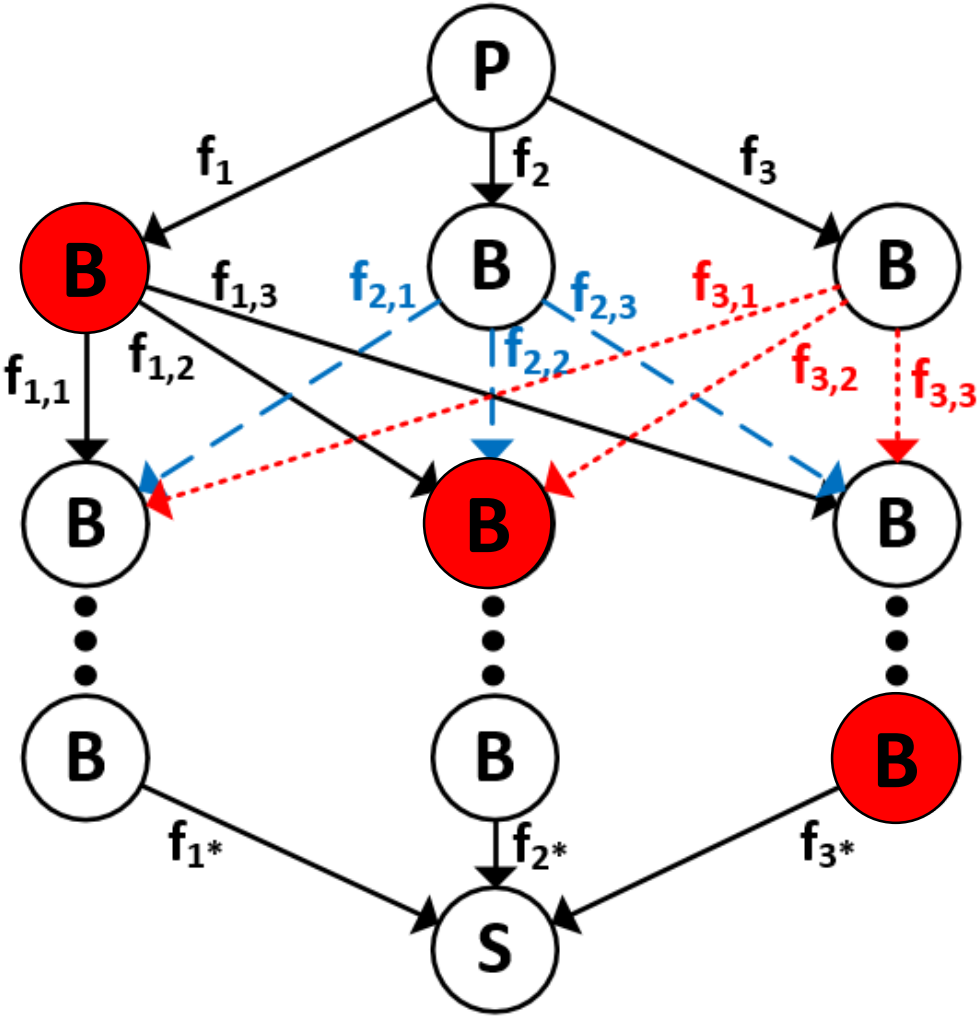
Iterative SSPS (ISSPS)

