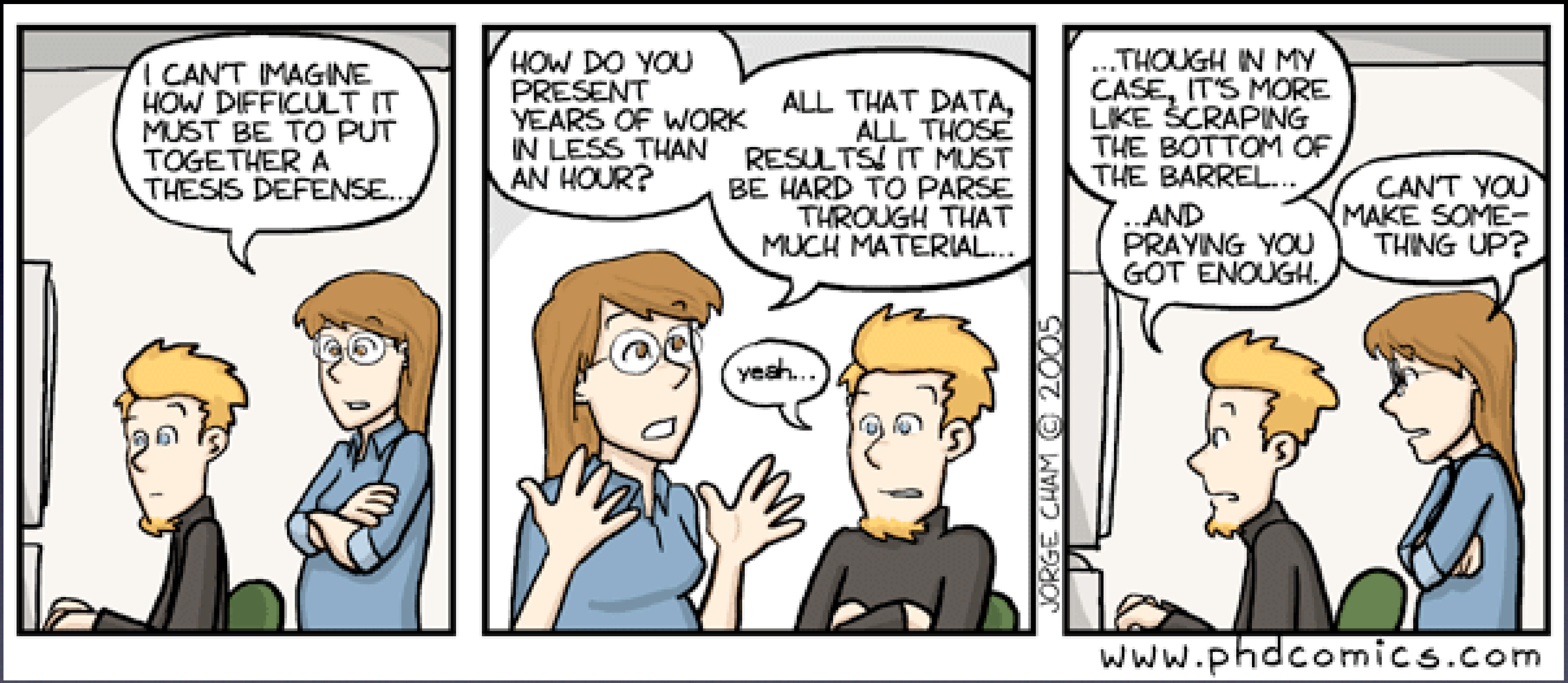
Privacy-Preserving Distributed Event Corroboration

Janak J. Parekh
Thesis Defense
March 23, 2007


I CAN'T IMAGINE HOW DIFFICULT IT MUST BE TO PUT TOGETHER A THESIS DEFENSE...
HOW DO YOU PRESENT YEARS OF WORK IN LESS THAN AN HOUR?
ALL THAT DATA, ALL THOSE RESULTS! IT MUST BE HARD TO PARSE THROUGH THAT MUCH MATERIAL...
yeah...
JORGE CHAM © 2005
...THOUGH IN MY CASE, IT'S MORE LIKE SCRAPING THE BOTTOM OF THE BARREL...
...AND PRAYING YOU GOT ENOUGH.
CAN'T YOU MAKE SOME-THING UP?
www.phdcomics.com

