New Constructions and Practical Applications for Private Stream Searching

(Extended Abstract)

John Bethencourt
CMU

Dawn Song
CMU

Brent Waters
SRI

Searching for Information

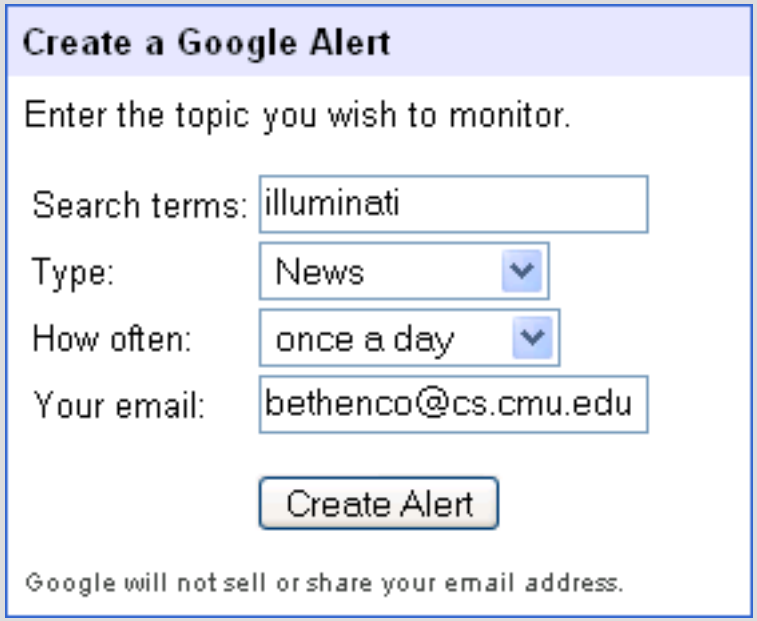
- Too much on-line info to download
  - WWW pages
  - Message boards
  - Web email, USENET
- Search
  - User specifies search criteria
    - Textual keywords
  - Server returns relevant content

Motivation for Private Searches

- What if keywords are secret?
  - Personal privacy
    (example query: “lice removal”)
  - Commercial interests
    (“takeover bid”)
  - Intelligence gathering
    (“Al Qaeda”)

- We want *private* searches
  - Client gives server encrypted query
  - Server runs search algorithm, returns data
  - Client recovers matching content
  - Server does not know what client searched for

Practical Example: Google News Alerts

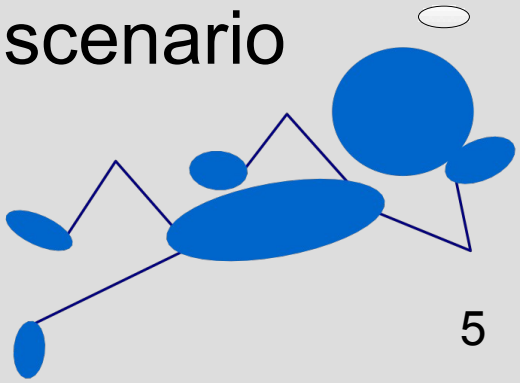


- Google News
  - Google continuously crawls 4,500 news sources
  - Estimated 135,000 news articles each day
  - Alerts service
    - User registers search keywords
    - Matching articles emailed as they are discovered

Private Google News Alerts

- What about a *private* alerts service?
  - User registers encrypted search keywords
  - Periodically receives matching articles
  - Google remains oblivious!

