SSARES: Secure Searchable Automated Remote Email

A usable, secure email system on a remote untrusted server

Adam J. Aviv, Michael E. Locasto, Shaya Potter
Angelos D. Keromytis

Columbia University Network Security Lab

Trends

- More and more information is being stored on remote servers
  - think Google
  - but also your organization's IMAP server
- How do we protect all this information "at rest" on a remote server, while still provide the same service?
- privacy, protection, and convenience
- Good example of this service is email

The Problem

- Two options for email storage
  - remote
  - local
- Remote Email Servers have full access to email
- PGP?
- Complete Encryption
  - breaks what's nice about remote service
  - no remote searching - a service we need and use

Our Solution

- SSARES: Secure Searchable Automated Remote Email Storage
- Public/Private Key Encryption Algorithm
  - no private information ever at the server
- Complete Email Encryption but searchable by server
- Built using a combination of PEKS and Bloom Filters

Threat Model

- Two types of attackers
  - break into server, download mailbox, and do off-line analysis
  - observes the system in action and watches how messages are matched to try and determine the contents
- Once server is compromised
  - all newly arriving mail trivially compromised
  - prior received mail still protected

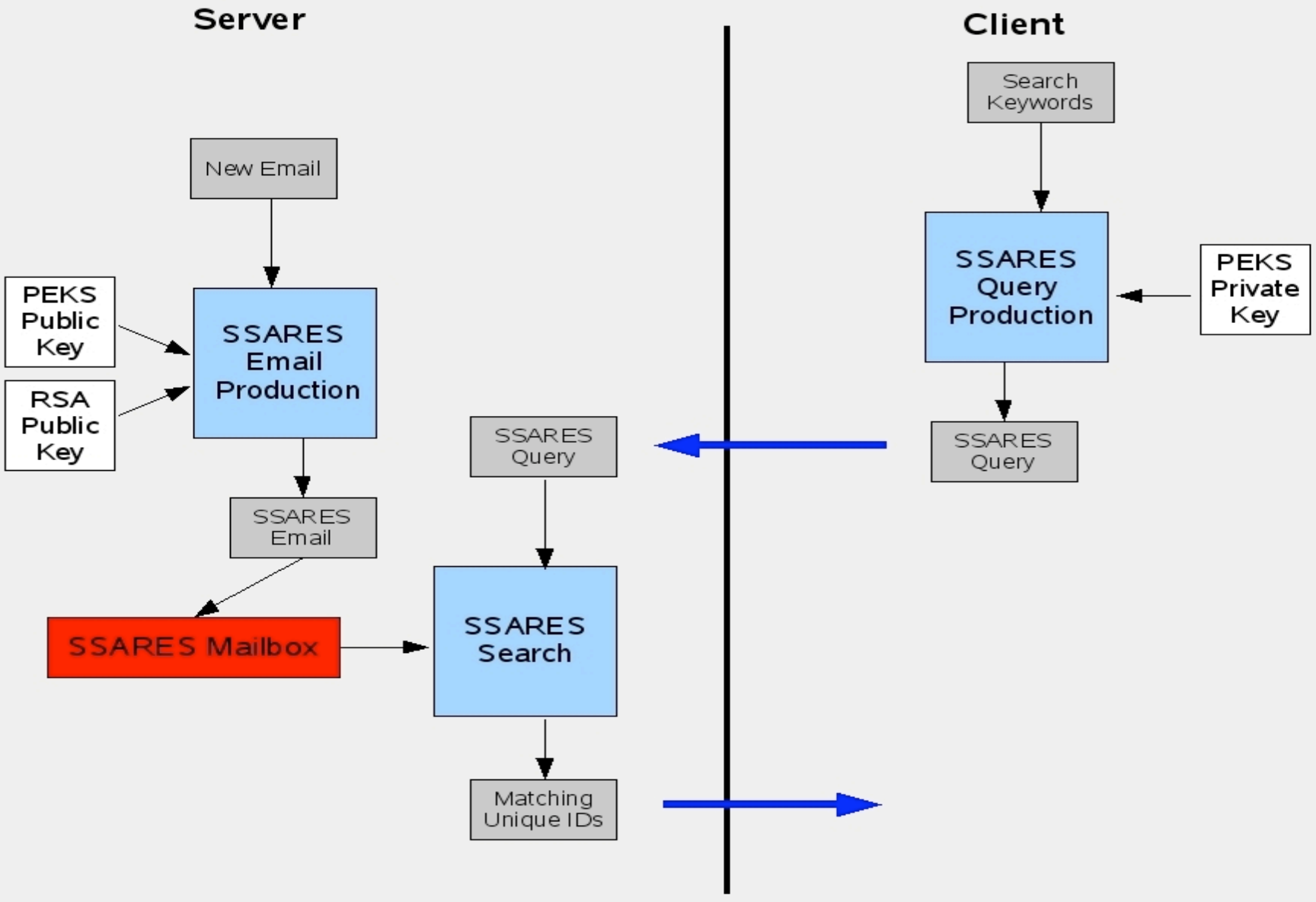
Naïve Solutions

- Hash Table
  - client will have likely keywords
  - possibility of a dictionary attack
- Encrypted Hash Table
  - can't search until downloaded hash table
  - how big will this hash table be?
- The search routine needs protection
  - should stay autonomous

Goals

- Transparency
  - The actions of the user do not need to change
  - The actions of the sender do not need to change
- Autonomy
  - There is no additional interaction between the client and the server needed
  - All cryptography can be done without the client private information or client interaction

Design


Server
Client
New Email
Search Keywords
PEKS Public Key
SSARES Email Production
SSARES Query Production
PEKS Private Key
RSA Public Key
SSARES Email
SSARES Query
SSARES Query
SSARES Mailbox
SSARES Search
Matching Unique IDs

PEKS

- Public Key Encryption with Keyword Searching
- Server encrypts keyword with user's public key to create a PEKS
- User encrypts keyword with private key to create a Trapdoor
- Server can securely compare PEKS and Trapdoor to determine if they represent the same keyword

PEKS: functional definition

KeyGen(s): generate public/private key pair A_{pub} , A_{priv}

PEKS(A_{pub} , W): given a public-key, A_{pub}, and a word, W produce a PEKS, S.