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

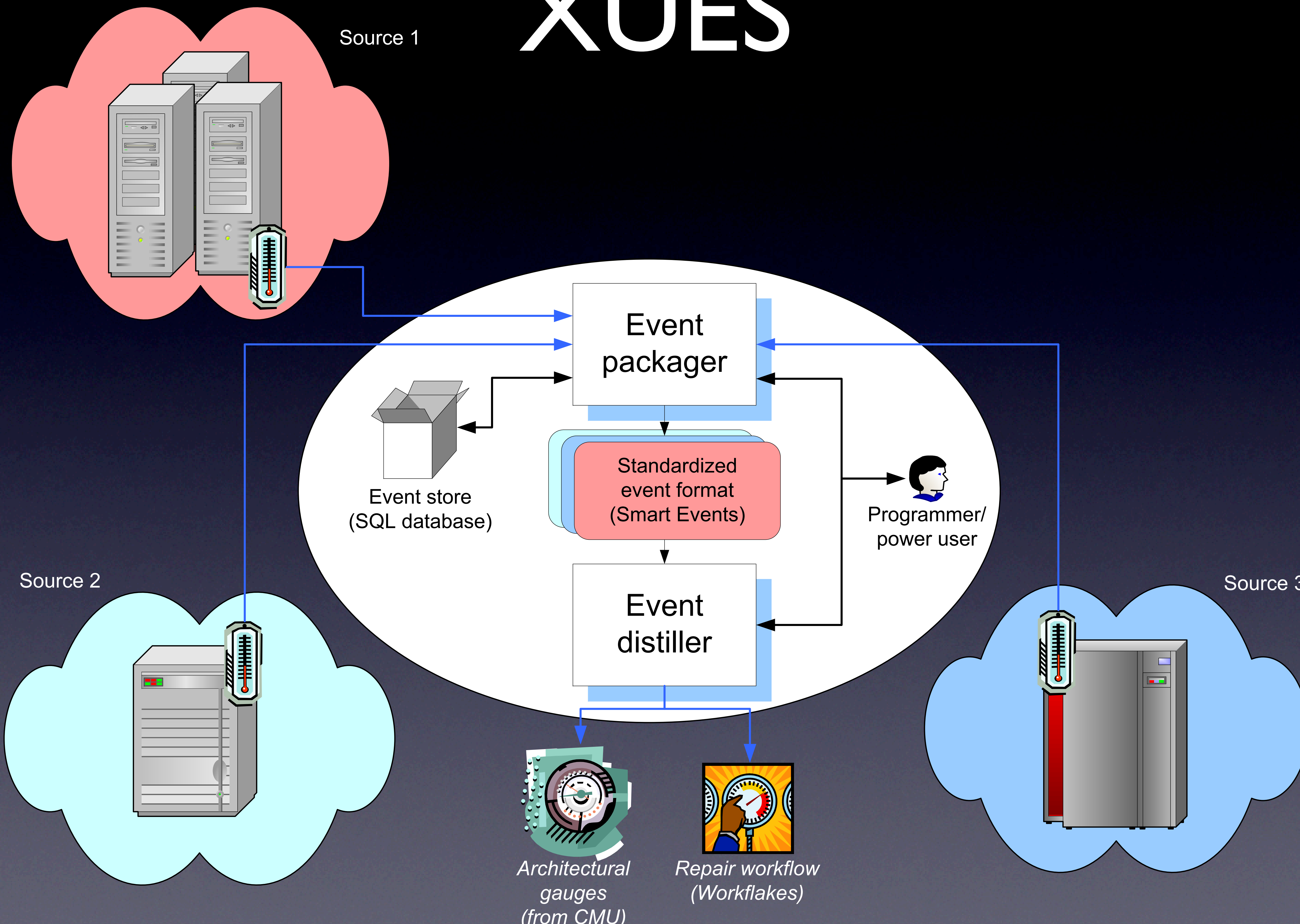
Motivation

- 1999-2003: Developed KX (Kinesthetics Extreme), a software monitoring and repair architecture
  - Software reliability → *autonomic computing*
  - Internet-scale, decentralized, *event-driven*
  - **Sensor** and **gauge** model
- XUES (XML Universal Event Service): temporal-driven event processor

KX

XUES

XUES


Source 1
Event packager
Event store
(SQL database)
Standardized event format
(Smart Events)
Programmer/
power user
Source 2
Event distiller
Source 3
Architectural gauges
(from CMU)
Repair workflow
(Workflakes)

XUES Applications

- *Service failure robustness*, *load balancing*: DARPA challenge problem to instrument, improve robustness of distributed GeoWorlds GIS/news visualization platform
- *QoS*: Internet-scale deployment in joint work with TILab, instrumenting instant-message platform
- *Spam detection* via temporal patterns

What about...