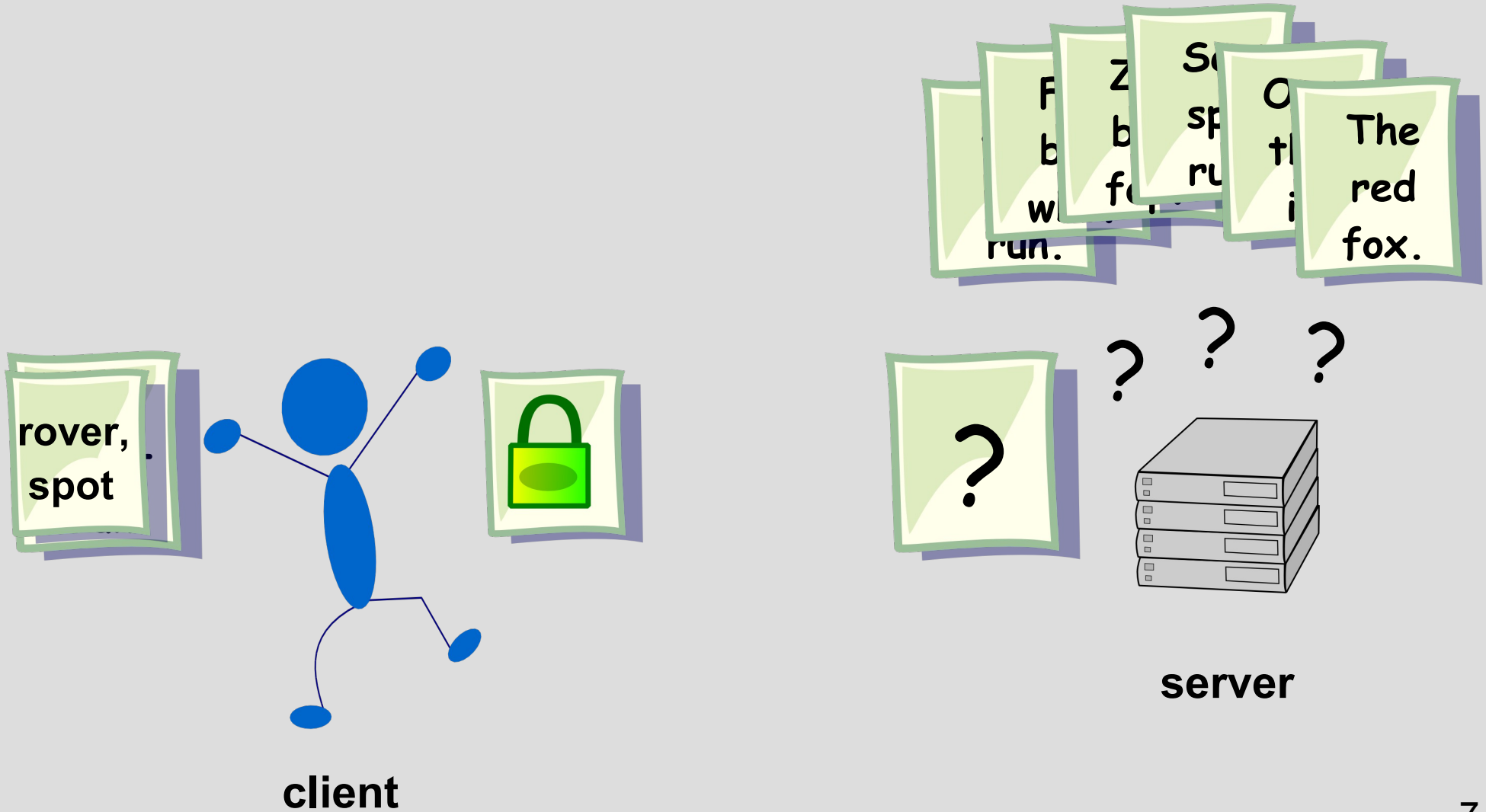
- Our techniques enable private alerts service
  - Previous schemes not practical for this scenario
  - Our results later in talk



