UNIVERSITÉ DE NEUCHÂTEL

Efficient confidentiality preservation for cloud-supported content-based publish/subscribe

Etienne Rivière

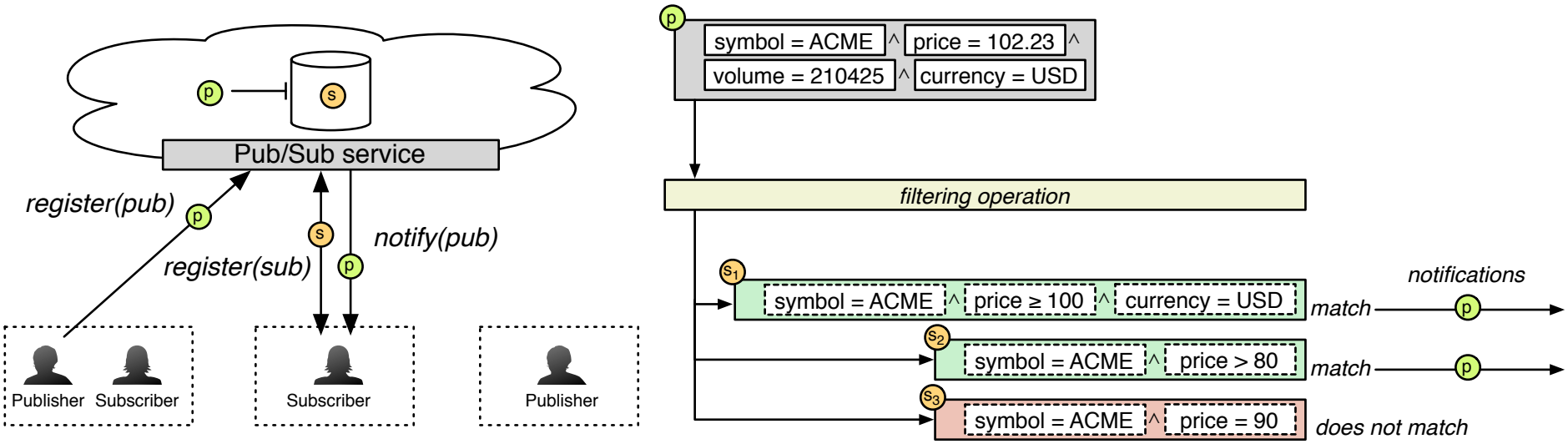
University of Neuchâtel, Switzerland

joint work with Raphaël Barazzutti, Pascal Felber, Hugues Mercier and Emanuel Onica

SEC2 - July 2016

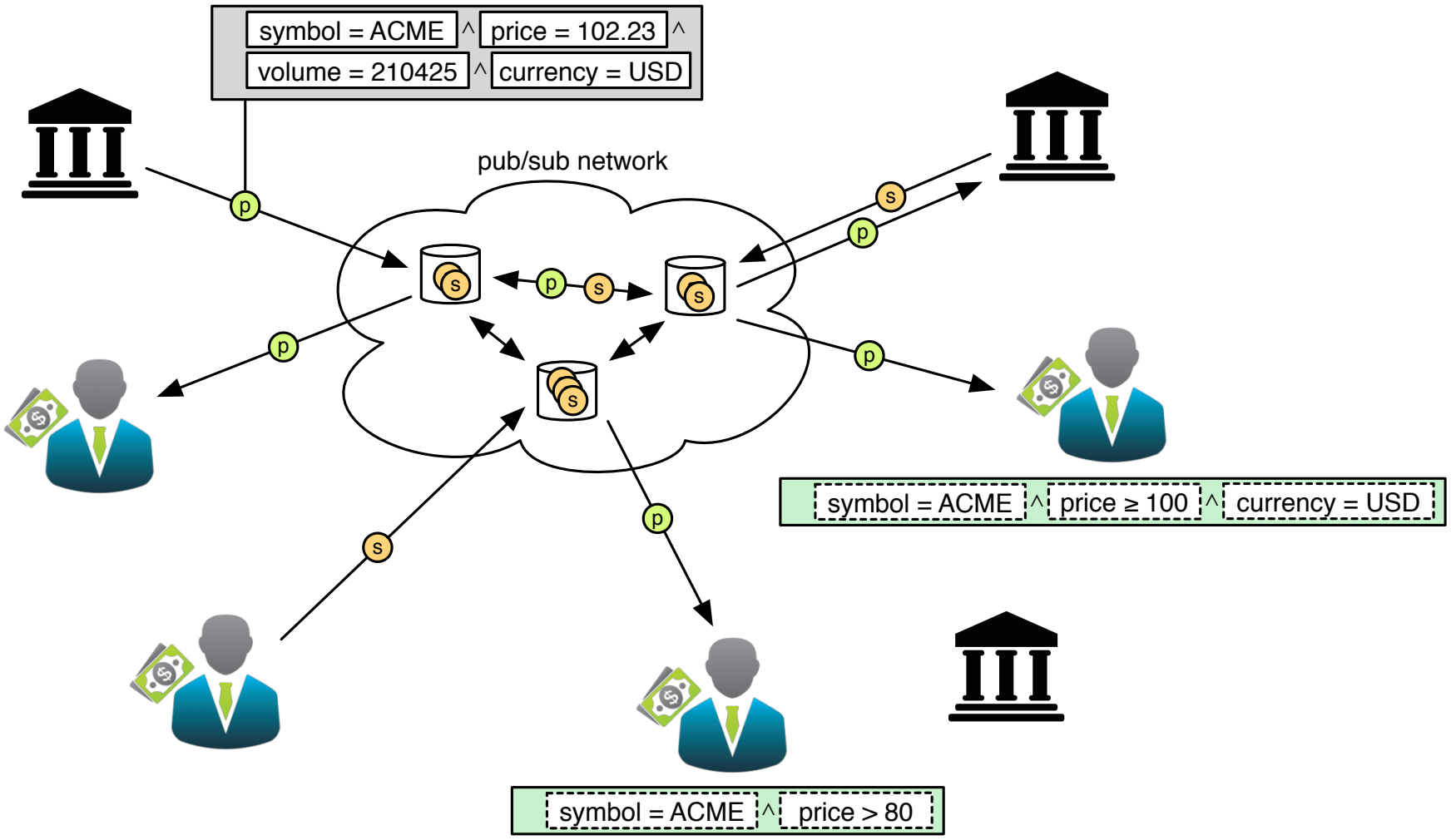
etienne.riviere@unine.ch

Content-based pub/sub

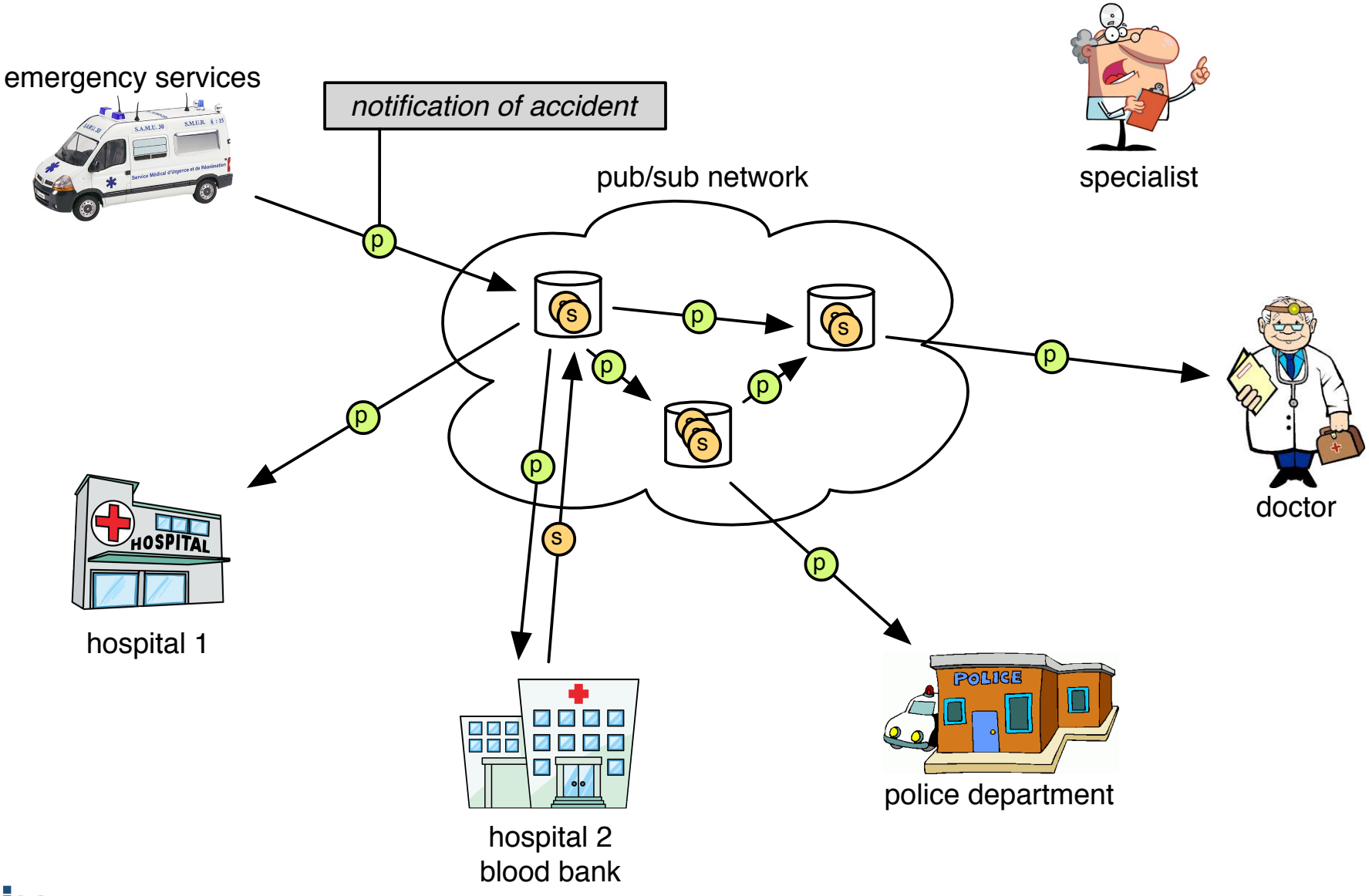


- Decoupled communication: *subscriptions* on the content of *publications* (typically represented by a set of attributes)
- Provided by a network of brokers [SIENA, PADRES, …] or a stream-processing engine [StreamHub]
- Filtering operation at the core of the pub/sub service