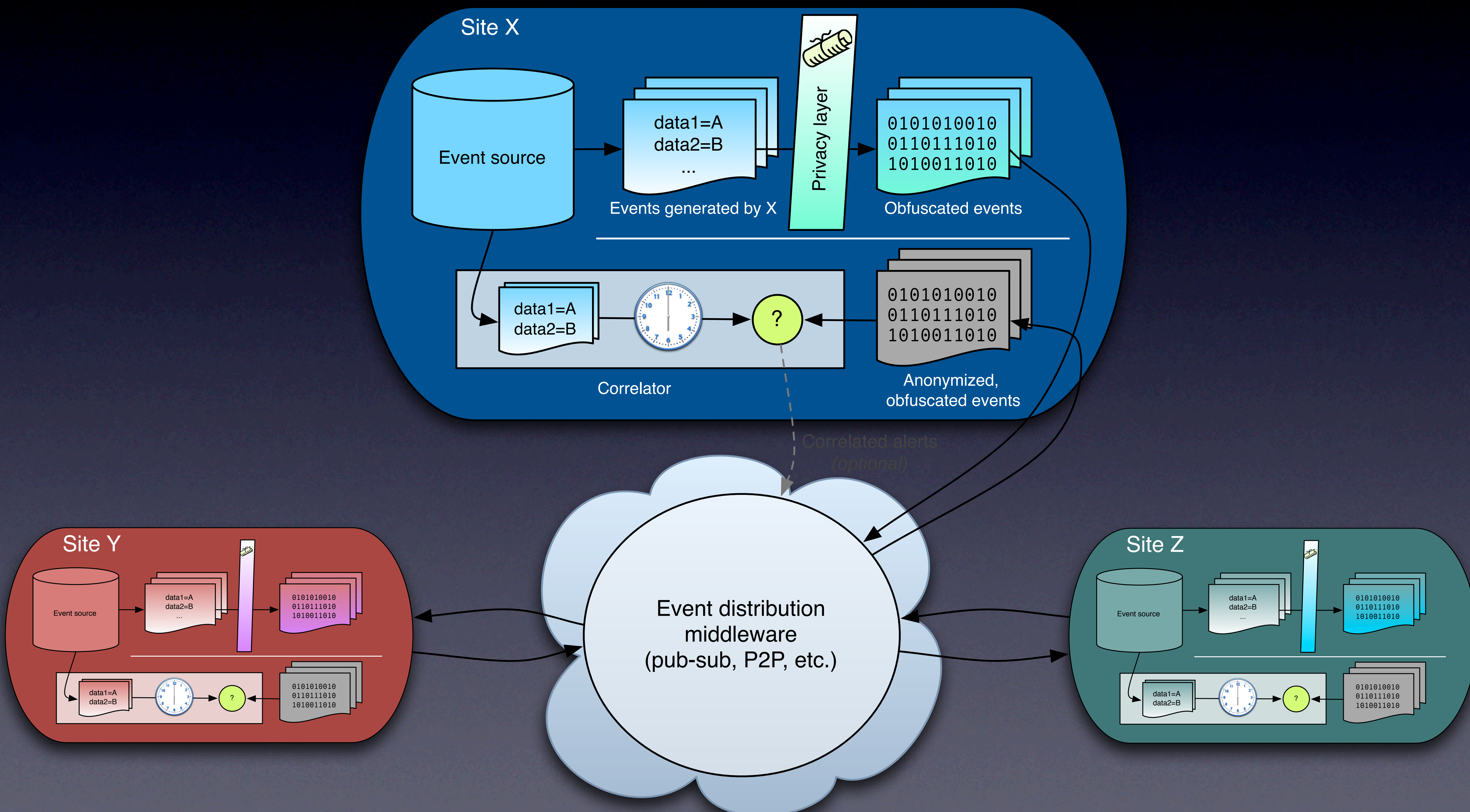
- Distributed service failure detection
  - *Distributed intrusion detection*
- P2P QoS
- Distributed spam detection

Each of the above require information disclosure, and are subject to privacy policies

Definitions

- *Events* are discrete, structured data objects generated at a specific point in time
    - Opaque, flat, and hierarchical
- *Privacy-preserving transformation* ***p(d)***, $d' = p(d)$, $d = p^{-1}(d')$ intractable
- *Privacy policy* is a promise by an organization to originators and consumers of data

XUES + Privacy



Problem Statement

Design an event processing methodology, appropriate event transformation techniques, and a distribution and corroboration architecture to process transformed events that:

- Supports Internet-scale collaboration;
- Approximates generalized event correlation for software reliability and network security;
- Enables information sharing between organizations whose privacy policies would ordinarily forbid such event-driven information exchange.