Contributions

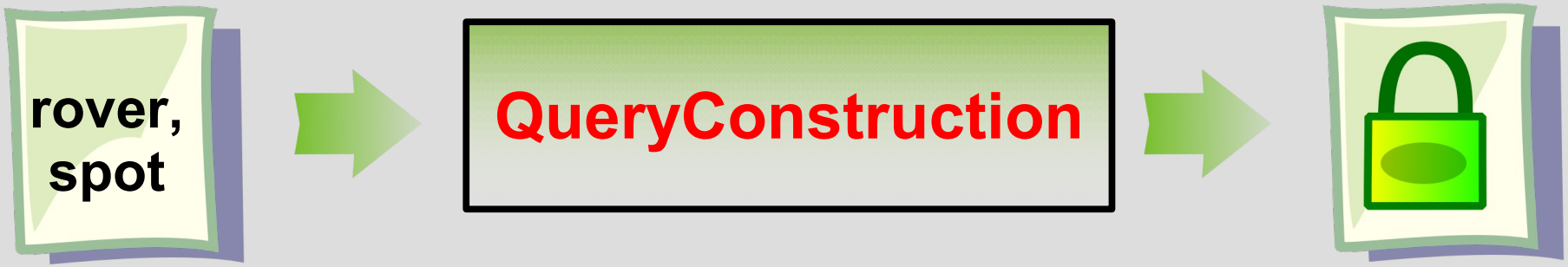
- New scheme for private stream searching
- Several novel constructions
  - Reconstructing matching documents by solving linear systems
  - Encrypted Bloom filters
- Practical analysis demonstrating feasibility
  - Orders of magnitude more efficient than previous work [Ostrovsky-Skeith CRYPTO 2005]

Basic Architecture: Overview



Basic Architecture
Step 1: Query Construction

- Client runs QueryConstruction algorithm to produce encrypted query
  - Queries are disjunctions of textual keywords (simplest case)
  - Variations and extensions possible
- Sends encrypted query to server