- Brokers apply (k, n)-Secret Sharing Scheme on each received fragment



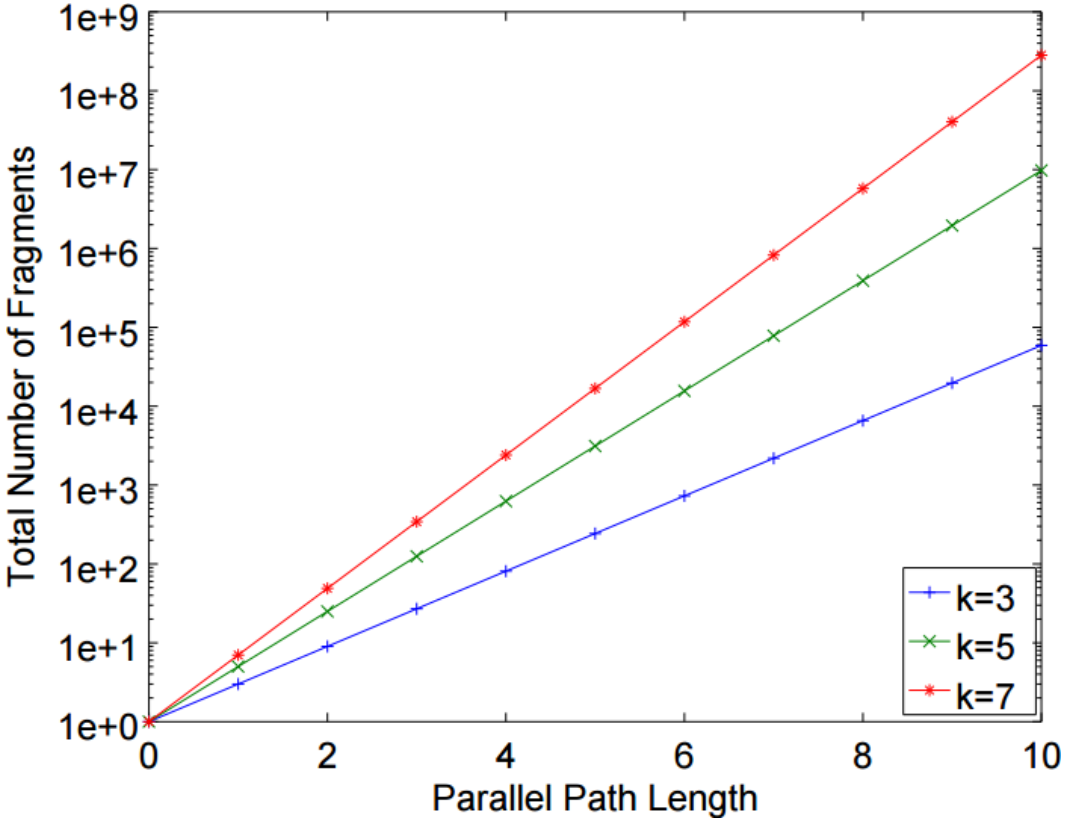
Iterative SSPS Advantage

- All paths collude without defeating scheme
- Proof published in our DEBS 2018 paper



ISSPS Disadvantage

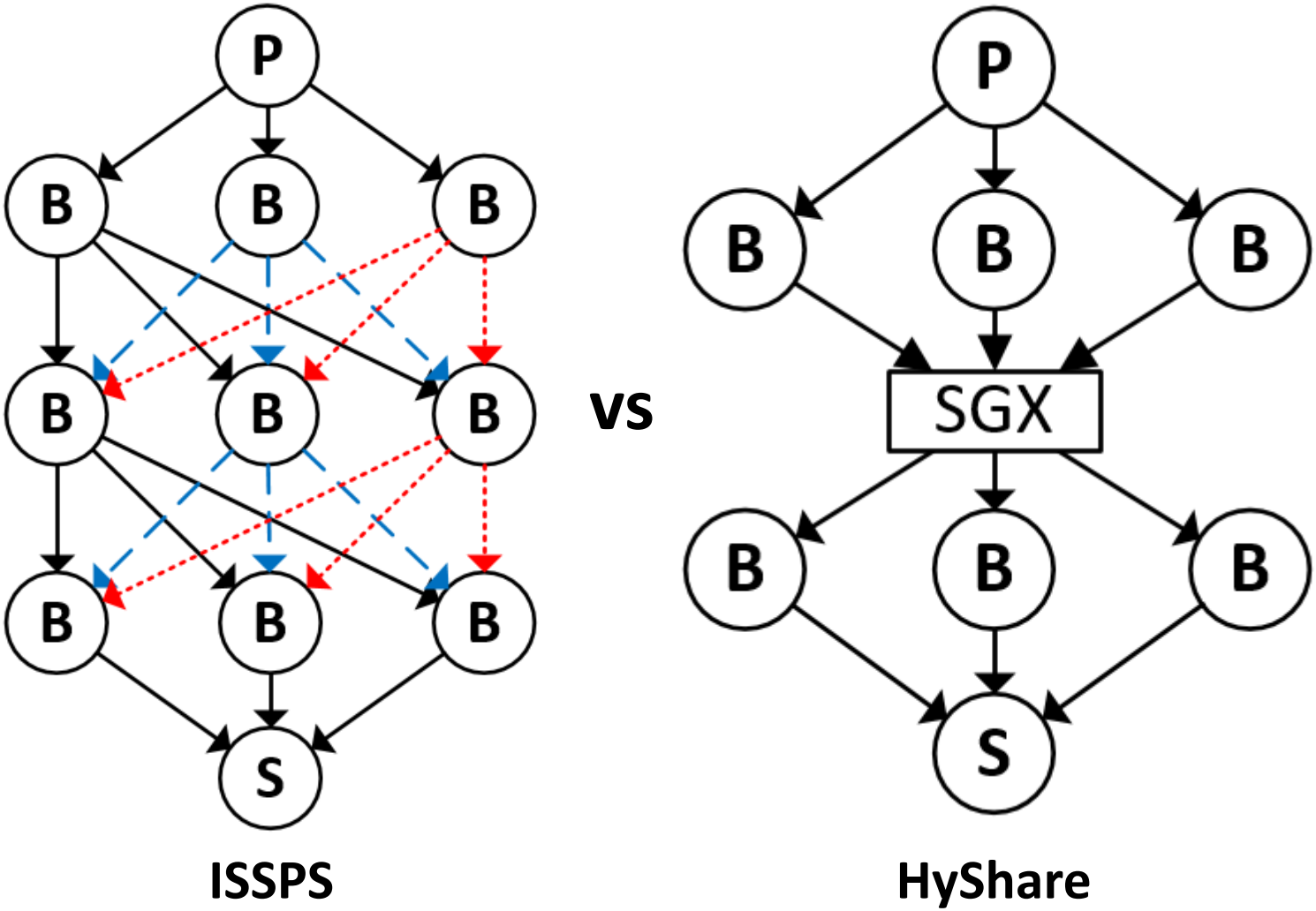
- Exponential growth in number of messages needed as path length between publisher and subscribers increases