Example 1: stock quotes dissemination



Example 2: federated e-Health system



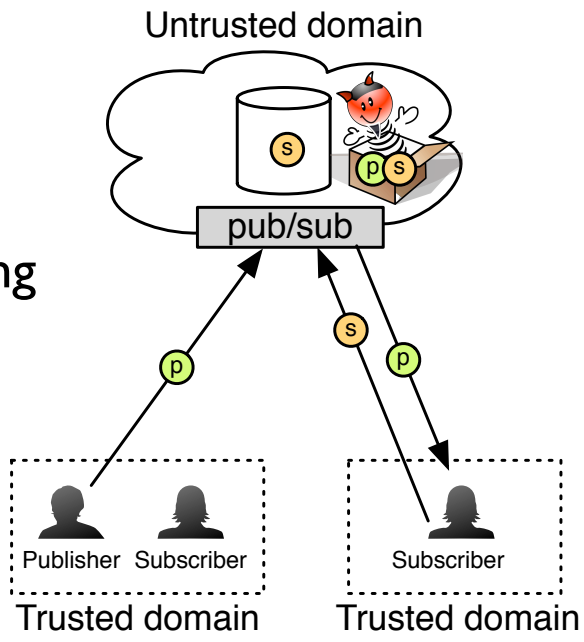
Pub/sub services on public clouds

- Content-based pub/sub ideal for interconnecting applications in different administrative domains
  - No coupling: easy to add and remove information producers/consumers
  - Expressive communication model

- Public clouds are ideal for hosting pub/sub services
  - Cheap and *pay-as-you-go*
  - Always on and geographically distributed
  - Easily reachable from all communicating parties

Beware of eavesdroppers!

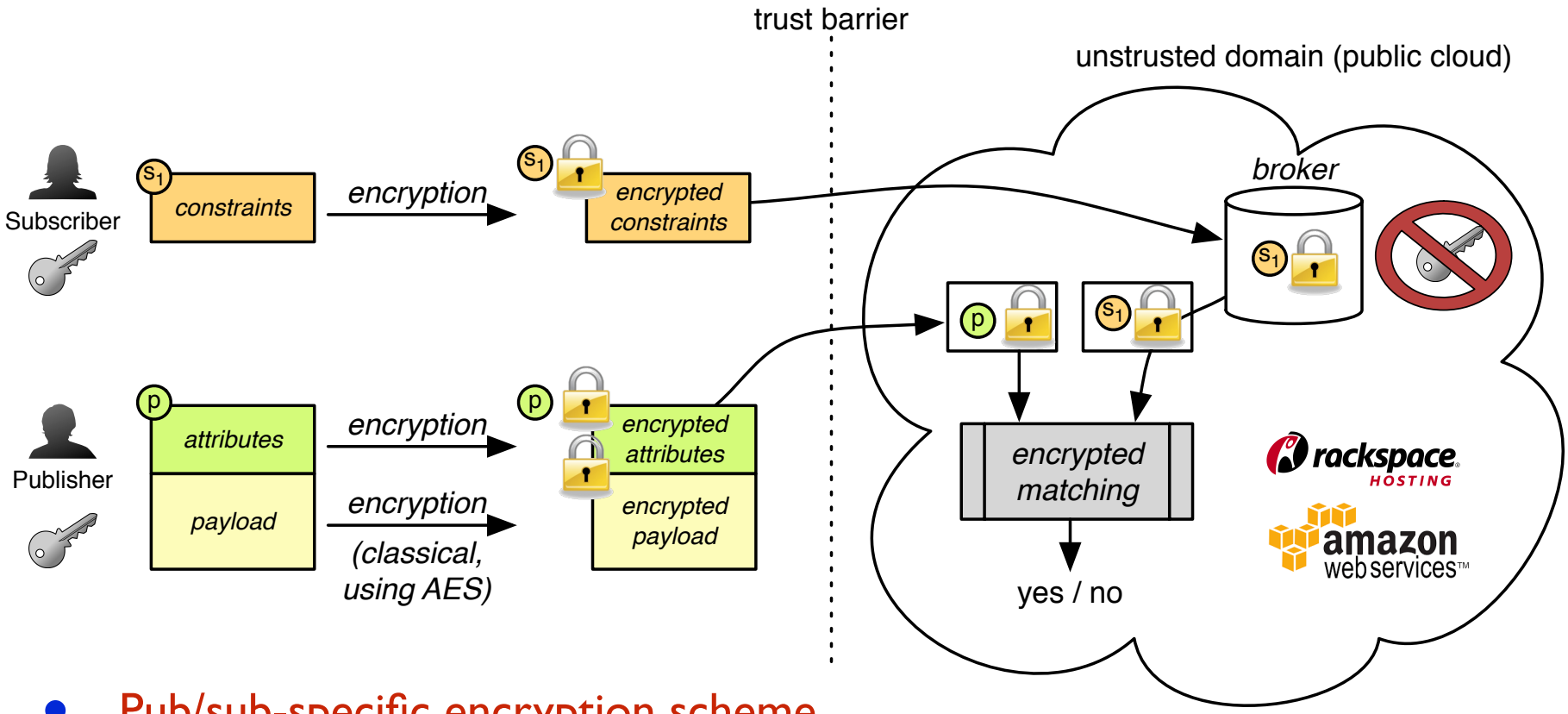
- Public clouds should be untrusted by default
  - Colocation attacks, malicious operators
- Leaking plaintext subscriptions constraints
  - stock quotes: know one's interest in some stock
  - e-health: know that a specialist doctor is following one particular patient
- Leaking plaintext publications attributes
  - e-health: know private information about the treatments received by some patient
- This talk: efficient support for *confidentiality-preserving* pub/sub
  - We will only consider the *honest-but-curious* model



Confidentiality through access control

- Prevent access to pub/sub data by using access control and classical encryption (AES, etc.)
  - Only allow trusted domains to access sensitive/routable fields
  - Examples: RBAC-based [Bacon et al., DEBS 2008], proxy re-encryption [Khurana, SAC 2005], policy-based [Wun & Jacobsen, Middleware 2007], EventGuard [Srivatsa et al., CCS 2008]
- Limitation: classical encryption is opaque
  - Prevents any node/broker in an untrusted domain to perform filtering and routing using sensitive fields
  - Forces to flood and post-filter at the end domains
  - Limits interest of hosting pub/sub in public clouds
- Alternative: leverage encrypted processing

Encrypted matching



- Pub/sub-specific encryption scheme
  - A form of encrypted computation: determining if an encrypted publication's attributes match encrypted subscription constraints
  - Encryption key shared only between subscribers and publishers
  - Most schemes derived from encrypted databases solutions (specialized schemes = less costly than homomorphic encryption)

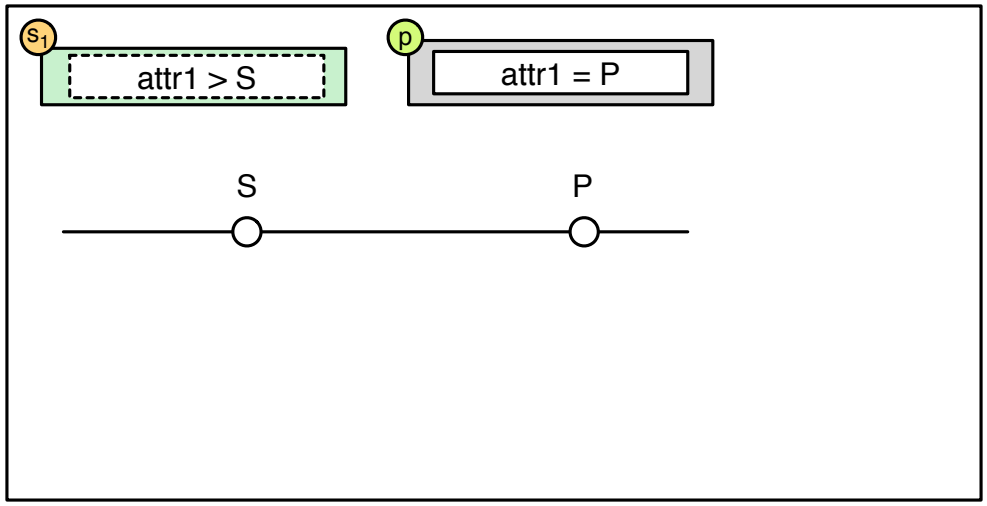
ASPE

(Asymmetric Scalar Product Preserving Encryption)

- Symmetric encryption scheme
  - Invented for encrypted kNN queries [Wong et al., SIGMOD 2009]
    - search for *k* data points closest to a target in an encrypted database
  - Adapted for pub/sub in [Choi et al., DEXA 2010]
  - Extended in [Onica et al., Middleware 2015]
- Set of attributes fixed
  - Publications = values
  - Subscriptions = set of constraints (< , > , =)
- Matching problem
  - Evaluating distance relations between points representing subscriptions constraints and publications values
- Let's see a simple example in one dimension …