Trapdoor(A_{priv} ,W): given a private-key, A_{priv}, and a word, W, produce a trapdoor, T_W

TEST(A_{pub} , S, T_W): given public key A_{pub}, trapdoor, T_W, and PEKS S = PEKS(A_{pub} , W'), output match when W=W', no match otherwise

PEKS: definitions

- Two Groups, G1, G2 of prime order p
- Bilinear map [e: G1xG1-->G2]
- Two Hash Functions

  H1 : {0, 1}* --> G1

  H2 : G2 --> {0,1}log_p

PEKS: generation

- KeyGen(p): security parameter determines the size, p, of the groups G1, G2.
  - pick a random α and a generator g of G1
  - output: $A_{pub} = [g, h=g^{\alpha}]$, $A_{priv} = \alpha$
- PEKS(A_{pub}, W): compute t = e(H1(W), h^r), where r is a randomly generated
  - output: [g^r , H2(t)] = S[A,B]

PEKS: testing

- Trapdoor(A_{priv},W): T_W=H1(W)a which is contained in G1
- Test(A_{pub},S[A,B],T_W): if H2(e(T_W,A))=B then it is a match and no match otherwise

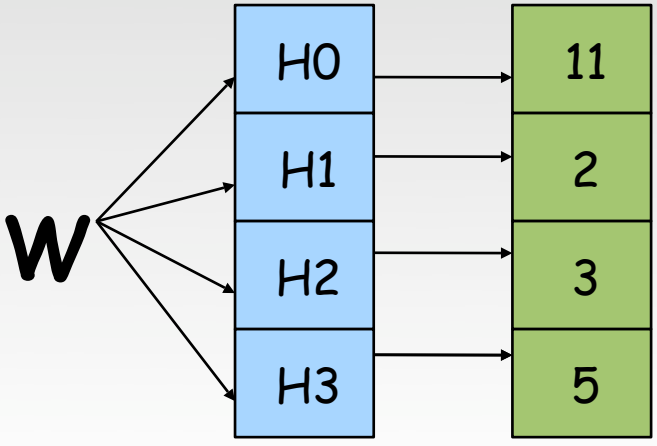
Our Contribution

- PEKS slow
  - 100 keywords per message, 1000 messages
  - 100,000 PEKS to test for an exhaustive search
- Minimize number of PEKS to test
  - only test PEKS likely to match
- Bloom Filters with a high error rate
  - eliminate 75% of message before testing any PEKS
  - High error rate limits information leakage

What is a Bloom Filter?

- Space efficient and time efficient way to test set membership
- Non-invertible
- No false negatives
- Probabilistic false-positives or error-rate
  - number of hash functions
  - number of words represented in the filter