HyShare

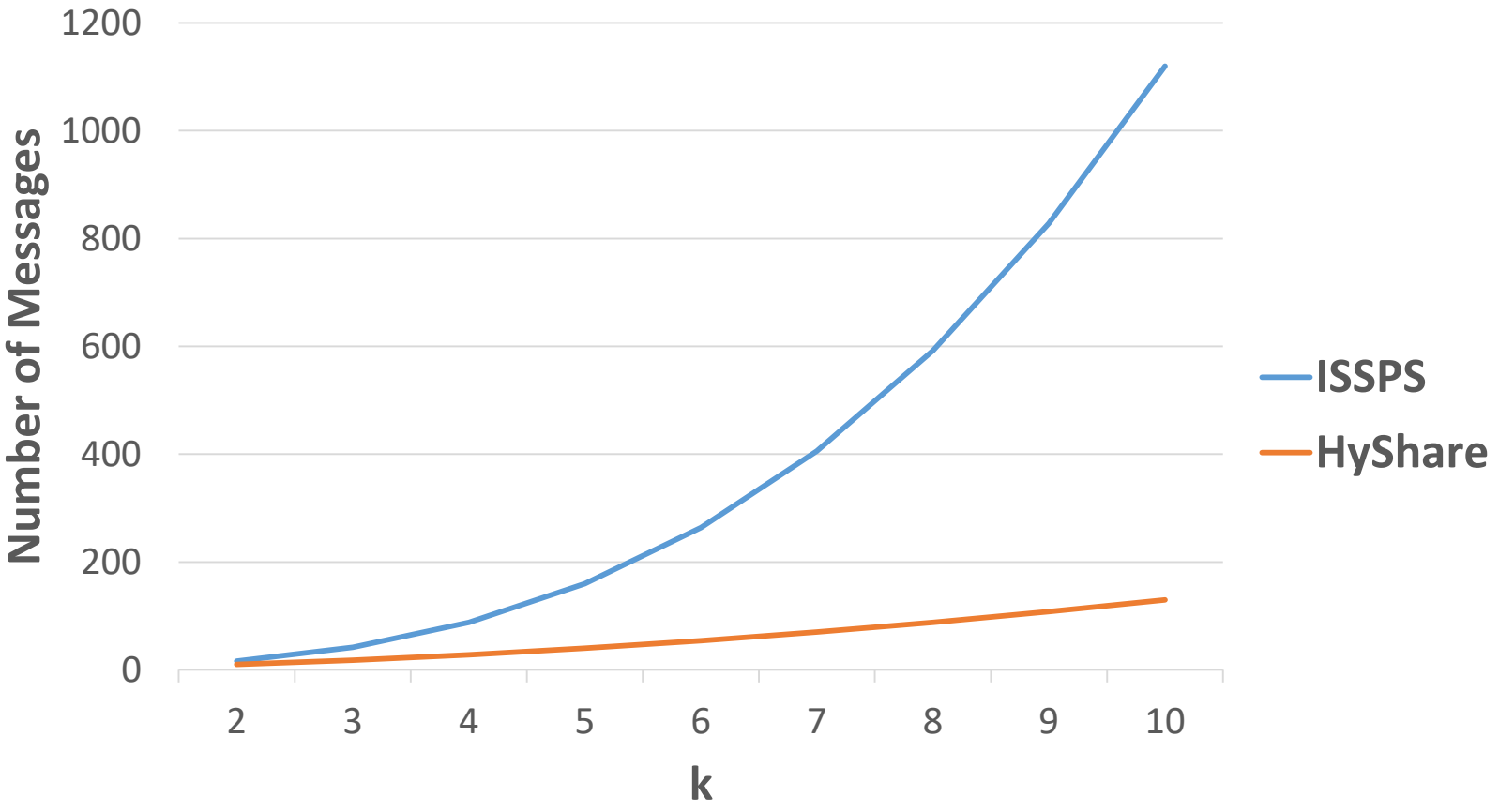
- Use hybrid broker networks to share secret
- Leverages recent developments in Intel's SGX
- Hybrid broker network composed of SGX-enabled brokers and ISSPS-brokers
- ISSPS-brokers run ISSPS
- SGX-enabled brokers regenerate the original secret then run ISSPS
- More SGX-enabled brokers means:
  - Fewer messages required to share a secret
  - Less hardware diversity
  - Increased cost