ASPE

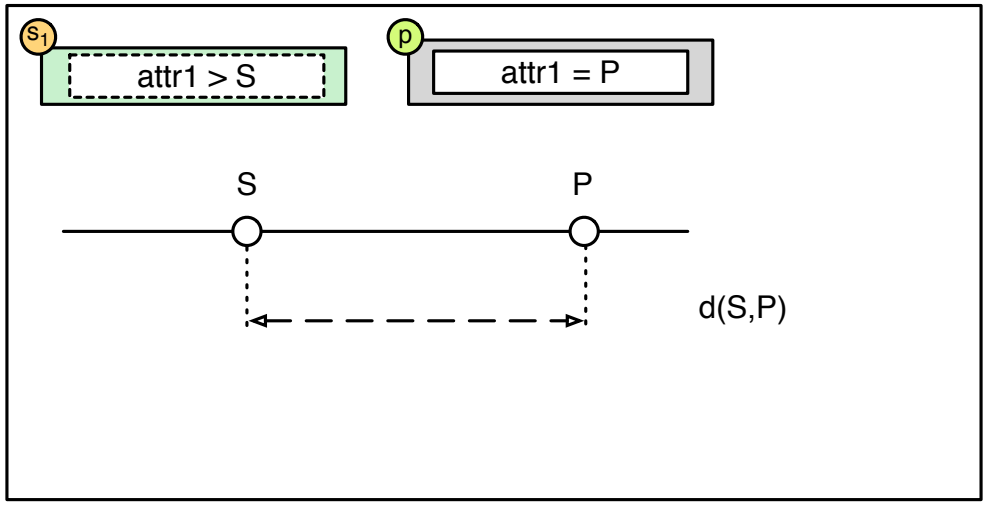
(Asymmetric Scalar Product Preserving Encryption)



- The scheme should not allow determining the distances …
  - Distance-Recoverable Encryption (DRE) is vulnerable to Known-Plaintext Attacks

ASPE

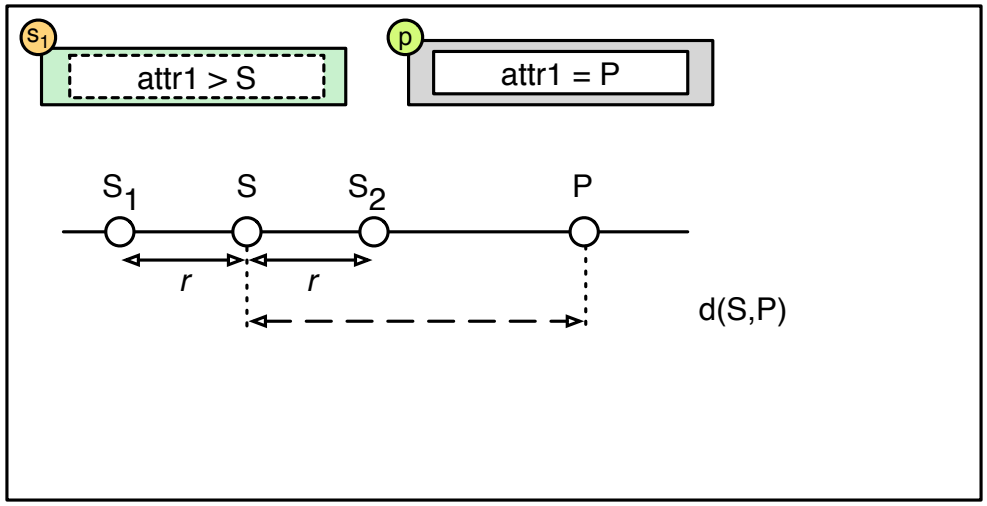
(Asymmetric Scalar Product Preserving Encryption)



- The scheme should not allow determining the distances …
  - Distance-Recoverable Encryption (DRE) is vulnerable to Known-Plaintext Attacks

ASPE

(Asymmetric Scalar Product Preserving Encryption)



- The scheme should not allow determining the distances …
  - Distance-Recoverable Encryption (DRE) is vulnerable to Known-Plaintext Attacks
- … but it should allow comparing distances

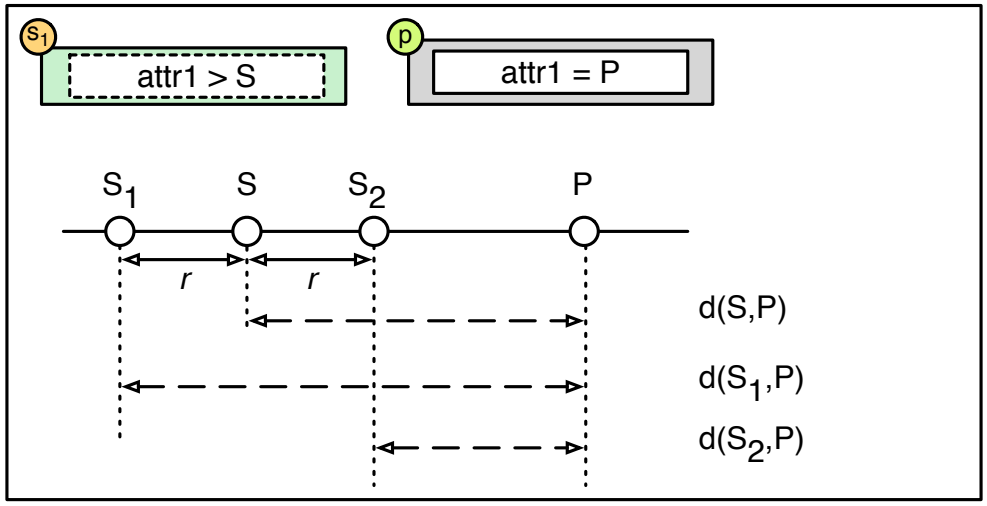
$$d(S_1,P)-d(S_2,P) > 0 \Leftrightarrow$$
$$d(S_1,P) > d(S_2,P) \Leftrightarrow S < P$$

This expression (same sign) can be expressed as a sum of scalar products:

$$D'_i = d(S_{i1},P)^2-d(S_{i2},P)^2$$
$$=\| S_{i1} \|^2 - \| S_{i2} \|^2 +2(S_{i2}-S_{i1})P$$

ASPE

(Asymmetric Scalar Product Preserving Encryption)



- The scheme should not allow determining the distances …
  - Distance-Recoverable Encryption (DRE) is vulnerable to Known-Plaintext Attacks
- … but it should allow comparing distances

$$d(S_1,P)-d(S_2,P) > 0 \Leftrightarrow$$
$$d(S_1,P) > d(S_2,P) \Leftrightarrow S < P$$

This expression (same sign) can be expressed as a sum of scalar products:

$$D'_i = d(S_{i1},P)^2-d(S_{i2},P)^2$$
$$=\| S_{i1} \|^2 - \| S_{i2} \|^2 +2(S_{i2}-S_{i1})P$$

ASPE

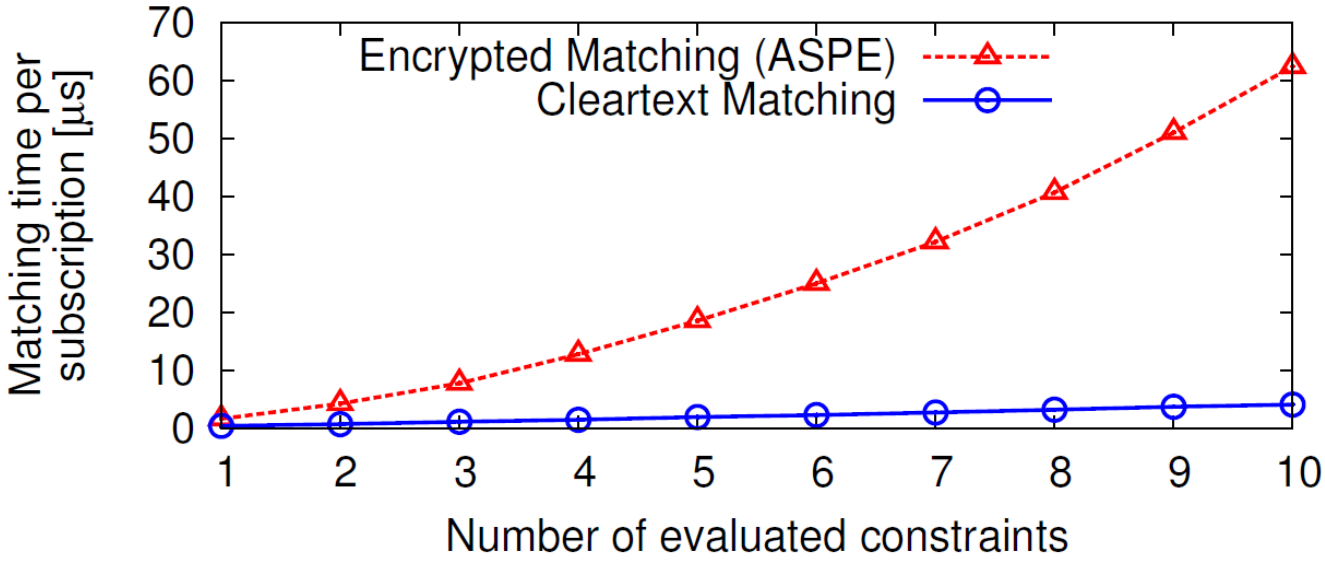
(Asymmetric Scalar Product Preserving Encryption)