Requirements

- Event source anonymity and data privacy
  - Varying levels and types of data privacy depending on application
- Event *corroboration*
- Temporal constraints
- Near real-time performance, scalable to large-scale distributed systems

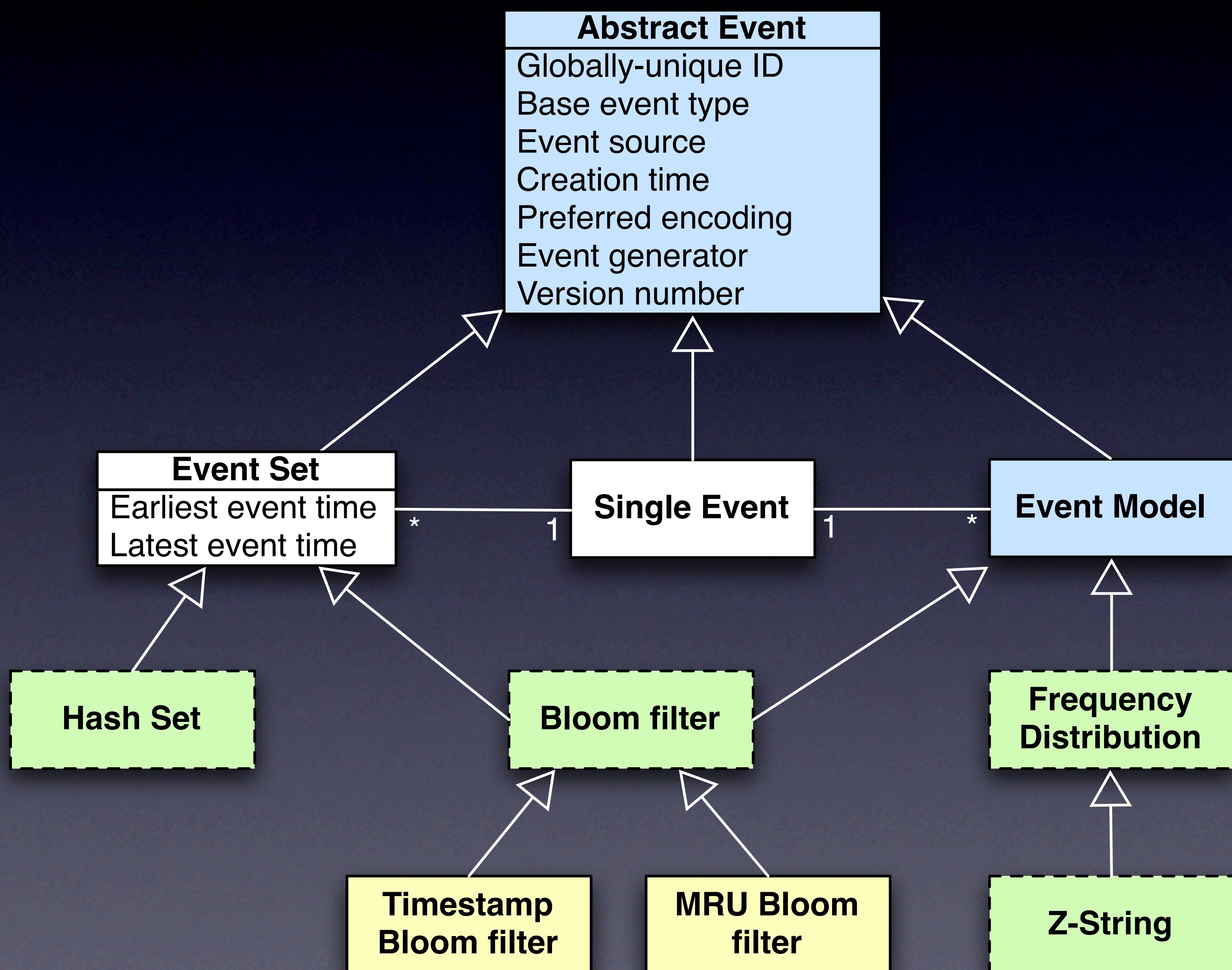
Hypotheses

- The addition of *one-way data transformations* will enable effective corroboration despite organizational privacy-preserving requirements
- A *typed event-driven framework* supporting a range of one-way (and two-way) data structures enables matching heterogeneous privacy-preservation requirements

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

Event Model



Abstract Event
Globally-unique ID
Base event type
Event source
Creation time
Preferred encoding
Event generator
Version number
Event Set
Earliest event time
Latest event time
*
1
Single Event
1
*
Event Model
Hash Set
Bloom filter
Frequency Distribution
Timestamp Bloom filter
MRU Bloom filter
Z-String

Corroboration Model

Given the non-privacy-preserving corroboration $C_A(\mathcal{E}_B) = \mathcal{E}_A \cap \mathcal{E}_B = \{e_{A_1}, e_{A_2}, \ldots, e_{A_n}\} \cap \{e_{B_1}, e_{B_2}, \ldots, e_{B_n}\}$, we can devise both a *privacy-preserving set* $\mathcal{E}'$

$$\mathcal{E}' = \mathcal{P}(\mathcal{E}) = \{p(e_1), p(e_2), \ldots, p(e_n)\}$$

and/or a *privacy-preserving model* $\mathcal{E}' = \mathcal{M}(\mathcal{E})$ with similarity metric $\mathcal{S}(e, \mathcal{E}') \rightarrow [0, 1]$.