HyShare Simple Example

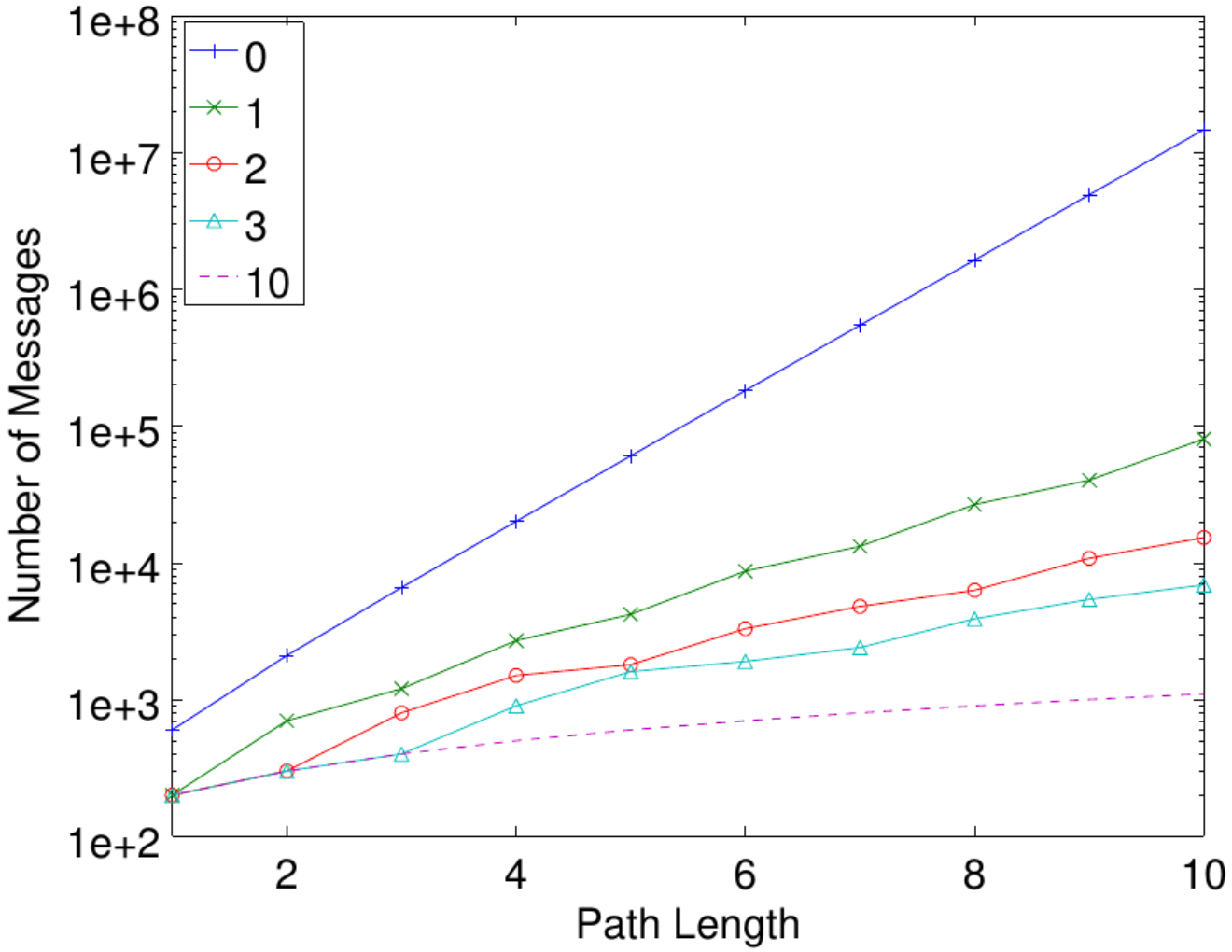


ISSPS **HyShare**

Number of Messages Needed to Share a Secret

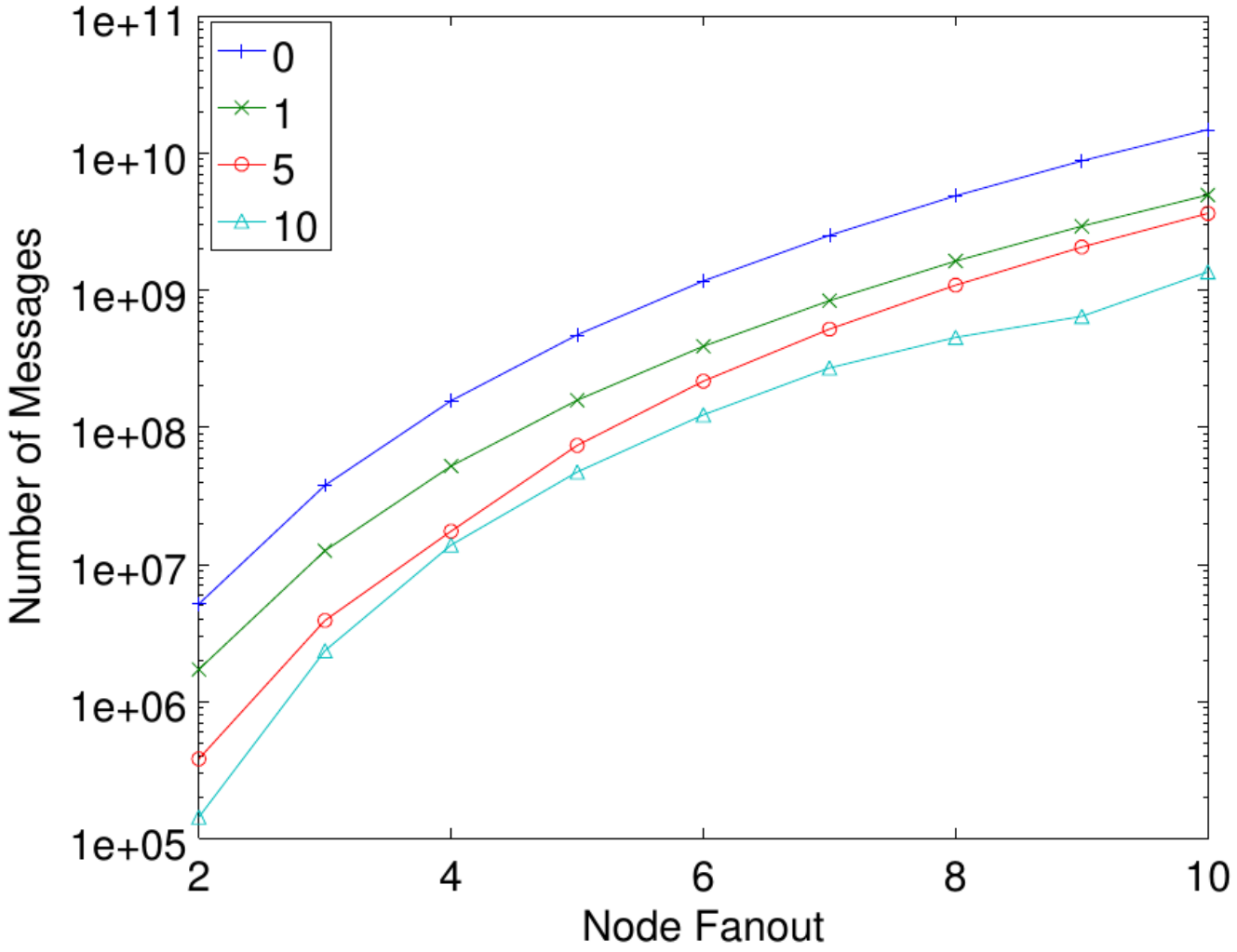


Number of Messages Needed to Share a Secret



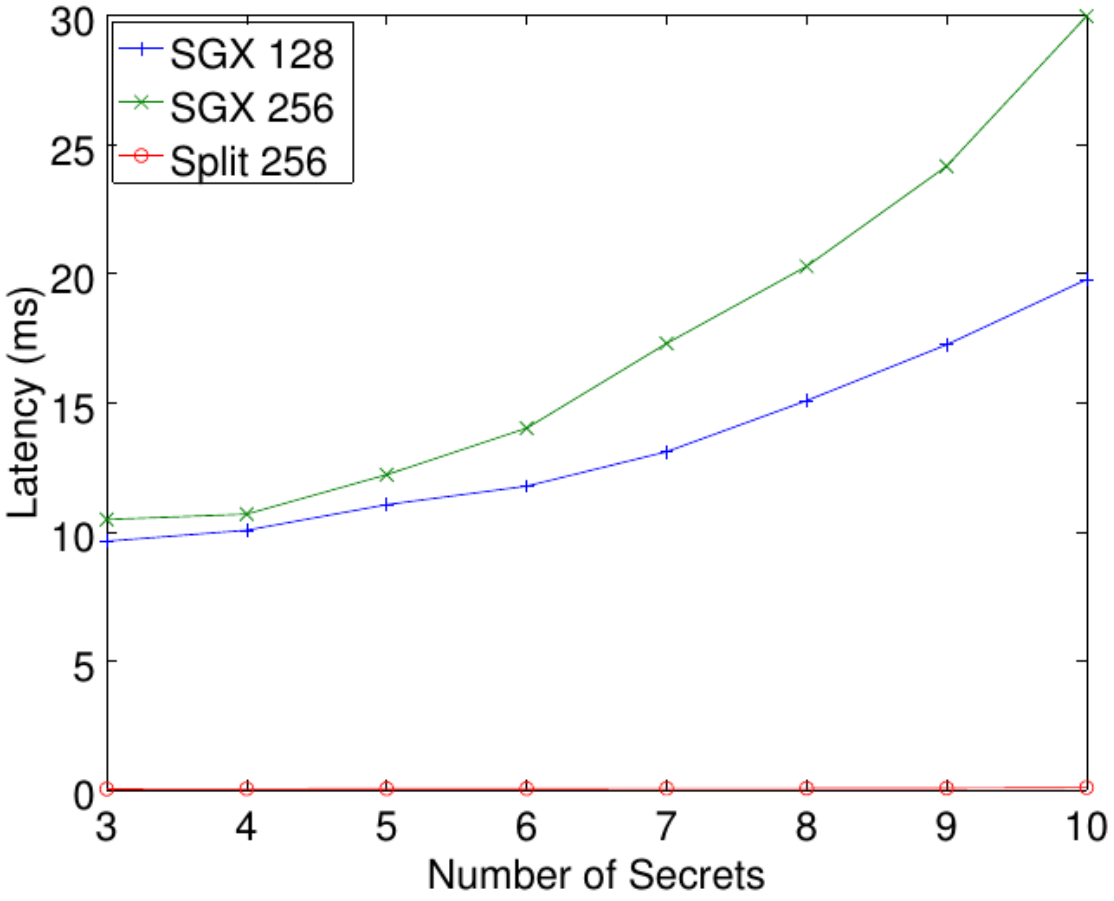
$k = 3$

Stock Quote Dissemination



$k = 3$

Secret Regeneration Overhead

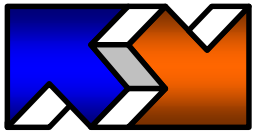


Conclusions

- Categorized the existing pub/sub research
- Identified limitations of existing approaches
- Introduced HyShare, a novel secret sharing scheme that leverages SGX in pub/sub for the first time
- Trusted service used only when bringing up pub/sub network
- No other trusted or out-of-band resources required
- Broker collusion tolerance

Future Work

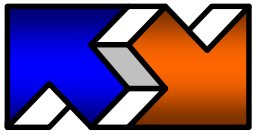