- For multi-dimensional subscriptions:
  - Embed each criteria in a vector: $S_i = (q_1, q_2, \ldots, q_{i-1}, v_i, q_{i+1}, \ldots, q_d)$
  - Values for $q_{c \neq i}$ are selected randomly
  - Generate, for a random value r_i: $S_{i1} = (q_1, q_2, \ldots, v_i - r_i, \ldots, q_d)$
    $S_{i2} = (q_1, q_2, \ldots, v_i + r_i, \ldots, q_d)$
  - Additional dimensions can be added for increased resilience to BFA

- Principle: encrypt S_{i1}, S_{i2} and P such that the scalar product can be computed
- Key: invertible (non orthogonal) matrix M
  - Subscriber uses transpose M^T, publisher uses inverse M^{-1}
- Encryption:
$$S'_{M^T i1} = M^T (S_{i1}, -0.5 \parallel S_{i1} \parallel)^T;$$
$$S'_{M^T i2} = M^T (S_{i2}, -0.5 \parallel S_{i2} \parallel)^T;$$
$$P'_M = M^{-1} q (P, 1)^T.$$
(q: random scaling factor)

- Evaluation:
$$D'_i \triangleq d(S_{i1}, P)^2 - d(S_{i2}, P)^2$$
$$(S'_{M^T i2} - S'_{M^T i1}) P'_M = 0.5 q D'_i$$

ASPE performance



- ASPE compared to plaintext filtering
  - using SIENA's optimized counting algorithm
  - average over 10.000 randomly generated subscriptions
- Clear performance gap between plaintext and encrypted matching
  - More complex: matrix based, need for high-precision number representations, use of additional dimensions for resilience, etc.

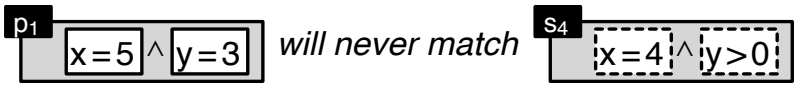
Prefiltering

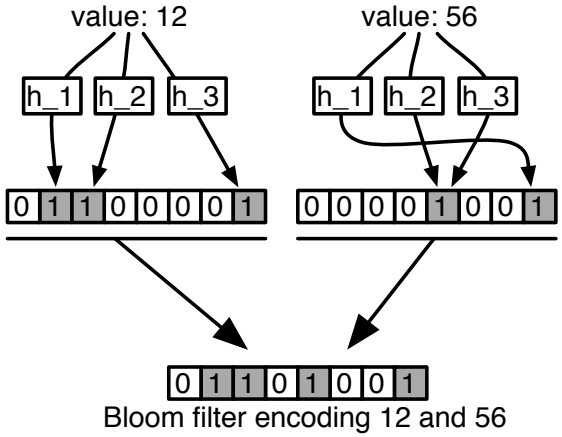
- How can we reduce the performance gap?
  - Prefiltering phase that discards subscriptions before they are sent to the costly encrypted matching operator

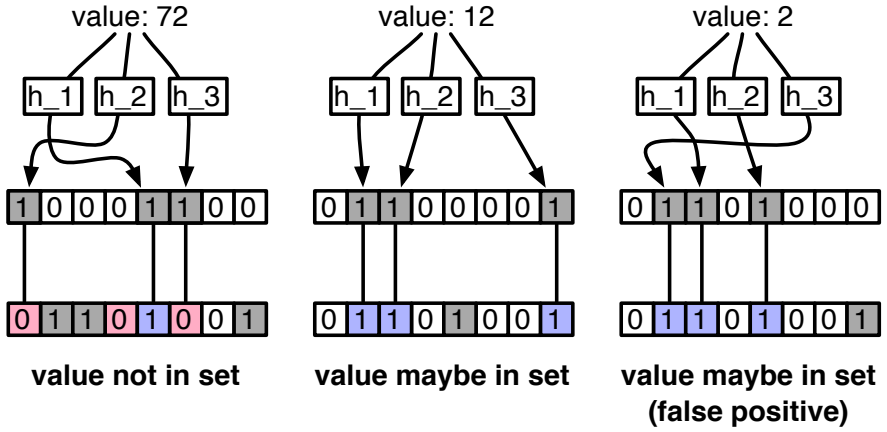
- Principle
  - Add additional information to subscriptions and publications
  - Discard subscriptions known *not to match* a publication
  - Use encrypted matching for others
  - Ability to use *containment* information if available

Prefiltering with Bloom filters

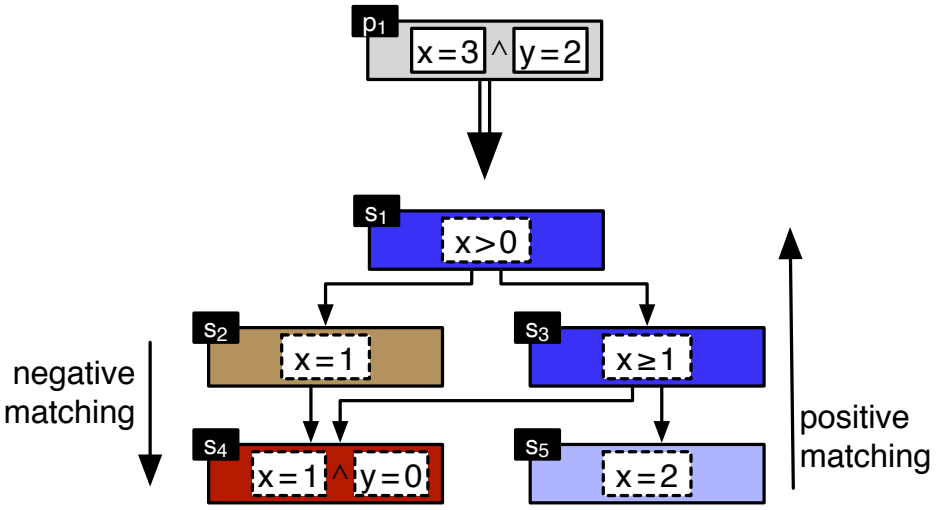
- Embed Bloom filters with *both* subscriptions and publications
  - Compact set representation
  - Allow deciding when an element does *not* belong to a set
  - Use *k* hash functions, return positions of *k* bits to set in bit field
- Subscriptions: values associated to equality (=) constraints
- Publications: values for all attributes
- If the value for an equality is know to be absent in attribute values, this can never be a match:

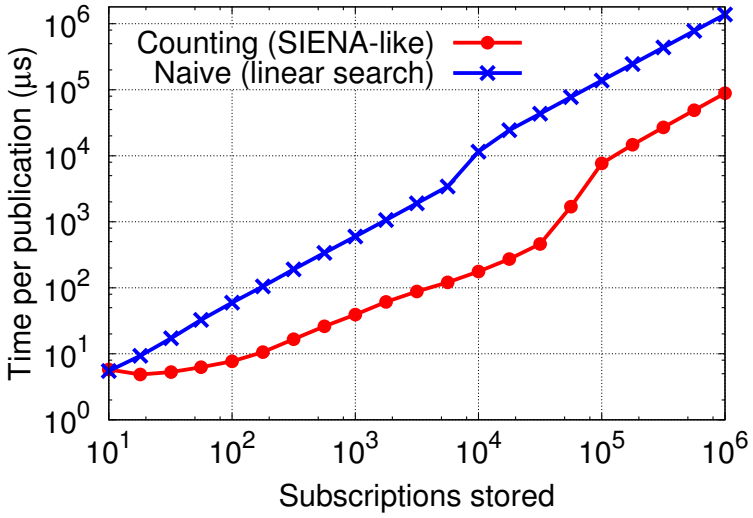






Subscription containment

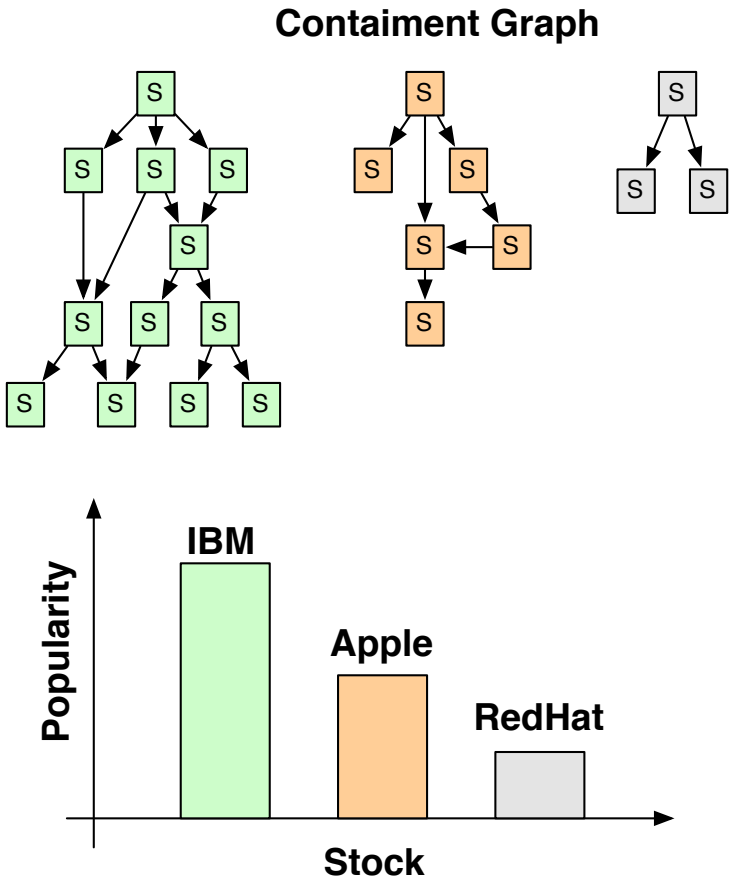




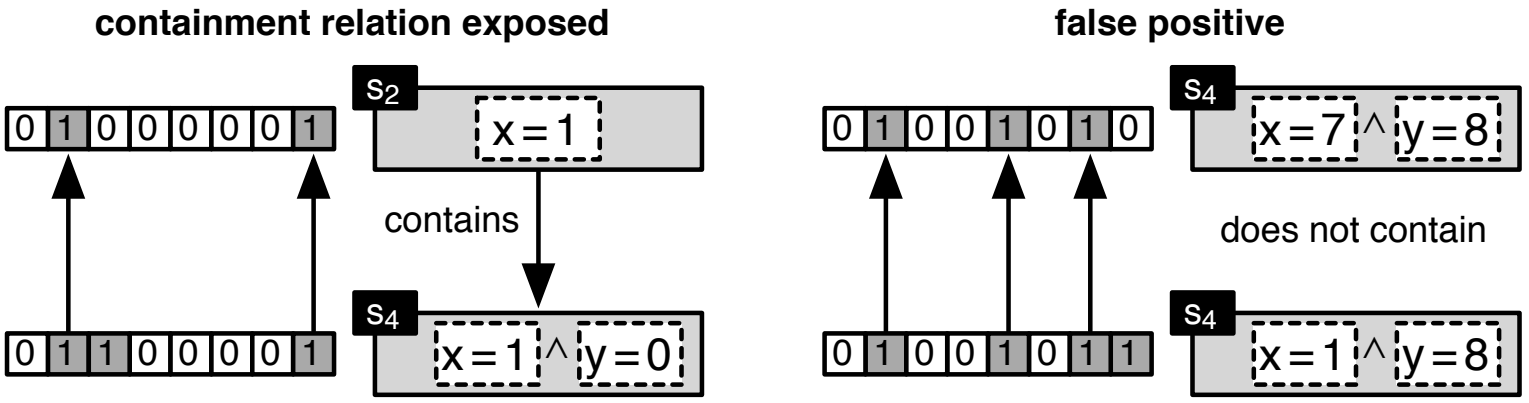
- A subscription S_3 contains S_5 if every event matching S_5 also matches S_3
- Allows positive and negative matching
  - Organize subscriptions in a DAG
  - Match from leaves: use positive matching to select entire branches of the tree
  - Match from root: use negative matching to discard branches
- Containment *optionally* supported by ASPE through additional matrices
- Reason: not always desirable as it allows statistical attacks on subscriptions similarities

Containment confidentiality

- The ability to determine containment may lead to confidentiality breaches
- Attack scenario
  - Build containment graph
  - Subscriptions with the same value for equality contraints with some value *a* for an attribute grouped together
  - Use domain-specific knowledge (e.g., distribution of interests) to map subscriptions to values
- Encrypted processing schemes typically make containment determination optional
  - ASPE: requires additional matrices



Bloom filters and confidentiality



- Adding extra information raises the power of the attacker
- Ability to determine part of the containment relationships from the BF
  - Bits of the (supposed) containee filter are all set in container filter
  - With false positives (collisions) and false negatives (inequalities)
- This might be a problem if containment confidentiality is a requirement
- Solution: containment obfuscation by truncation: unset random bits in subscriptions BF
- Privacy/performance tradeoffs analyzed in [Barazzutti et al., TDCS 2015]

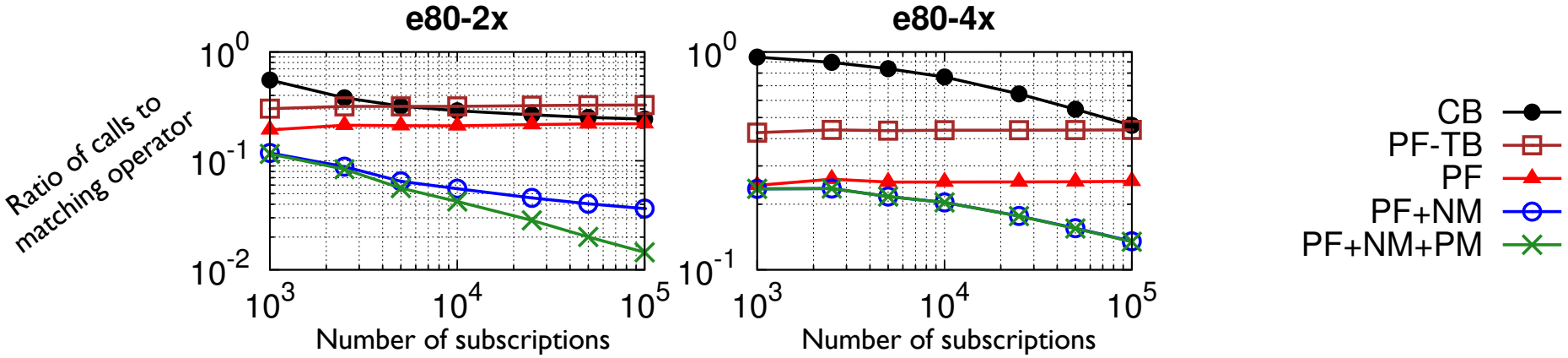
Prefiltering performance

- 250,000 publications from Yahoo! finance
- Synthetic subscriptions

Workload name	Number of equality predicates	Number of attributes	Distribution of values
e100	100% : 1 eq. pred.	8–11 (default)	Uniform
e80	20% : 0 eq. pred. 80% : 1 eq. pred.	8–11 (default)	Uniform
e80-2x	20% : 0 eq. pred. 80% : 1 eq. pred.	2× more	Uniform
e80-4x	20% : 0 eq. pred. 80% : 1 eq. pred.	4× more	Uniform
e+-2x	15% : 0 eq. pred. 60% : 1 eq. pred.	2× more	Uniform
e+-4x	15% : 2 eq. pred. 10% : 3 eq. pred.	4× more	Uniform
e80-z	20% : 0 eq. pred. 80% : 1 eq. pred.	8–11 (default)	Zipf on symbol
e80-z+	20% : 0 eq. pred. 80% : 1 eq. pred.	8–11 (default)	Zipf on all attributes
e100-z+	100% : 1 eq. pred.	8–11 (default)	Zipf on all attributes

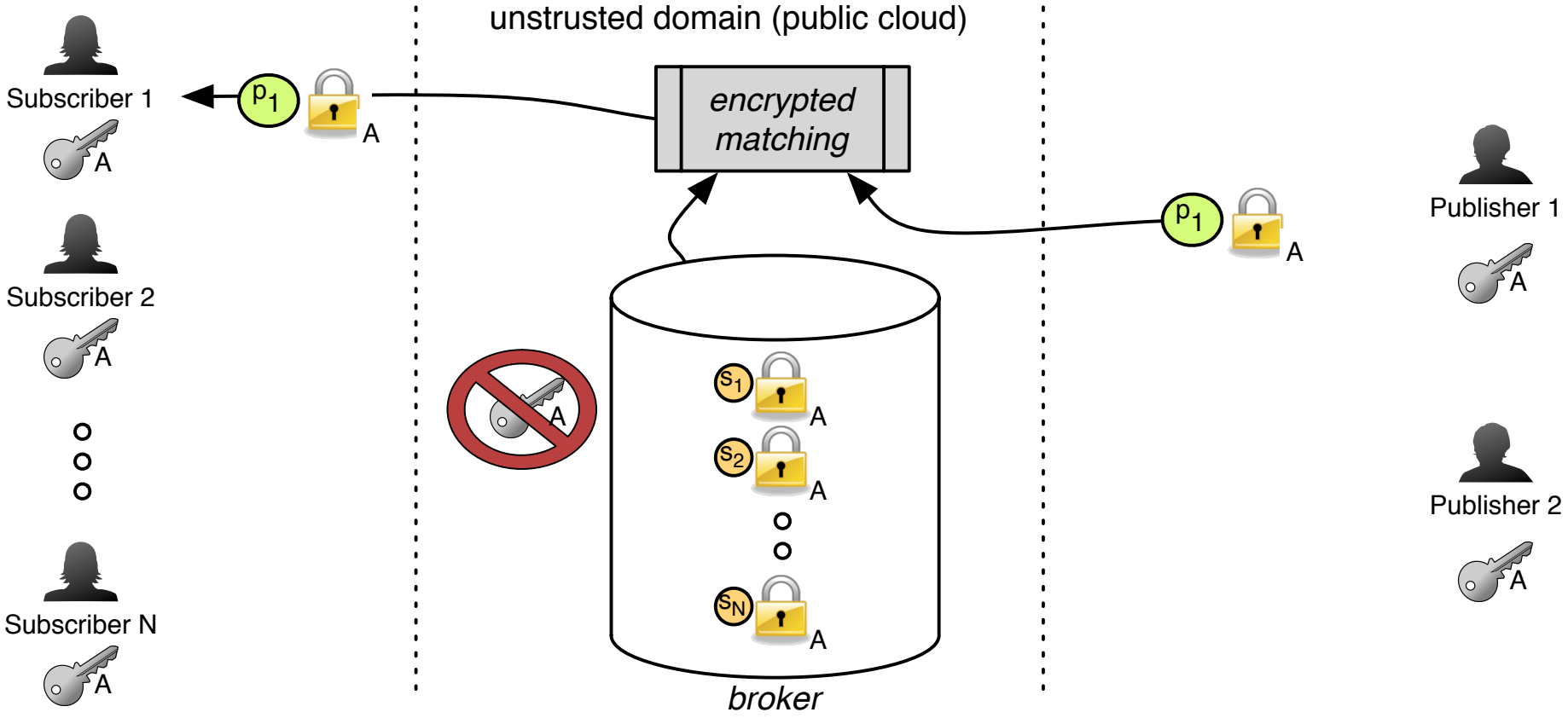
- Variants
  - CB: no prefiltering, only a containment graph
  - PF: pre-filtering (but no containment information)
  - PF-TB: truncated pre-filtering
  - PF+NM: pre-filtering with negative matching
  - PF+NM+PM: pre-filtering with negative and positive matching