Corroboration thus becomes:

$$C'_A(\mathcal{E}'_B) = \begin{cases} \{e_{A_i} \mid p(e_{A_i}) \in \mathcal{E}'_B\} & : \quad \mathcal{E}' \text{ is a set} \\ \{e_{A_i} \mid \mathcal{S}(e_{A_i}, \mathcal{E}') > \tau\} & : \quad \mathcal{E}' \text{ is a model} \end{cases}$$

Infrastructure Model

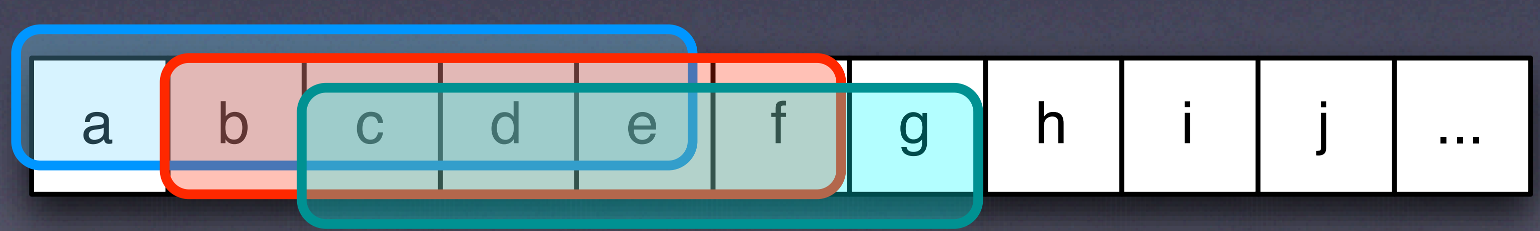
- *Provide* event middleware consisting of:
  - Type modules
  - Transform modules
  - Corroboration modules
- *Utilize* event distribution infrastructure capable of:
  - Anonymity (<u>up to publisher</u>)
  - Typing
  - Ordering/Timestamping (for constraints)
  - End-to-end Encryption

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

Data Privacy

- Goal: transform data before its publication in a form allowing corroboration
  - *Insert* and *verify* → one-way data structure; whole-entity matching
  - *Incremental analysis*, via feature extraction or N-grams, to allow partial matching in a one-way data structure



  - *Model comparison/combination*

Techniques Used

Technique	Applicability	Computation overhead	Space overhead	Privacy gain	Temporal corroboration
Hashing	General	Low	Large	Medium	Easy
Bloom filters	General	Low	Medium	Very good	Medium
Frequency transforms	Opaque/ non-feature-oriented	Very low	Large	Excellent	Hard
Z-Strings	Based on frequency transforms	Low	Small	Excellent	Easy