Bloom Filters



- W or W' will always match this filter.
- But so will words not in the Filter

Output of Hashing

W	0	0	1	1	0	1	0	0	0	0	0	1

OR Current State

W'	0	0	0	1	0	0	1	1	0	0	1	0

Resulting Filter

F	0	0	1	1	0	1	1	1	0	0	1	1

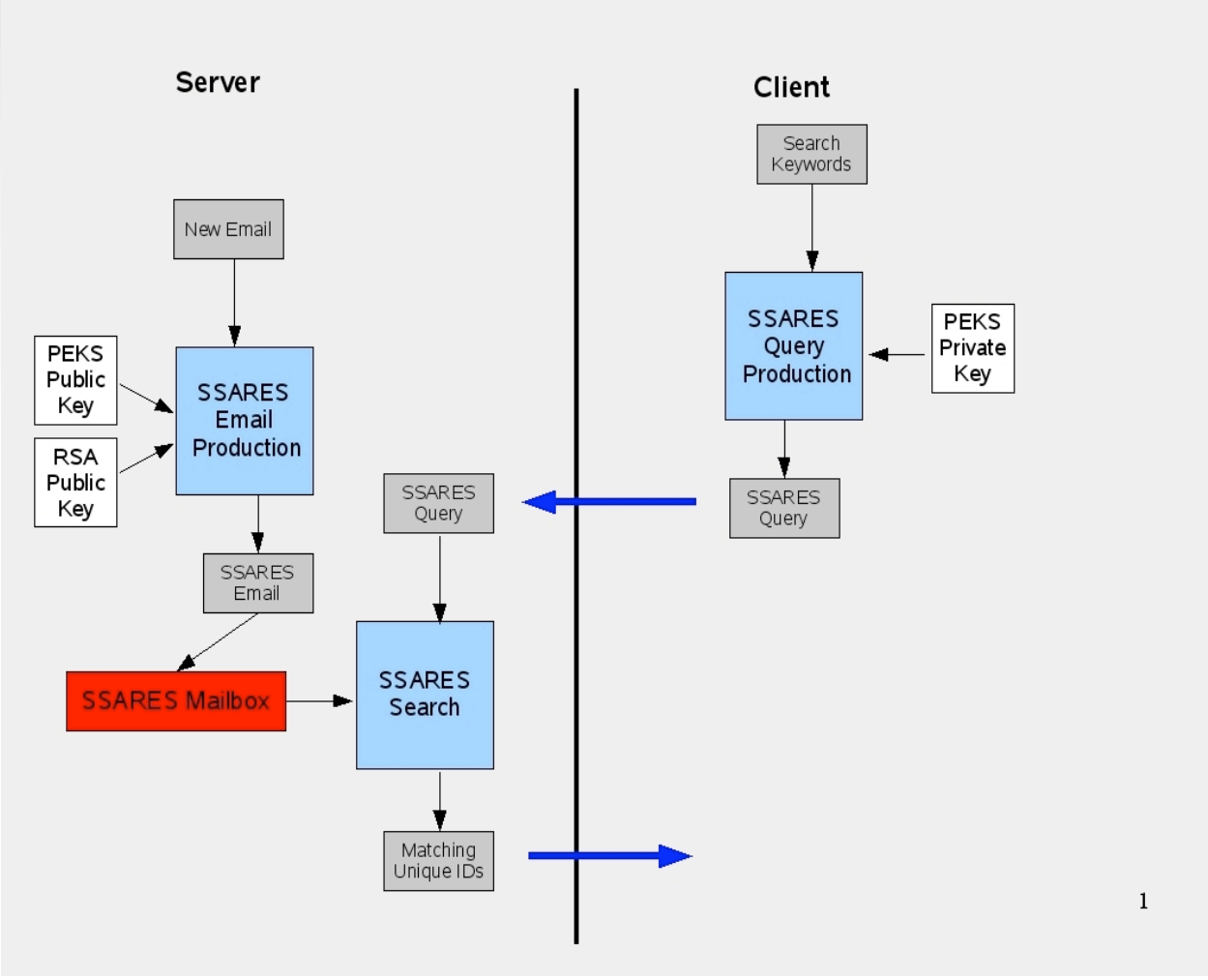
False-Positive Filter

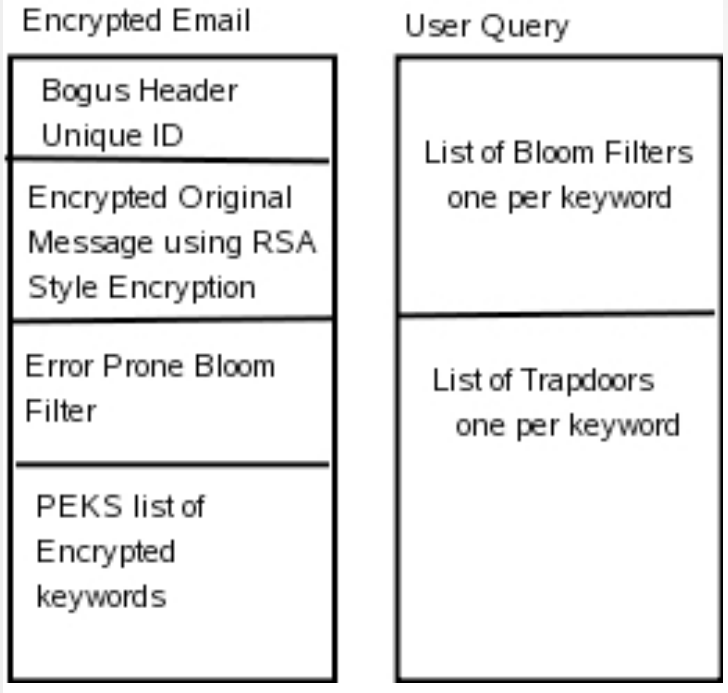
E	0	0	1	0	0	0	1	1	0	0	0	1

Error Prone Filters

- Normal error rate very low - much less then 1%
  - could lead to a dictionary attack
- We build error in – roughly 25%
  - eliminate 75% of the messages quickly
- Much harder to do a dictionary attack
- No error in query filters
  - results in false-negatives

Constructions





Additionally

- Divide PEKS lists into fields by message parts
  - To:, From:, Body:, Attachments:, etc
  - less PEKS to test, more precise searching
- Alpha-Sorting
  - each PEKS associated with unencrypted first letter of the keyword it represents
  - trapdoor comes with the unencrypted first letter

Implementation

- PEKS and Bloom Filter command line applications written in C
- Python wrapper scripts specific for each component

Evaluation

- Evaluated in three parts
  - email production, query production, searching
- Sample set of email from Enron data set
  - 100 emails

SSARES Email Production

- Average time of encryption 17 seconds
  - worst case 3 minutes
- 37x increase in size
- Both time and size are dependent on the number of keywords in the message
- Reasonable trade-offs – email slow transport