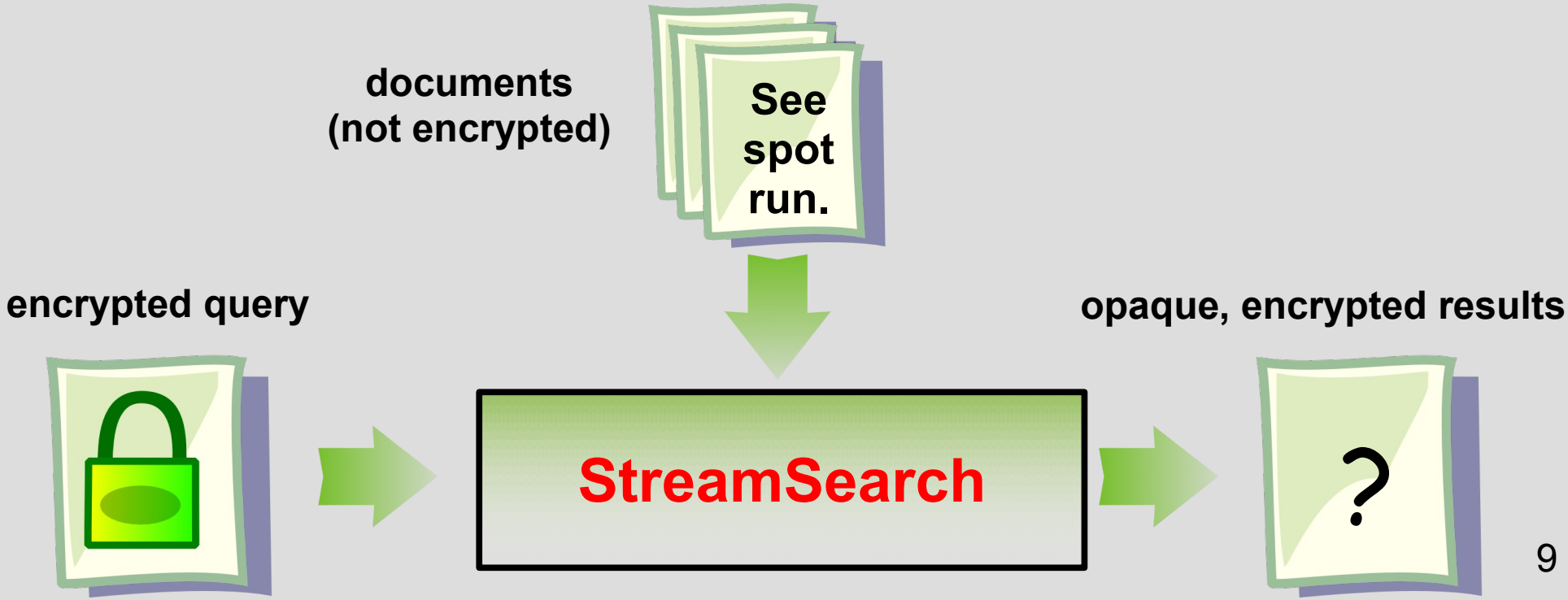


raw keyword list

encrypted query

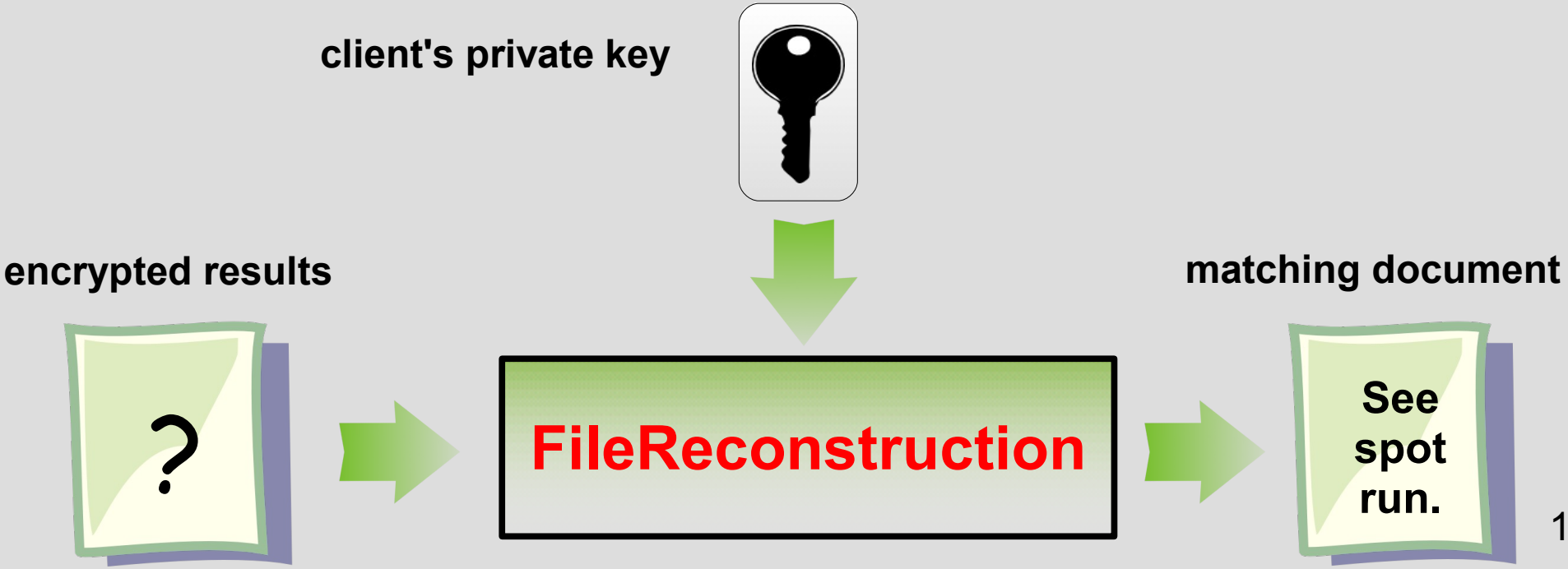
Basic Architecture
Step 2: Executing the Search

- Server gets encrypted query
- Server runs StreamSearch algorithm
  - Processes documents to produce encrypted buffers
  - Server remains unaware of search keywords



Basic Architecture
Step 3: Recovering the documents

- Client gets encrypted results
- Runs FileReconstruction algorithm
  - Uses private key
  - Recovers original matching documents