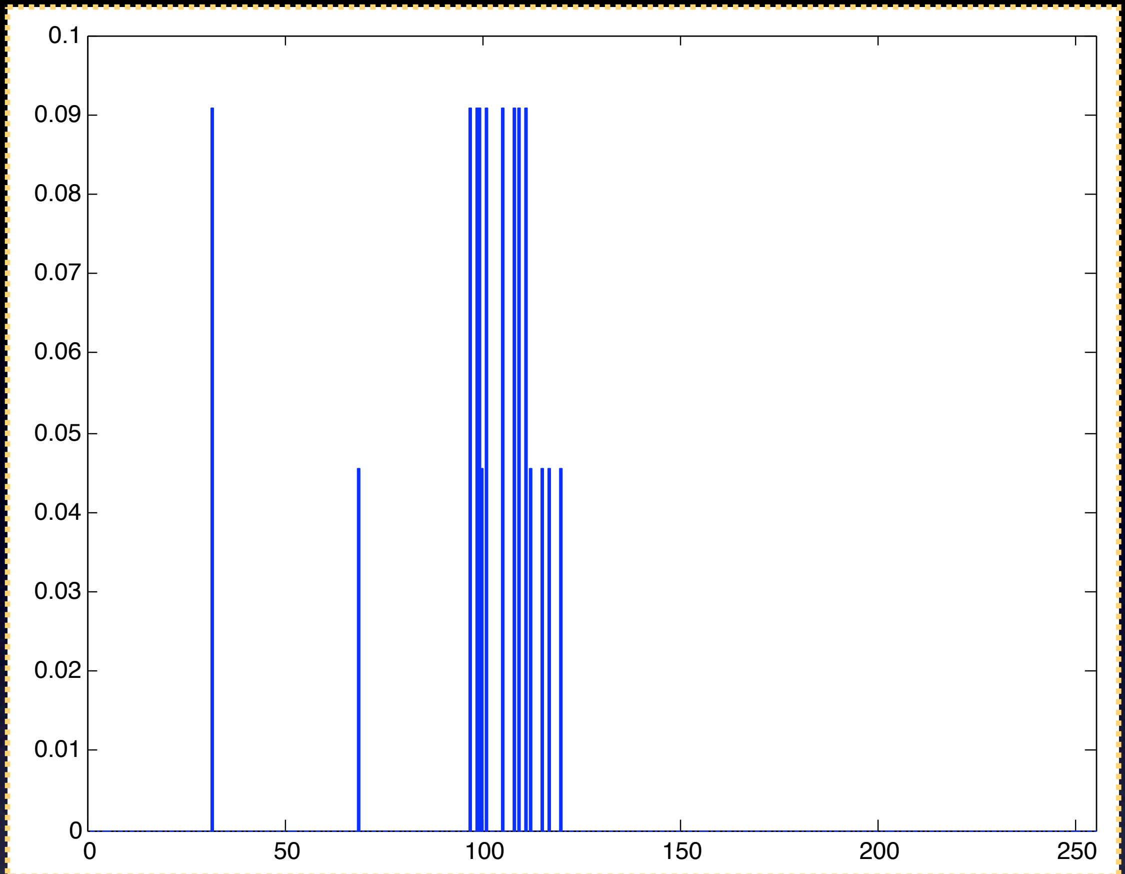
Examples

Example malicious code

Original content: 176 bits.

Exa, xam, amp, mpl, ple,
le□, e□m, □ma, mal, ali,
lic, ici, cio, iou, ous,
us□, s□c, □co, cod, ode

List of (unique) 3-grams in original string. A box represents a space; the underlined n-gram appears twice in the original alert. 20 n-grams take approximately 480 bits.



Frequency distribution; the most frequent character is a space (ASCII code 32). Size ≈ 8160 bits.

□aceilmoEdpsux

Z-String; the space (box) is the most frequent character. Non-appearing characters are removed. 15 characters = 120 bits.

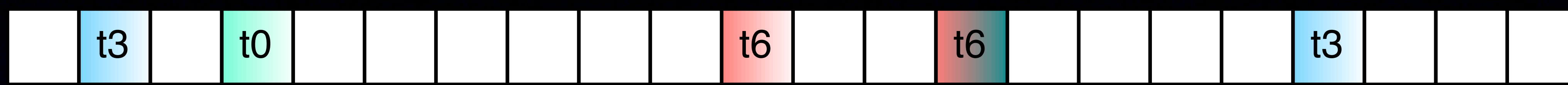
00000110101011010011011001101011010100...01010011101010101111000

Bloom filter of above n-grams. If three hash values are used, a minimum optimal size would be ~ 120 bits.

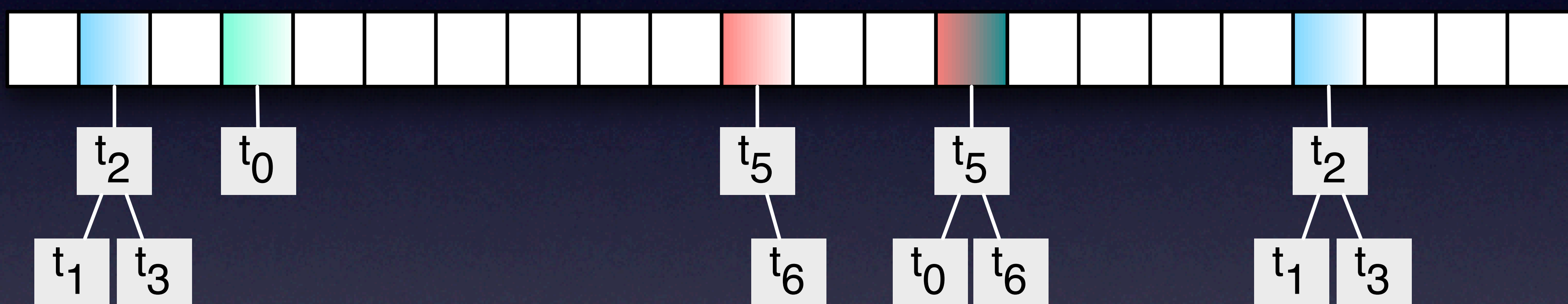
Temporal Corroboration

- How to corroborate against privacy-preserving *models* of events?
  - Linear search through all models → slow
  - Merge all models → saturation
  - *Merge & expire models* → *no range queries*
  - *Timestamp tree indices* → *for discrete models*
  - *Temporal clustering* → *general, but slower*

Temporal BFs



MRU Bloom filter (MRU BF)



Timestamp Bloom filter (TSBF)