- Identify what security features are provided by existing pub/sub systems
- Evaluate effects of smarter placement strategies of SGX-enabled brokers
- Examine the effects of leveraging an entire broker network that can use SGX



Questions?

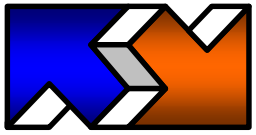
Symmetric-Key Encryption

Homomorphic Cryptosystems



ASPE

Functional Encryption

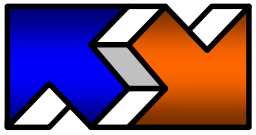


Multiple Layer Commutative Encryption

Oblivious Transfer

Authorization

- The control of actions being performed within the pub/sub system
- Client Access Control
  - Arbitrary control over messages send or received by client
  - Technique: policy (pre and post matching)
- Broker Access Control
  - Control over messages transmitted within broker network
  - Techniques: Hop-Level Access Control, Domain-Based Access Control

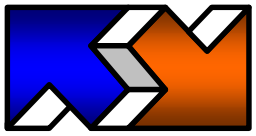


Policy

- Pre- and post-matching of policy rules

Hop-Level Access Control

Domain-Based Access Control



Anonymity

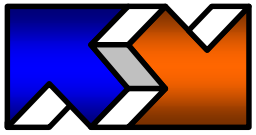
- The hiding of identities and identifying information
- Communication Anonymity
  - The hiding of identities during communication of messages
  - Techniques: Onion Routing, Logical Layer Scheme
- Data Anonymity
  - The sanitization of data being sent of identifying information
  - Techniques: k-anonymity, ℓ-diversity

Onion Routing

Logical Layer Scheme

k-Anonymity

ℓ-Diversity



Integrity

- Attacks and mitigation strategies specifically for pub/sub
- Attacks: Denial of Service, Overlay Scan Attack, Bogus Broker Attack