Scheme Highlights: Homomorphic Encryption

- How is this accomplished?
  - Scheme in full paper
  - Just highlights, general flavor here

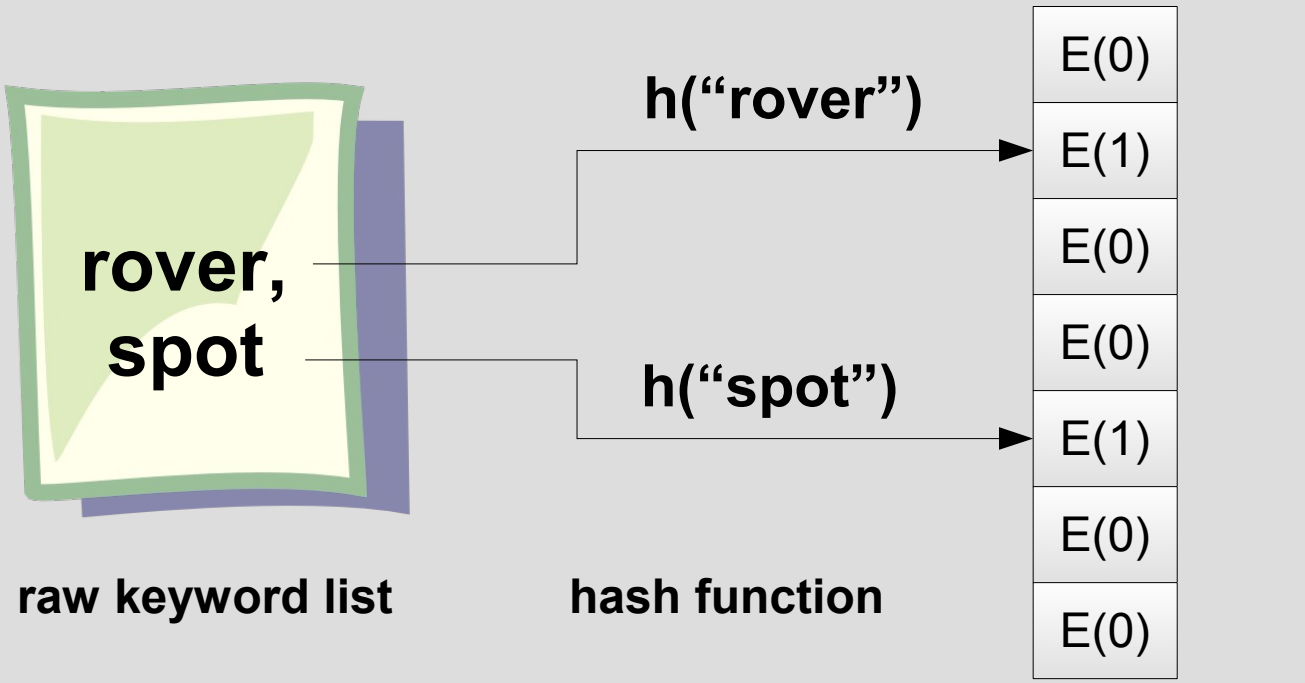
- Homomorphic encryption
  - Paillier cryptosystem
  - Public key system
  - Additive homomorphism:

$$E(a) \cdot E(b) = E(a+b)$$

  - Allows rudimentary computations on encrypted data
  - Lets server meaningfully use encrypted query