Prefiltering performance (extract): avoiding calls to encrypted matching

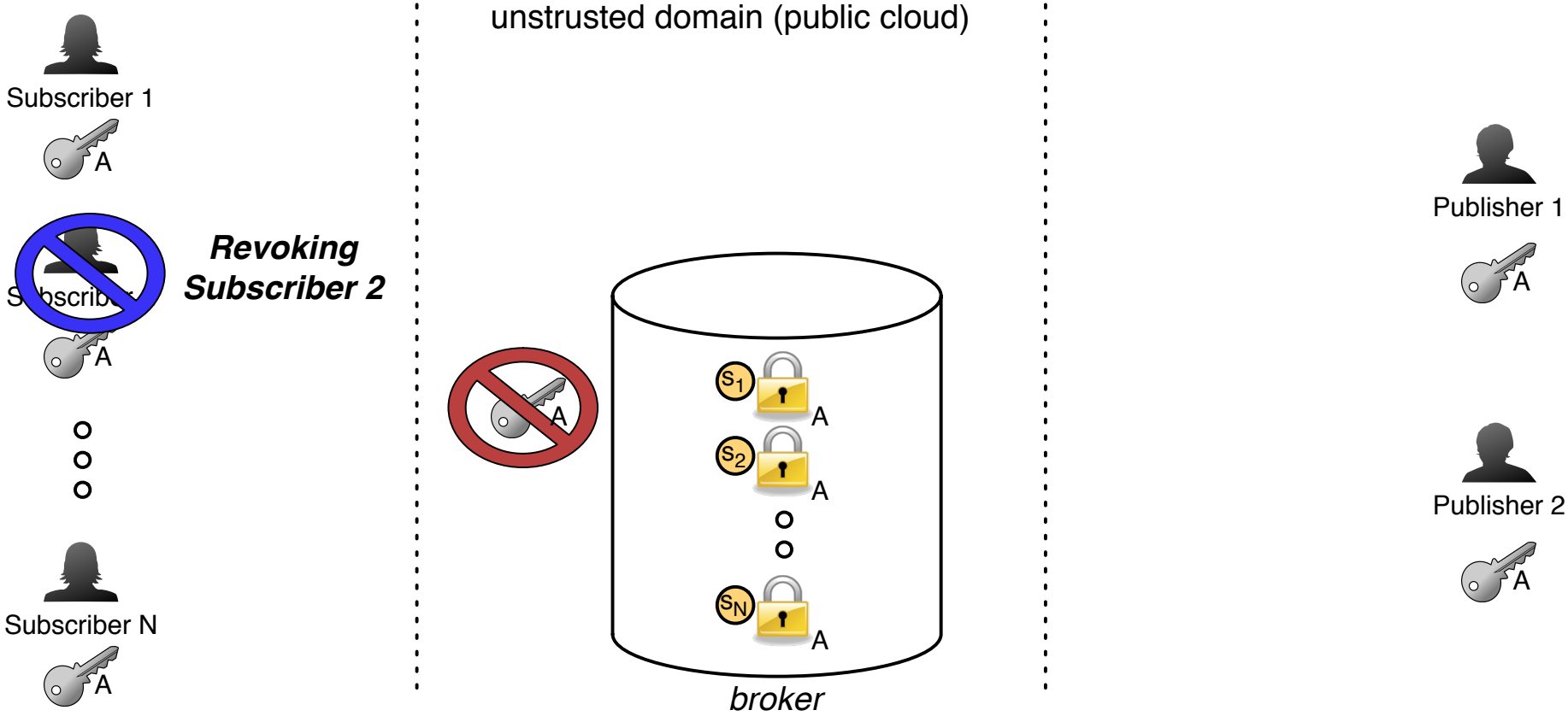


- Evolution of the number of calls to the costly encrypted matching function, when varying the # of subscriptions
  - 128 bits filters, 3 hash functions (truncation : keep 1 hash function)
- PF and PF-TB do not use containment: constant cost
- Using prefiltering improves over using only a containment graph
  - Reduces number of calls to encrypted matching in all cases
- Combining both techniques allows the greatest reduction

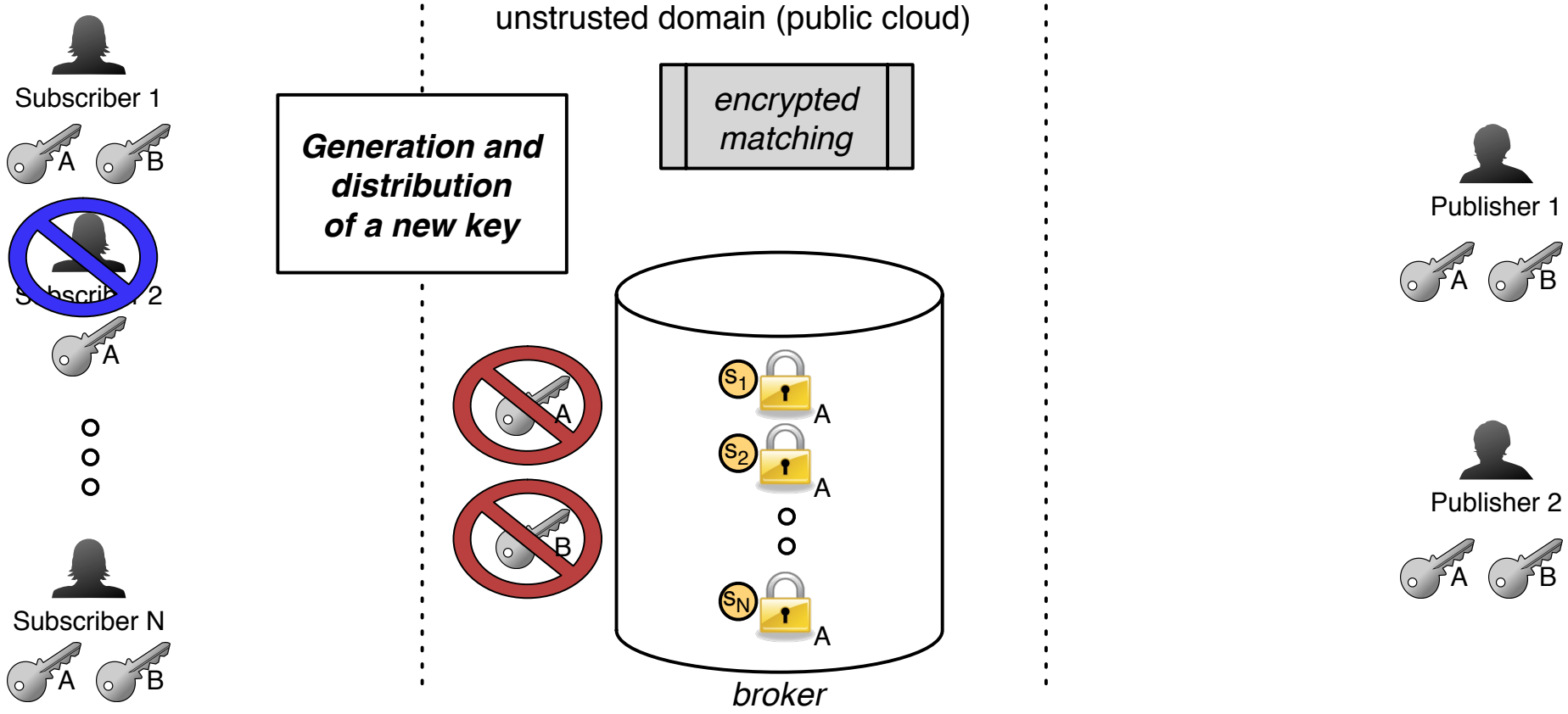
Confidentiality-preserving pub/sub and the issue of key updates