Denial of Service

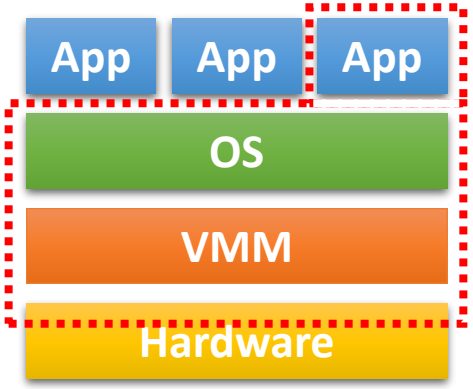
Overlay Scan Attack

Bogus Broker Attack

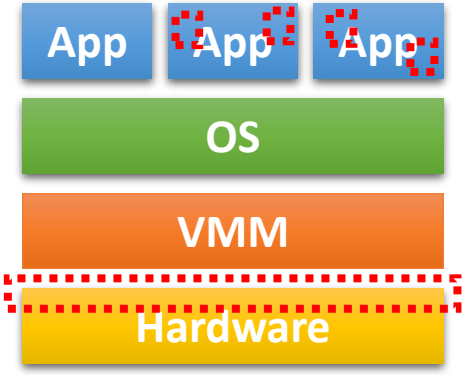
Trusted Execution Environments

- Intel SGX, AMD Platform Security Processor, ARM TrustZone
- Protected area of hardware
- Guarantees data confidentiality and integrity

Normal Attack Surface:



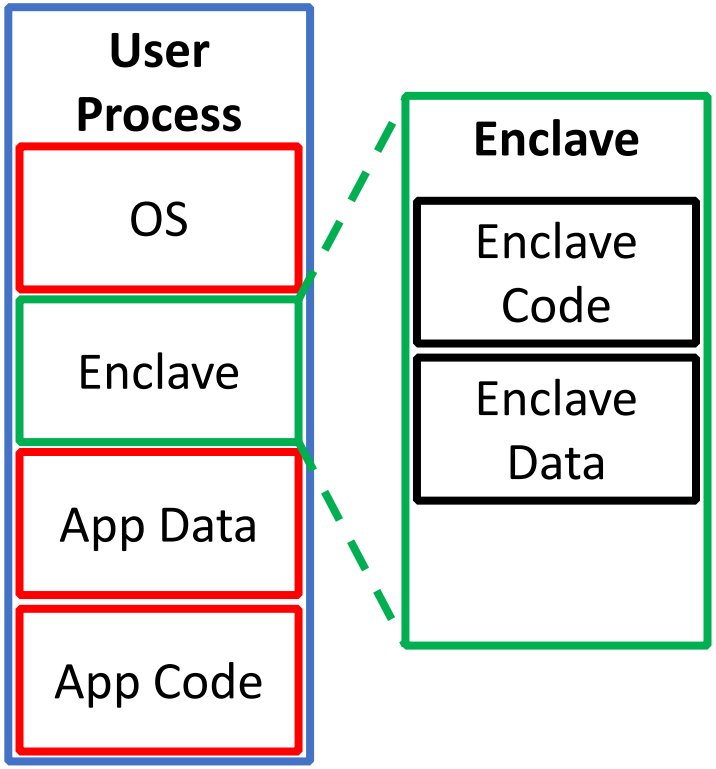
TEE Attack Surface:



Attack Surface

Intel Software Guard Extension (SGX)

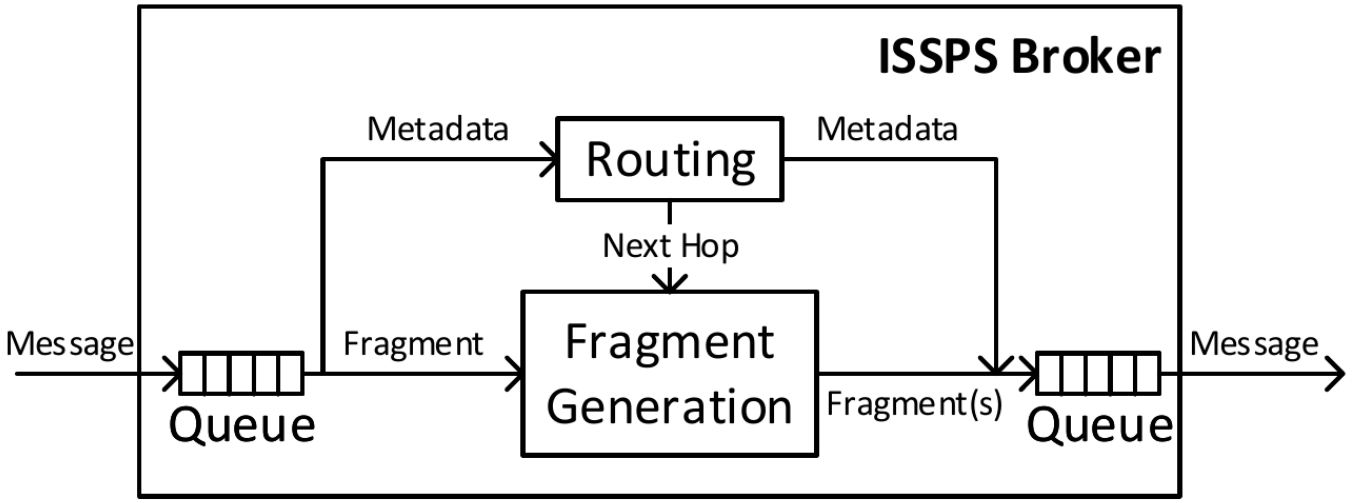
- SGX provides protected containers which:



- Execute code securely
- Provide confidentiality
- Provide integrity
- Have full access to app. data
- Can run attestation protocol

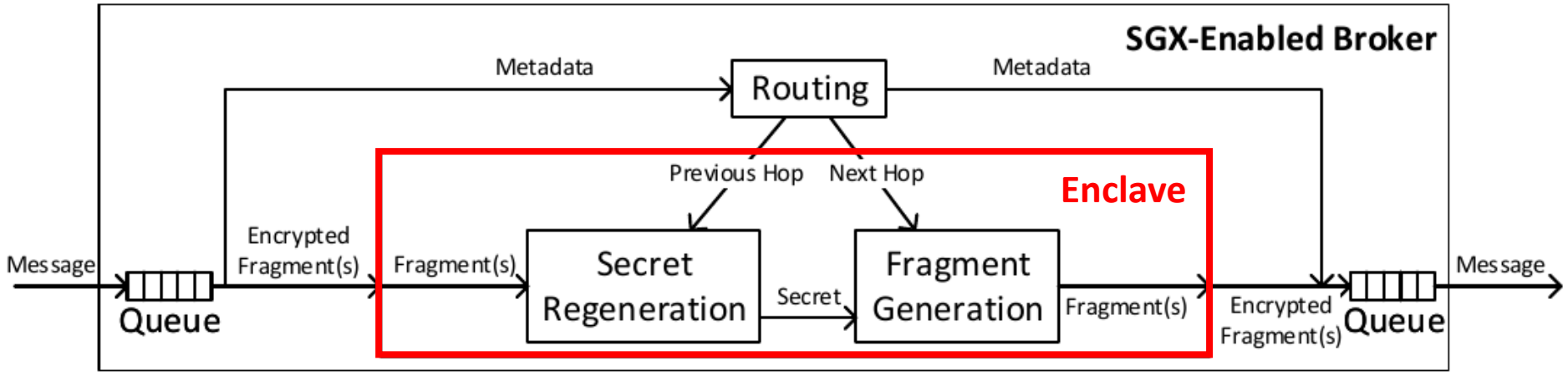
ISSPS-Brokers

- Operate similar to ISSPS
- Generates new fragments if next hop is another ISSPS-broker
- Otherwise, forwards fragments



SGX-Enabled Brokers

- Receives all fragments within enclave
- Regenerates original secret
- Generates n fragments if the next hop is an ISSPS-broker
- Forwards encrypted secret otherwise



Why Share Secrets?

- Keep sensitive information secret from the broker network
  - Personal information, healthcare records, etc.
  - Leak investment strategies
- It's the law
  - Healthcare records must be encrypted before sending
  - General Data Protection Regulation (EU)
- Required by the pub/sub confidentiality literature
  - Homomorphic Cryptosystems, ASPE, Functional Encryption, Multiple Layer Commutative Encryption

Research Problem

- Disseminate a secret/key from a publisher to a subset of subscribers
- Maintain the space decoupling property
- Brokers may share or leak information to learn secret (i.e., collude)
- Minimize trust assumptions required
- Minimize number of messages required

Assumptions of Confidentiality Solutions

- Require the sharing of a key, secret or security parameter(s)
- Rely on a dissemination service that is:
  - Out-of-band
  - Unilaterally trusted
  - Universally available
- Must use dissemination service whenever a subscription event occurs
- Weaken decoupling property between clients

Pure SGX-Enabled Broker Topologies

- Advantages:
  - Conceptually simple to have one type of broker
  - Number of messages needed to share a secret scales linearly with path length
- Disadvantages:
  - Cost of processors that support SGX
  - Lack of SGX adoption in cloud service providers
  - Severe limits on hardware diversity

Parallel Path Generation