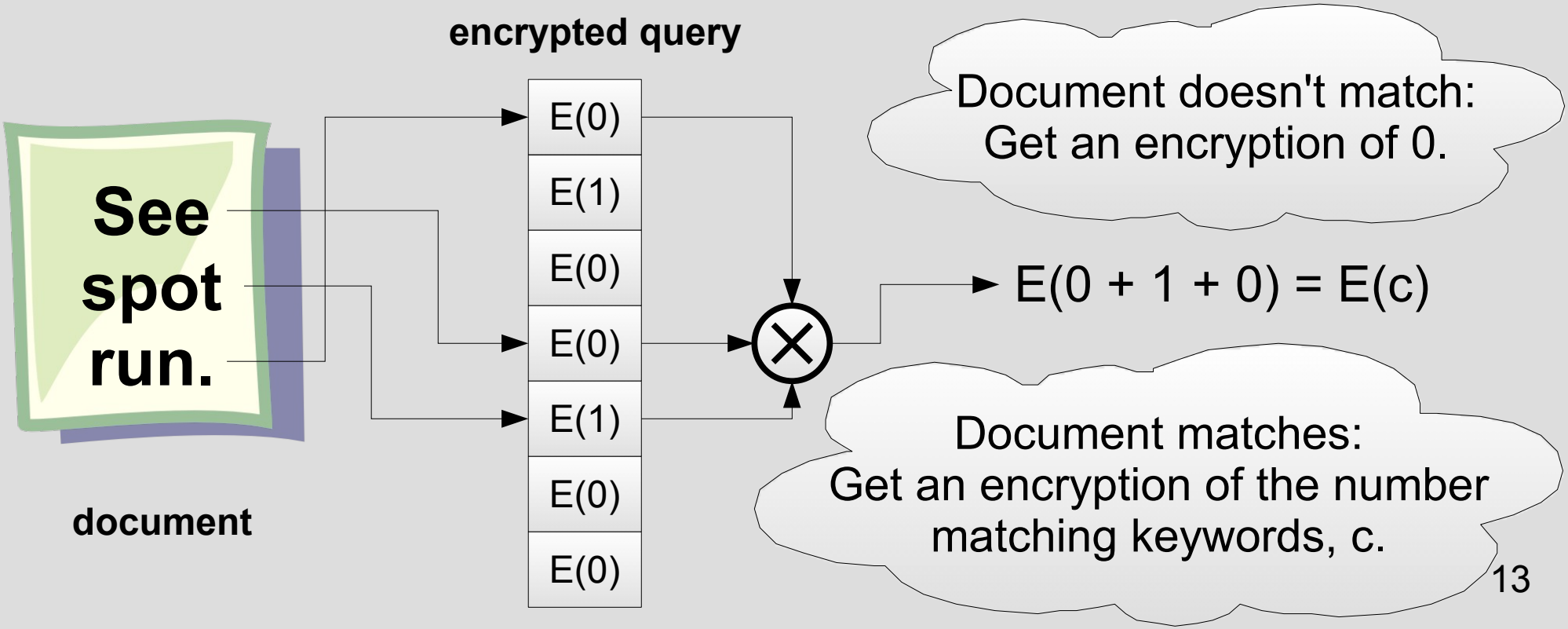
Scheme Highlights: Encrypted Queries

- Queries are encrypted hash table
  - Each cell is encryption of zero or one
  - Probabilistic encryption
  - Homomorphic encryption



Scheme Highlights: Conducting the Search

- Server can then obtain encryption of number of keyword matches
  - Used to bootstrap rest of algorithm



Scheme Highlights: Recovering the Documents

- Client decrypts returned data with private key
- Can construct linear system in the matching documents
  - Special metadata returned from server
  - Encrypted Bloom filter
- Solves linear system for documents
  - System solveable if random 0,1 matrix is non-singular
  - Almost always the case

Singular with exponentially low probability with respect to size! [Komlós, Tao-Vu STOC 2005]