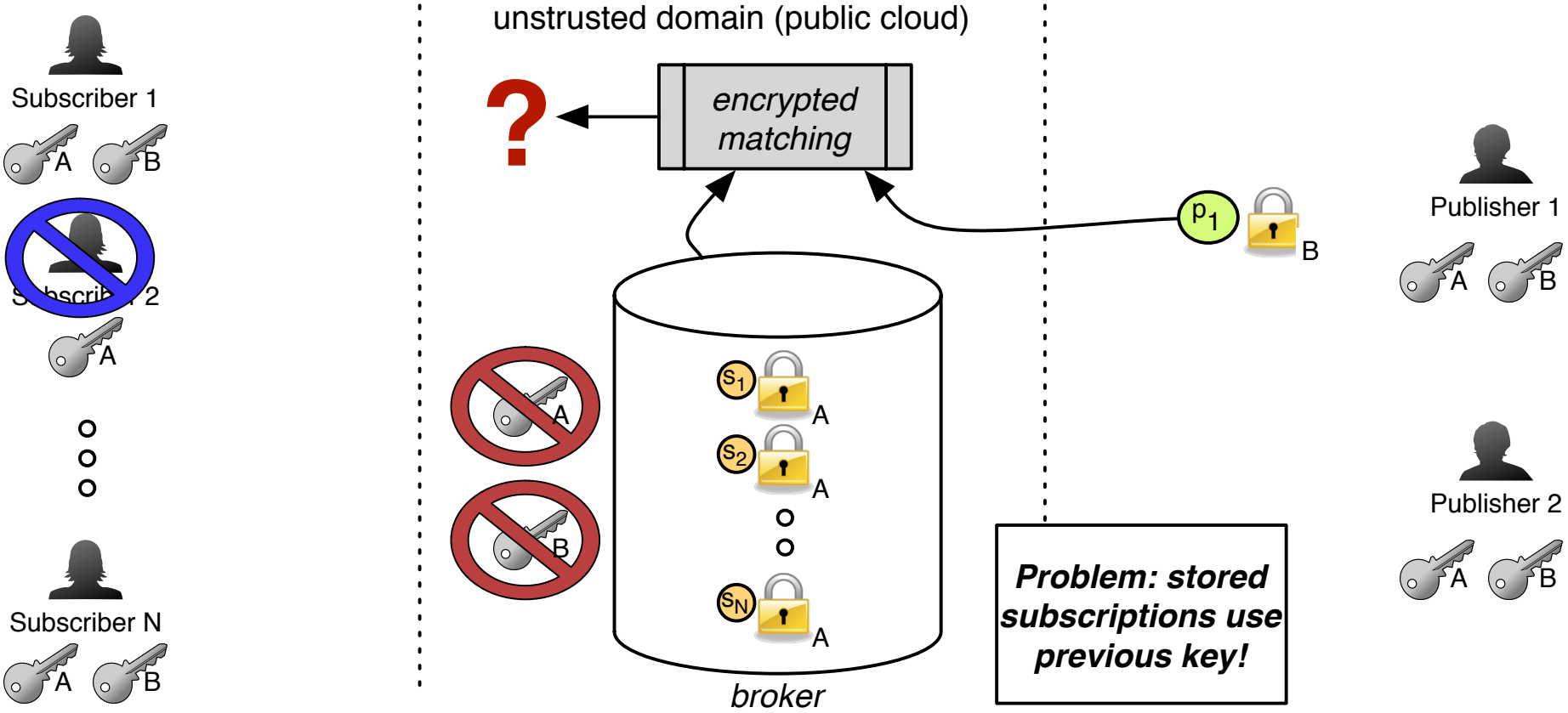
Confidentiality-preserving pub/sub and the issue of key updates



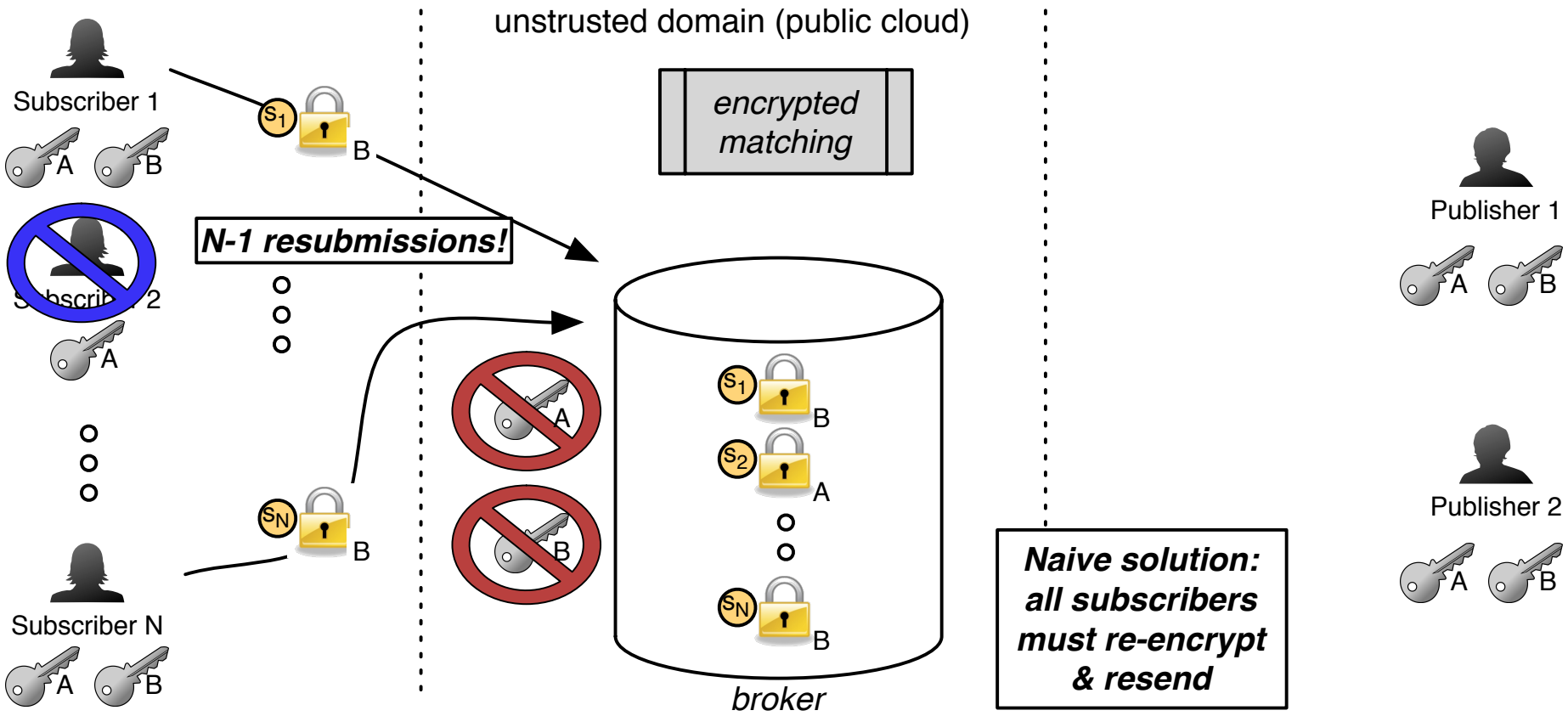
Confidentiality-preserving pub/sub and the issue of key updates



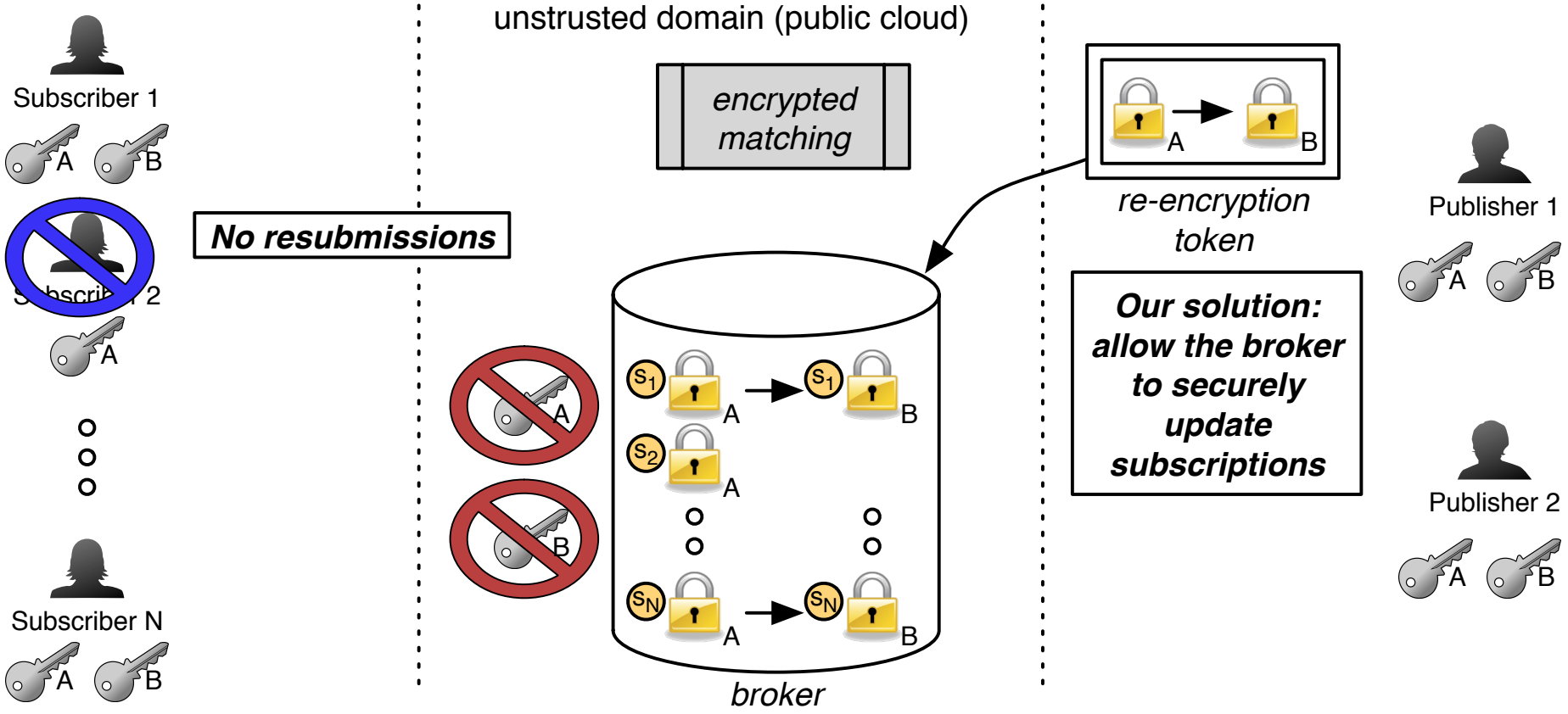
Confidentiality-preserving pub/sub and the issue of key updates