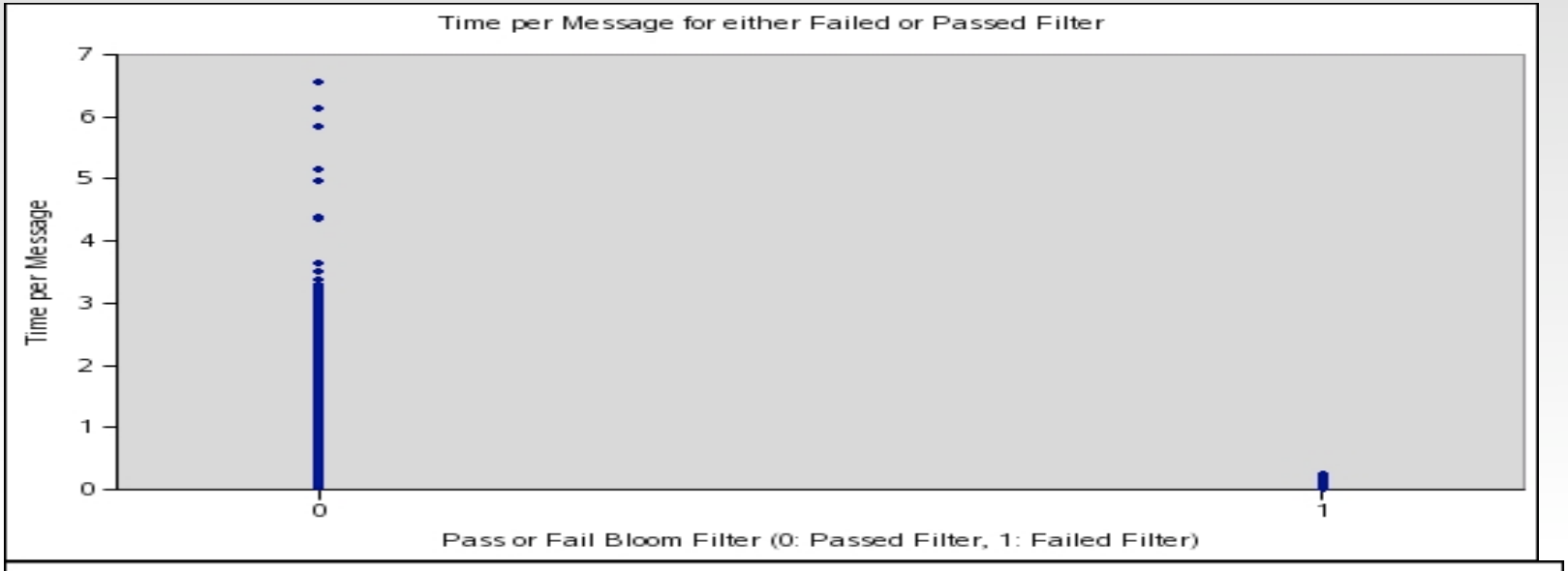
Query Production

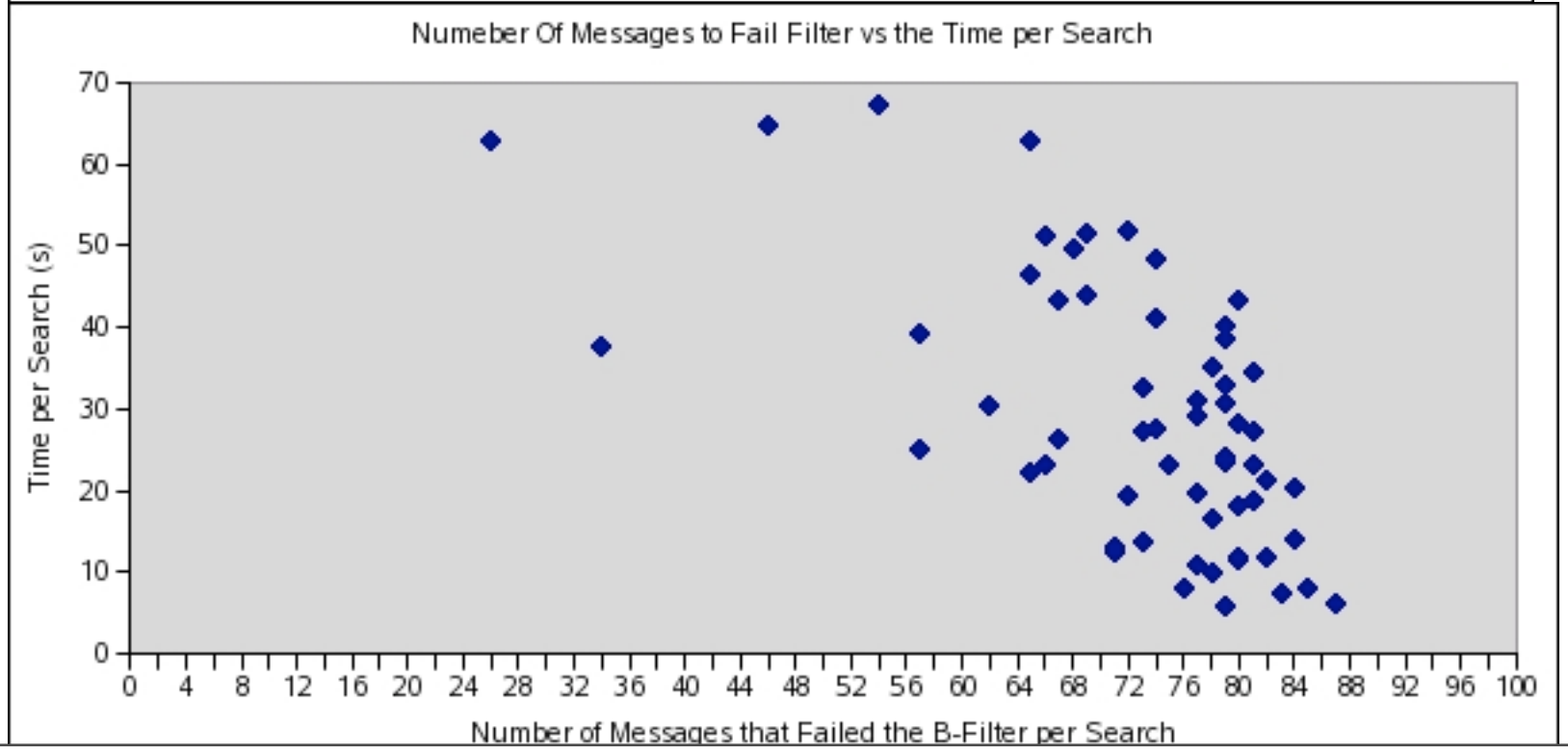
- Created queries with 1-20 keywords
- Three flavors
  - first match
  - last match
  - no match
- 2 sec to create for 20 keywords
- At most 9 kb for 20 keywords