random (0,1) matrix

contents of decrypted buffer

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}$$

Communication Complexity

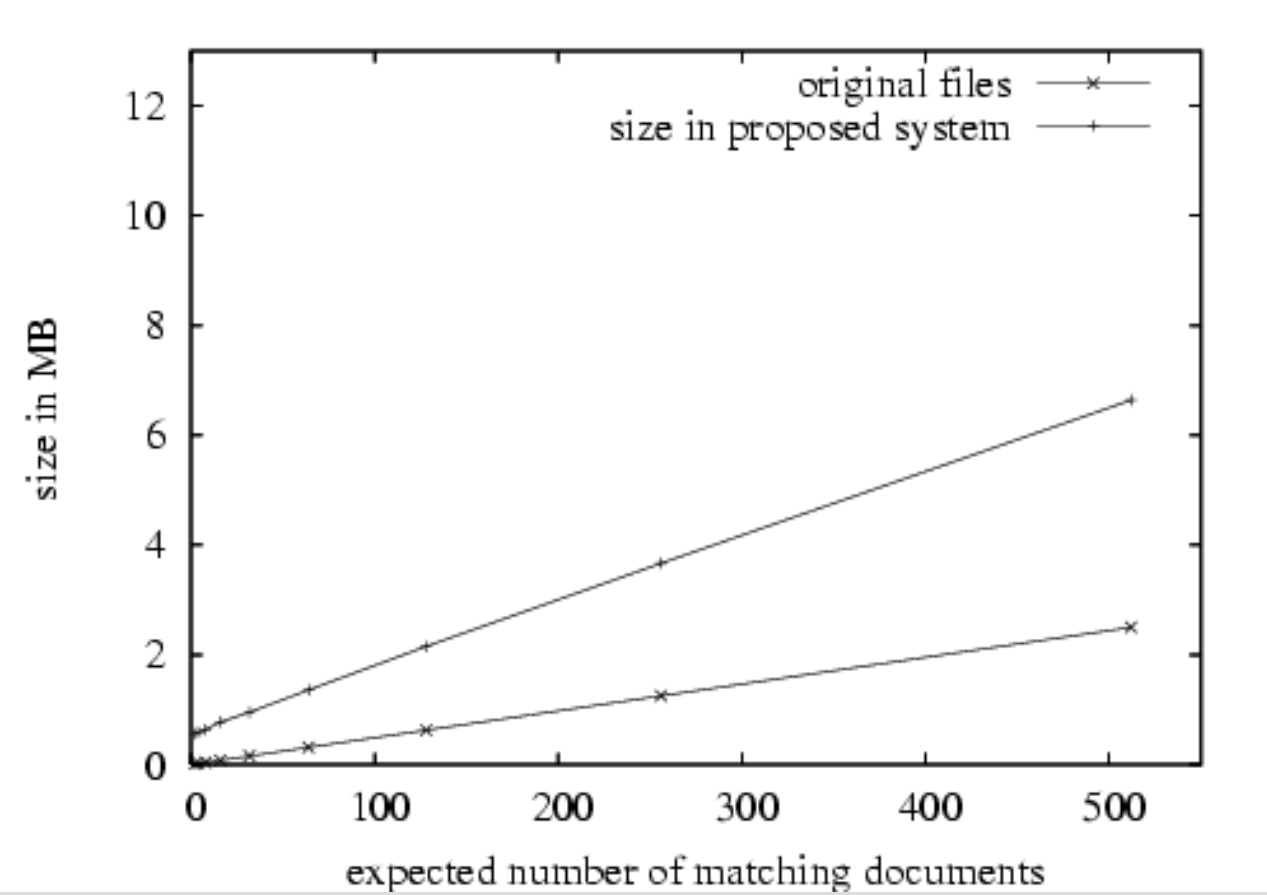
- Performance / robustness tradeoffs in size of queries and results
- New scheme
  - O(n) in the size of the returned content for the bulk content of the matching files
  - Some metadata is O(n log(t/n)) or O(n log n)
- Previous scheme
  - O(n log n) for bulk content
  - Much higher multiplicative constants

Practical Analysis: Private News Alerts

- Private news alerts scenario
  - 135,000 news articles searched each day
  - Retrieve results four times per day
  - 5KB article size (text only, compressed)
  - Up to 2000 matching articles per day
- Performance of our scheme
  - Query size: 4-30MB
  - Server processing time: 500ms per article
  - Communication size: 500KB – 7MB per time period
  - Client reconstruction time: 0 – 15min