Confidentiality-preserving pub/sub and the issue of key updates



Confidentiality-preserving pub/sub and the issue of key updates



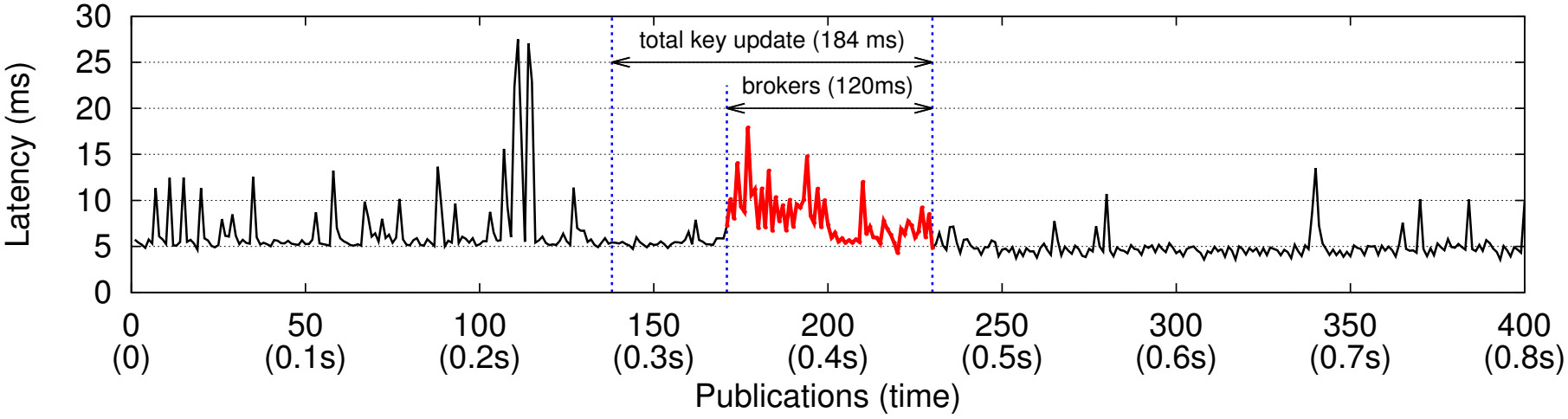
PS-recrypt: in-broker subscription re-encryption

- Our proposal: allow brokers in untrusted domains to re-encrypt subscriptions themselves
  - Does not require subscribers to re-submit their subscriptions
    - Faster process (smaller window of opportunity)
    - No need for stable storage of subscriptions @ subscribers
  - Goals
    - The untrusted broker must not know the previous key, the new key, or be able to derive any plaintext subscription/publication
- Example with the ASPE scheme
  - Key update token K_R and transformation τ
  - Allow updating constraints enc. with matrix M^T to constraints enc. with N^T
  - For ASPE, only the matrix key changes
  - τ is thus simply $S'_{N^T i} = K_R S'_{M^T i}$ with $K_R \triangleq (N^T r')M^{T^{-1}}$
    - r' : random scaling factor for obfuscation
  - Scheme proven correct (can only form system of eq. with ∞ of solutions)

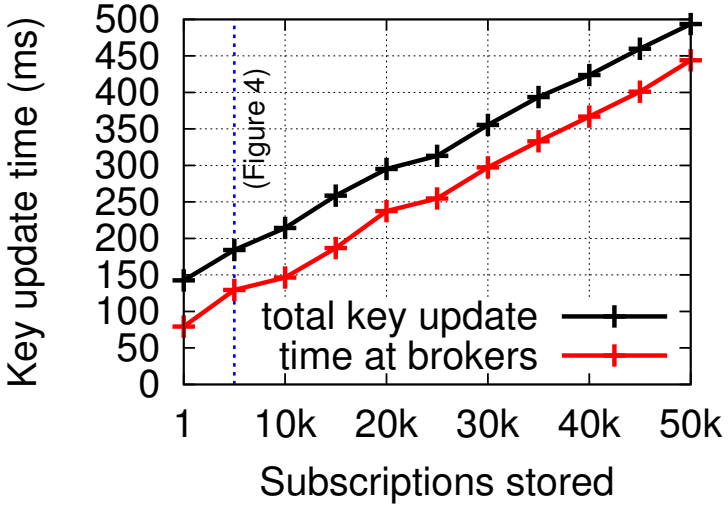
Evaluation

- Implemented in the StreamHub high-performance content-based pub/sub engine with ASPE
  - Implementation in C++ (with Crypto++)
  - Key management through a trusted ZooKeeper
- 15 cores dedicated to matching in parallel (subscriptions replicated, publications round-robin)
- Metrics
  - Delay between publisher and subscribers (same cluster)
  - Time for key update
  - Comparison to naive solution

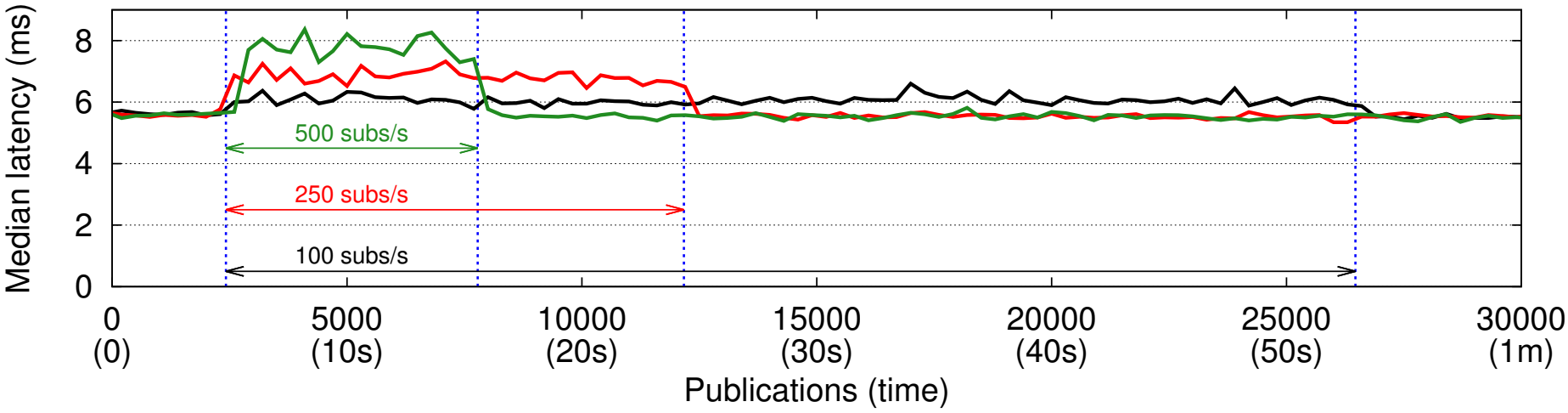
Impact on notification delays



- Constant flow of 500 publications/second
- 5,000 subscriptions (top)
- Varying # subscriptions (right)



Comparison with naive solution



- (naive) Comparison with the naive solution
  - All subscribers re-encrypt and resubmit their subscriptions
- 5,000 subscriptions, 500 publications / second
  - Presents median delays (each 0.1 second)
- Caping the rate of subscriptions reception at the network level

New trends and open directions

- Use of secure hardware (e.g., Intel SGX)
  - Filtering operation on plaintext filters, but only in protected *enclaves*
  - Early tests show significant performance gains, but requires specific system and hardware support
- Encrypted processing using more complex subscriptions and publications models
  - Regular expressions, bag of words, functions over multiple attributes, etc.
  - Integrate advances in encrypted information processing / Secure Multi-Party Computation
- Encrypted matching to be integrated in secure pub/sub solutions and existing trust and key management systems

Conclusion

- Deploying pub/sub services in public clouds is appealing
  - Untrusted domain = confidentiality threats
- Traditional encryption does not allow routing in untrusted domains
  - Use encrypted matching
- Performance gap with plaintext matching
  - Prefiltering using Bloom filters
  - Preserves containment confidentiality is required, but take advantage of it if available
- Encrypted subscriptions stored in untrusted domains
  - Key update & re-encryption directly in untrusted domains

References

- **Confidentiality-Preserving Publish/Subscribe: a Survey.** Emanuel Onica, Pascal Felber, Hugues Mercier and Etienne Rivière (ACM Computing Surveys, June 2016)
- **PS-recrypt: Efficient Key Update for Privacy-Preserving Publish/Subscribe.** Emanuel Onica, Pascal Felber, Hugues Mercier and Etienne Rivière (Middleware 2015)
- **Efficient Privacy-Preserving Content-Based Publish/Subscribe with Prefiltering.** Raphaël P. Barazzutti, Pascal Felber, Hugues Mercier, Emanuel Onica and Etienne Rivière. (IEEE Transactions on Dependable and Secure Computing, 2015)
- **Elastic Scaling of a High-Throughput Content-Based Publish/Subscribe Engine.** Raphaël Barazzutti, Thomas Heinze, André Martin, Emanuel Onica, Pascal Felber, Christof Fetzer, Zbigniew Jerzak, Marcelo Pasin and Etienne Rivière (ICDCS 2014)
- **StreamHub: A Massively Parallel Architecture for High-Performance Content-Based Publish/Subscribe**. Raphaël P. Barazzutti, Pascal Felber, Christof Fetzer, Emanuel Onica, Jean-François Pineau, Marcelo Pasin, Etienne Rivière and Stefan Weigert. (DEBS 2013)

UNIVERSITÉ DE NEUCHÂTEL

Efficient confidentiality preservation for cloud-supported content-based publish/subscribe

Etienne Rivière

University of Neuchâtel, Switzerland

Questions?

SEC2 - July 2016

etienne.riviere@unine.ch