- *Merge*, *expiry*, and (TSBF) *range lookups*
- Cost: memory overhead, lookup time (if saturated), privacy

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

Collaborative Intrusion Detection

- Increasing patterns of widespread scanning behavior across the Internet
- Existing COTS alerts have limited, single-site perspective and either are too noisy or miss slow/stealthy scans
- Goal: share intrusion alerts to gain global view on network threats

Hypotheses

A privacy-preserving architecture enables:

1. Participation of a broad group of contributors to detect slow scans/traffic patterns needed to build defenses;

2. Ability of contributors to exchange *vulnerability-specific* information for signature generation;

3. The ability of ad-hoc communication participants to determine each other's communication profiles, and <u>develop a trust model to determine exchange</u>

Worminator overview

- Rewrite of XUES platform with privacy type support
- Processes IDS sensor alerts and applies privacy transforms
- Fully modular, supports heterogeneous data types, sensors, communication networks
- Near real-time event processing and corroboration

010XXX011XX1010100110011101010XX110100001
WORMINATOR
COLLABORATIVE INTRUSION DETECTION

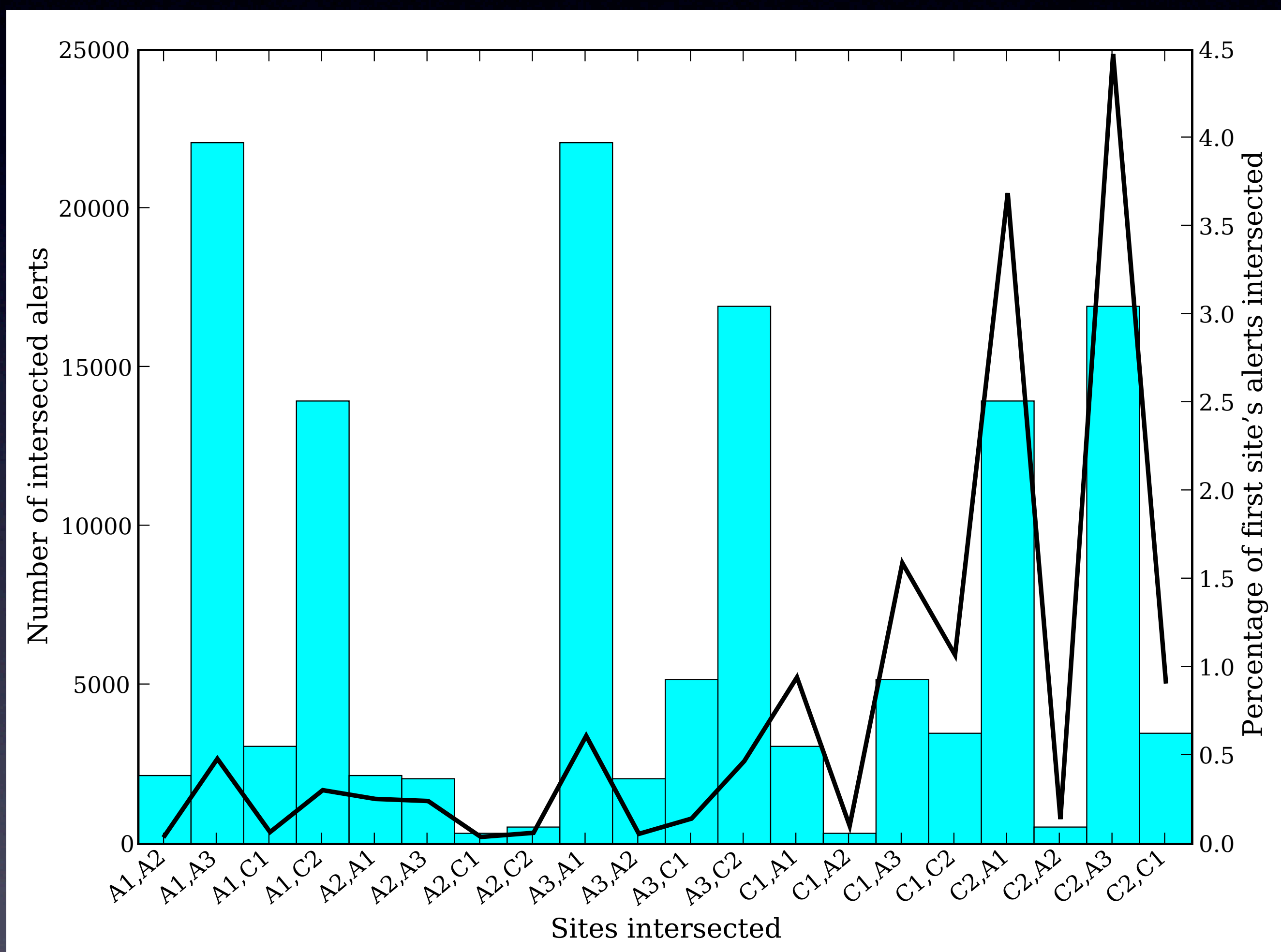
Implementation

- Written in about 20,000 lines of Java and Python code
- Performance tests using JDK 1.5 on dual-Xeon 3GHz with 4GB RAM
- IP-based alert exchange deployed at 3 commercial and 2 academic sites; <u>collected ~ 9 million alerts</u>
- Pluggable to support different sensor types; used Antura (misuse), PAYL and Anagram (in-house anomaly, 1-gram freq and n-gram BF, respectively)