Searching

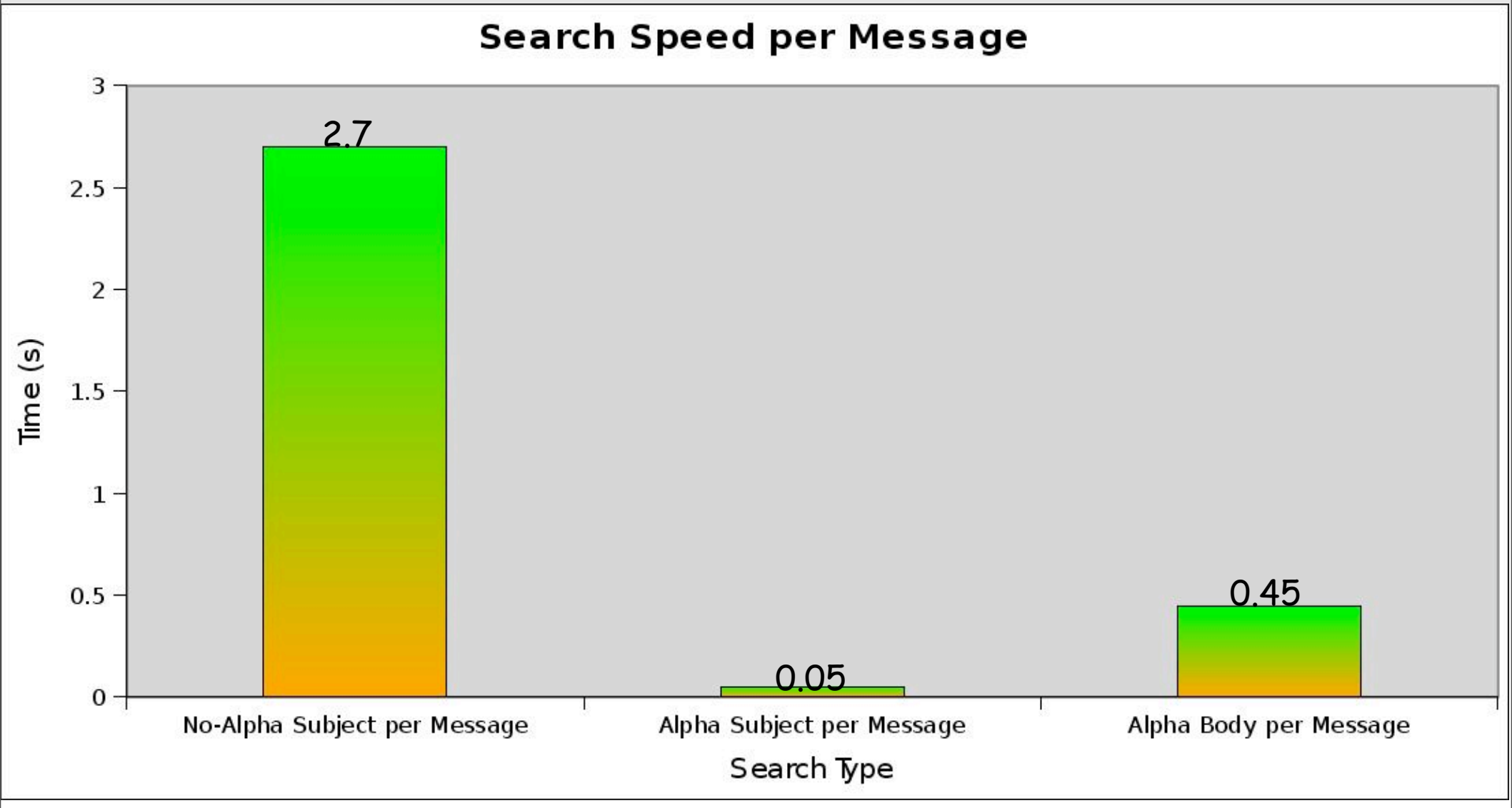
- Subject
  - with out Alpha-Sorting
  - with Alpha-Sorting
- Body with Alpha-Sorting