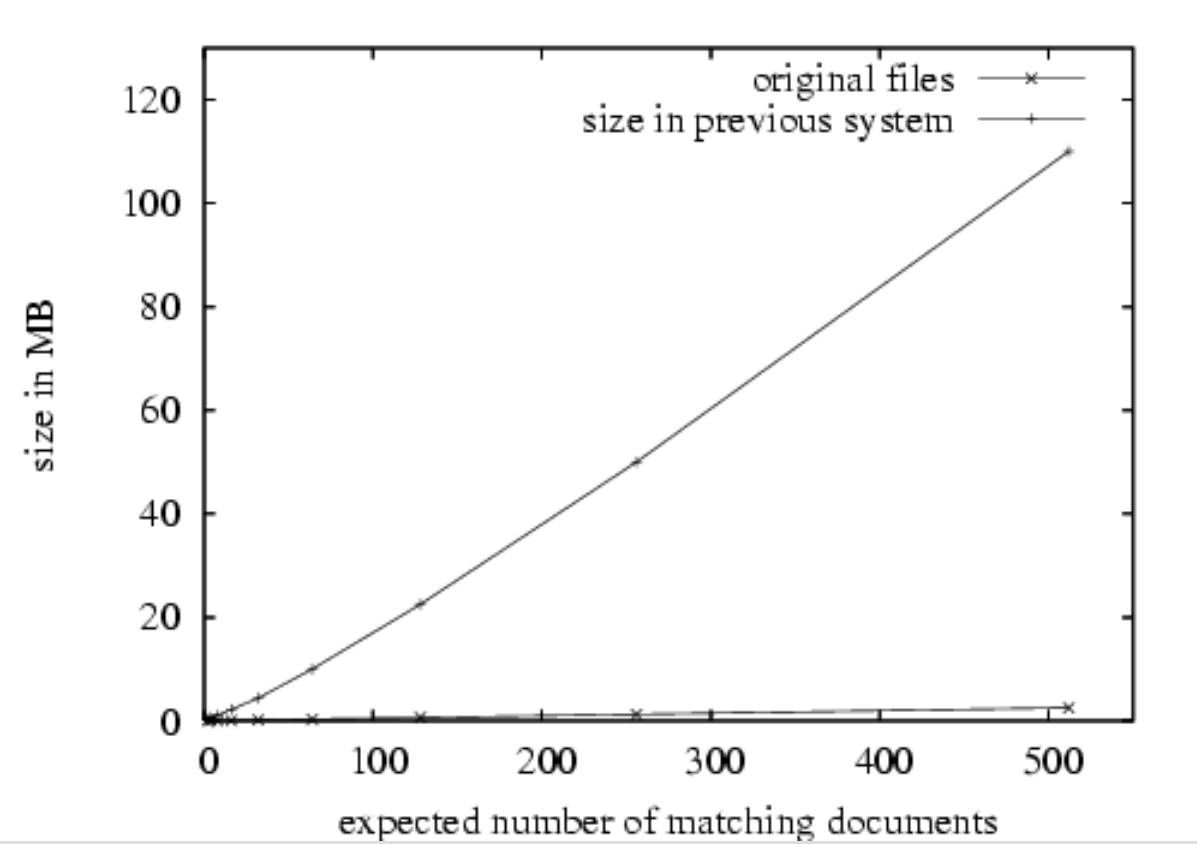
Communication Overhead: Our Scheme

- Low overhead
  - Factor of 1.2 before inflation due to Paillier
  - 2.4 after



Communication Overhead: Previous Scheme

- Previous scheme impractical
  - Communication overhead: 40 times
  - Reconstruction time: 4.7 hours



Conclusion

- Efficient system for private stream searching
  - Low communication overhead
    - Factor of 2.4 over actual matching documents
    - Previous system up to factor of 40
  - Extensions also developed
    - Arbitrary length documents
    - More complex queries
    - Other stuff

- For more info see full version of paper
  - http://www.cs.cmu.edu/~bethenco/search.ps
  - Includes details of actual scheme!
  - Also security proofs, etc.