Effects of Error-Filter

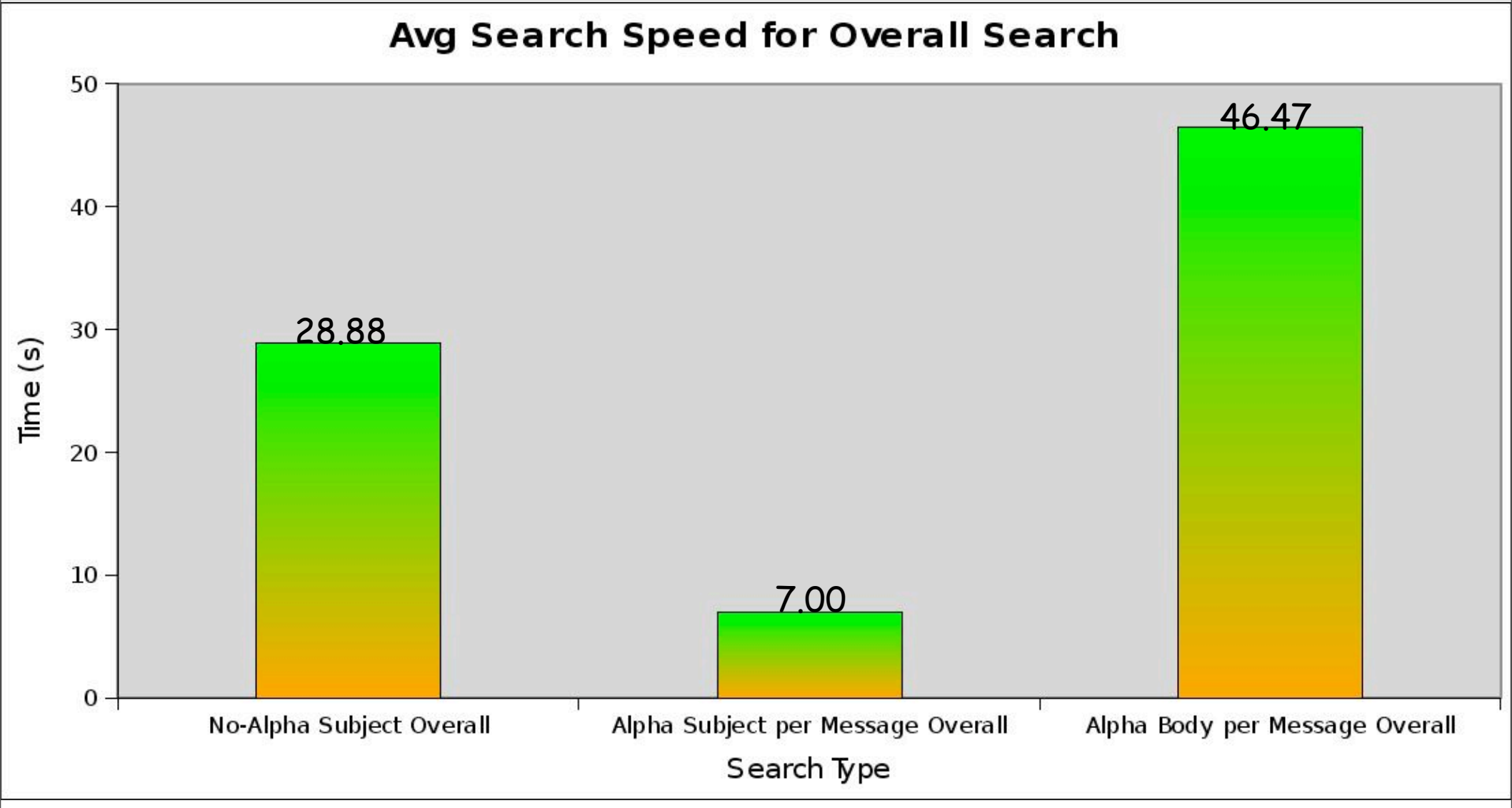




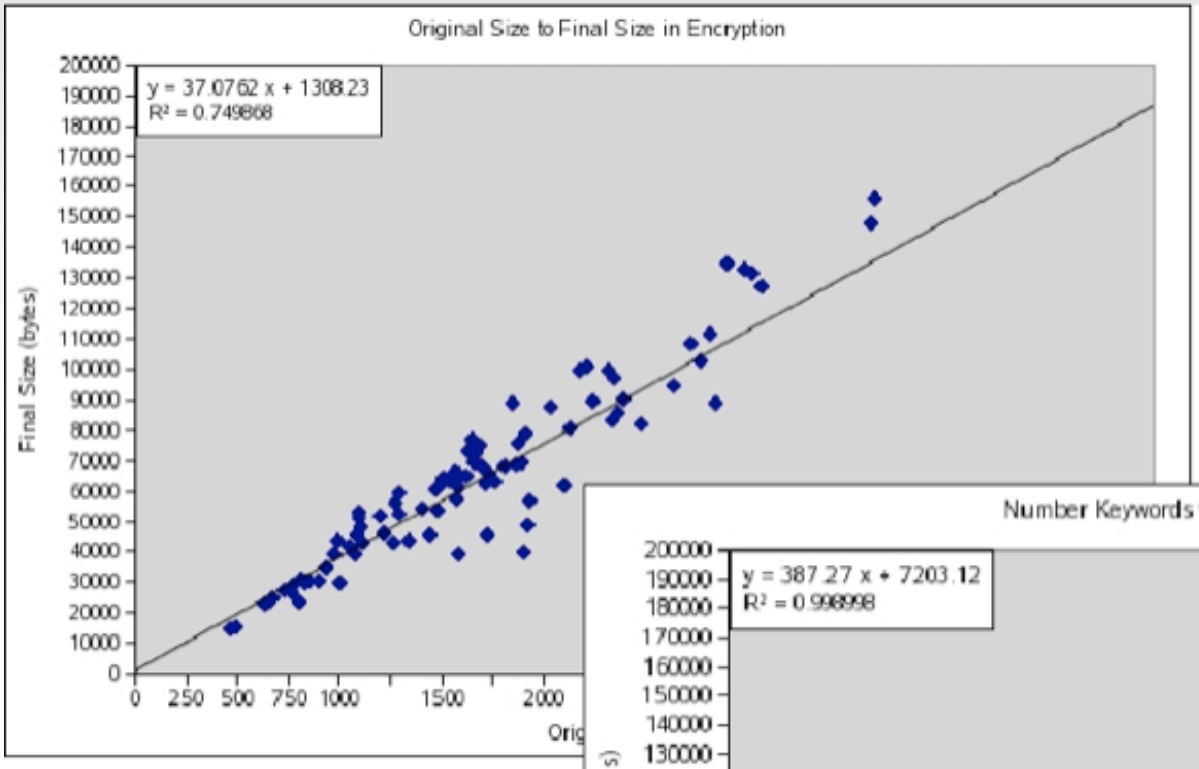
Search Speed Per Message

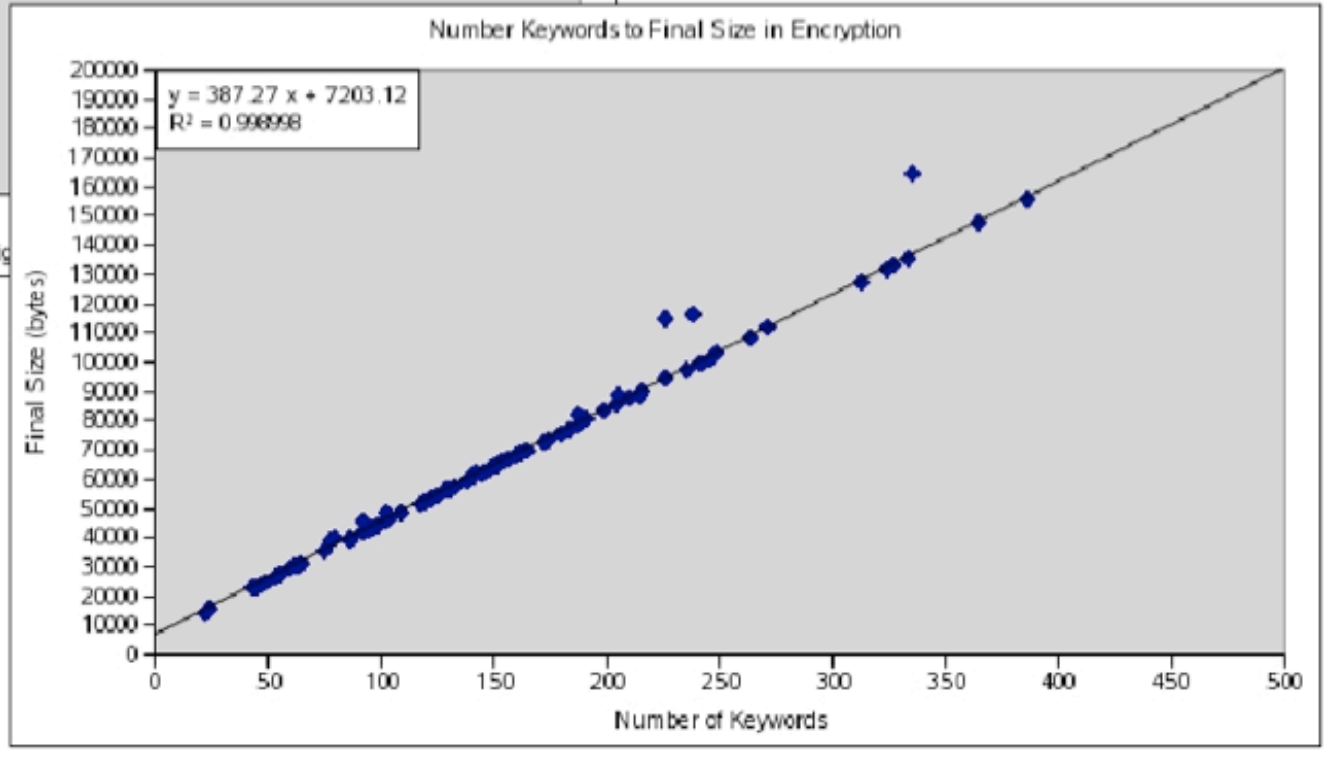


Search Speed Overall



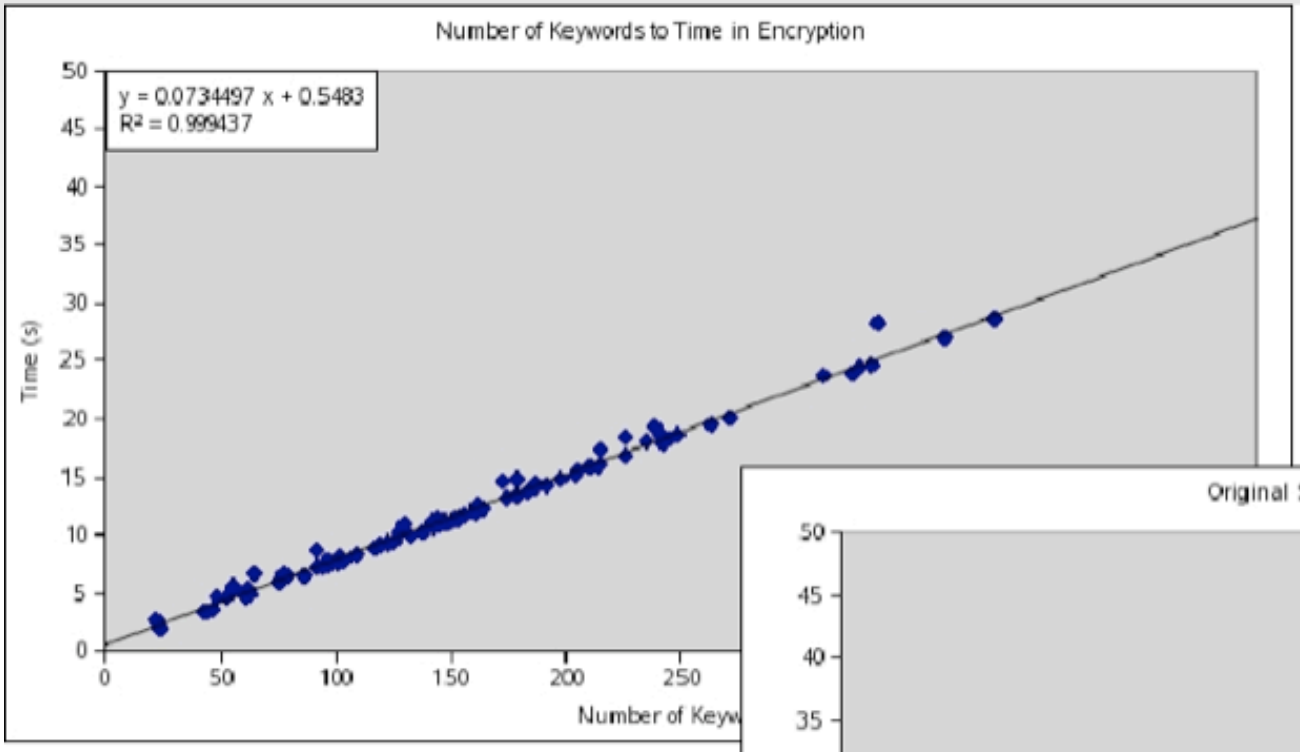
Message Size



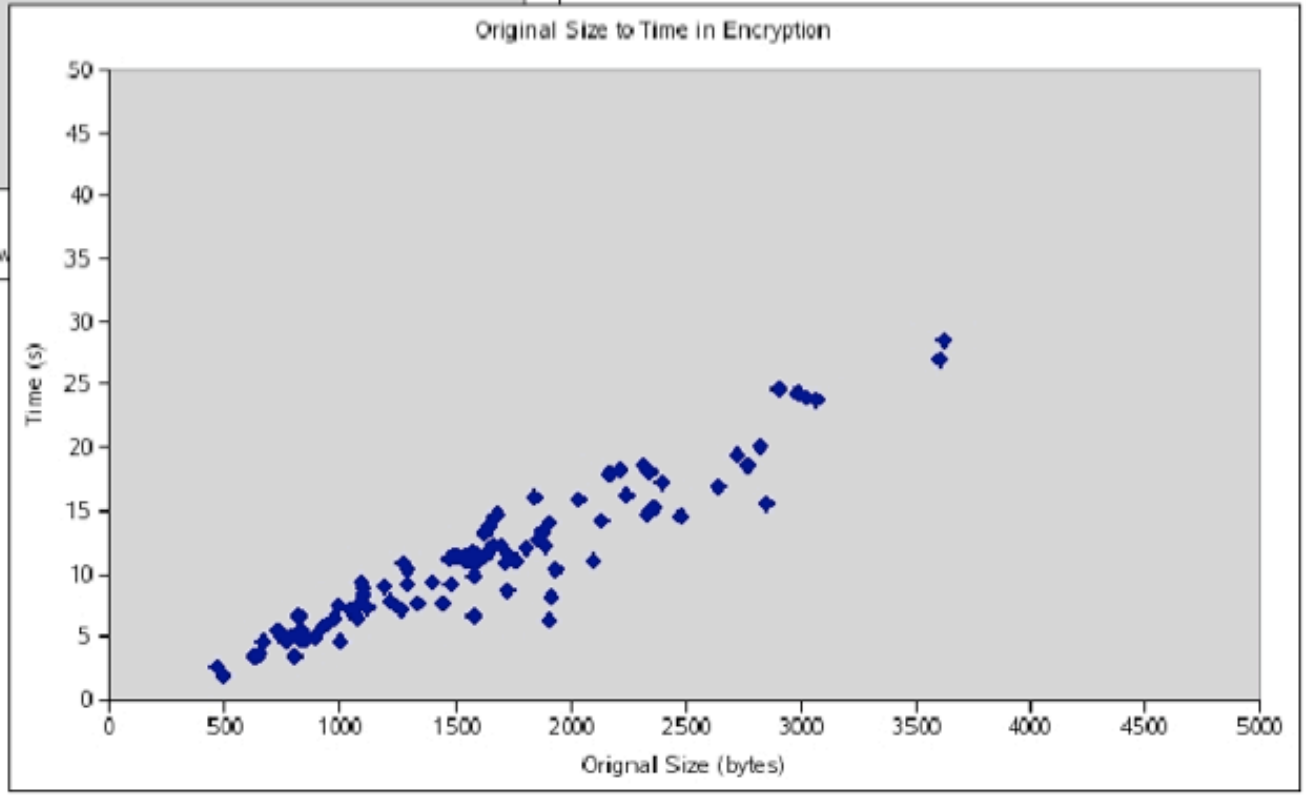


- 37x Increase Ratio
- Keywords rather then original size have largest effect

Message Production Time



- 17.17 s. average 24.17 std.
- Worst Case: 179.31 s.
- Keywords rather then original size again have largest effect



Conclusions

- We have presented SSARES and a preliminary implementation with an evaluation
  - no private information at server
  - protect "email at rest" and searching routine
- SSARES fits our goal of Autonomy and Transparency
- The system still needs improvement to be fully usable in a real working model

Future Work

- Secure NLP frequency analysis using the error-prone filters as indexes
  - select 15 most important words in body
- Use a similar error construction in query filters
- Implementation Improvements
- Launch a real working system