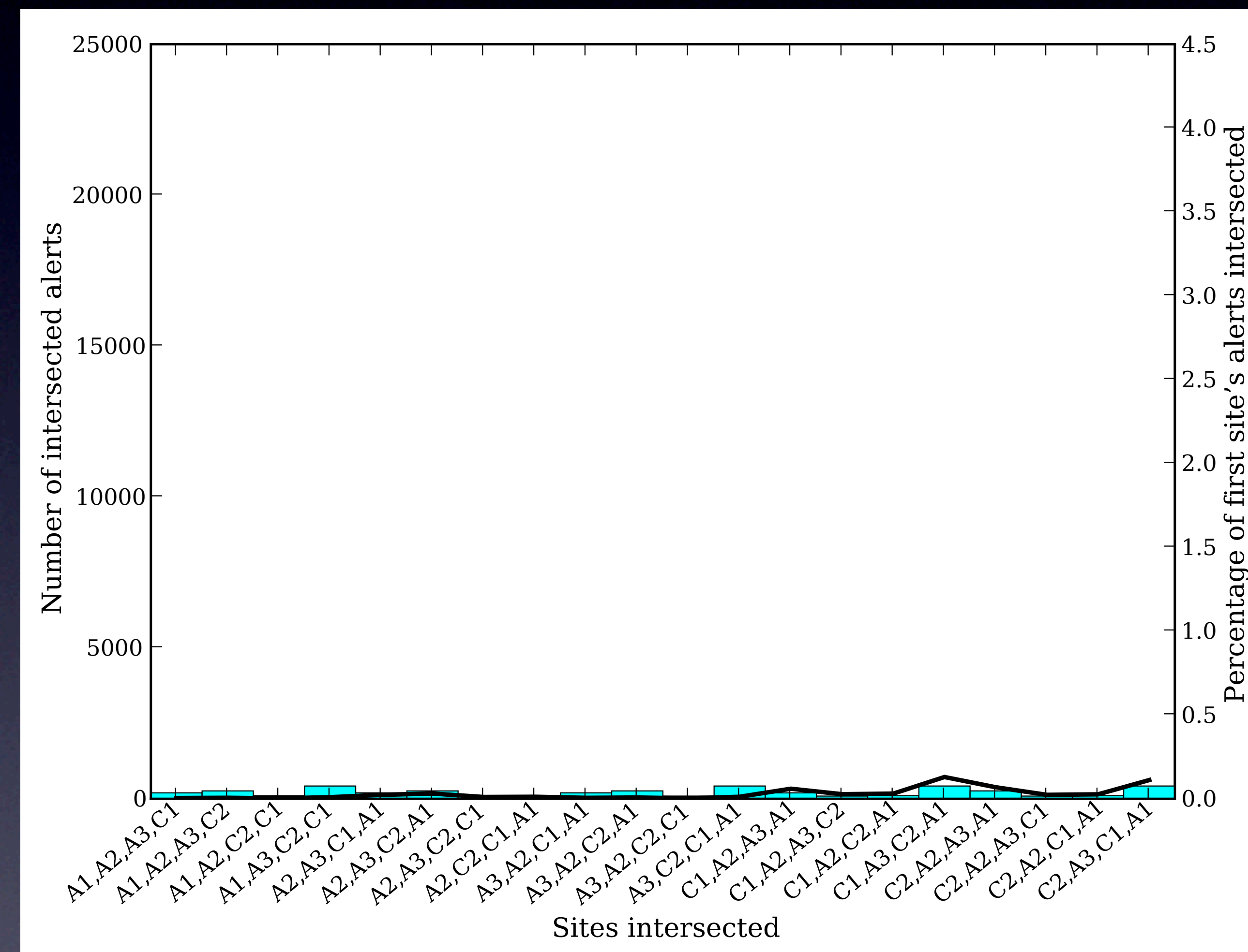
IP-based corroboration

- Key questions:
  - Is it *useful*?
  - Can corroboration be done *quickly*?
  - Can it be done *accurately*?
  - Does it *preserve* privacy?
- Techniques used
  - Hash functions
  - Bloom filters

Alert intersection, IP/port



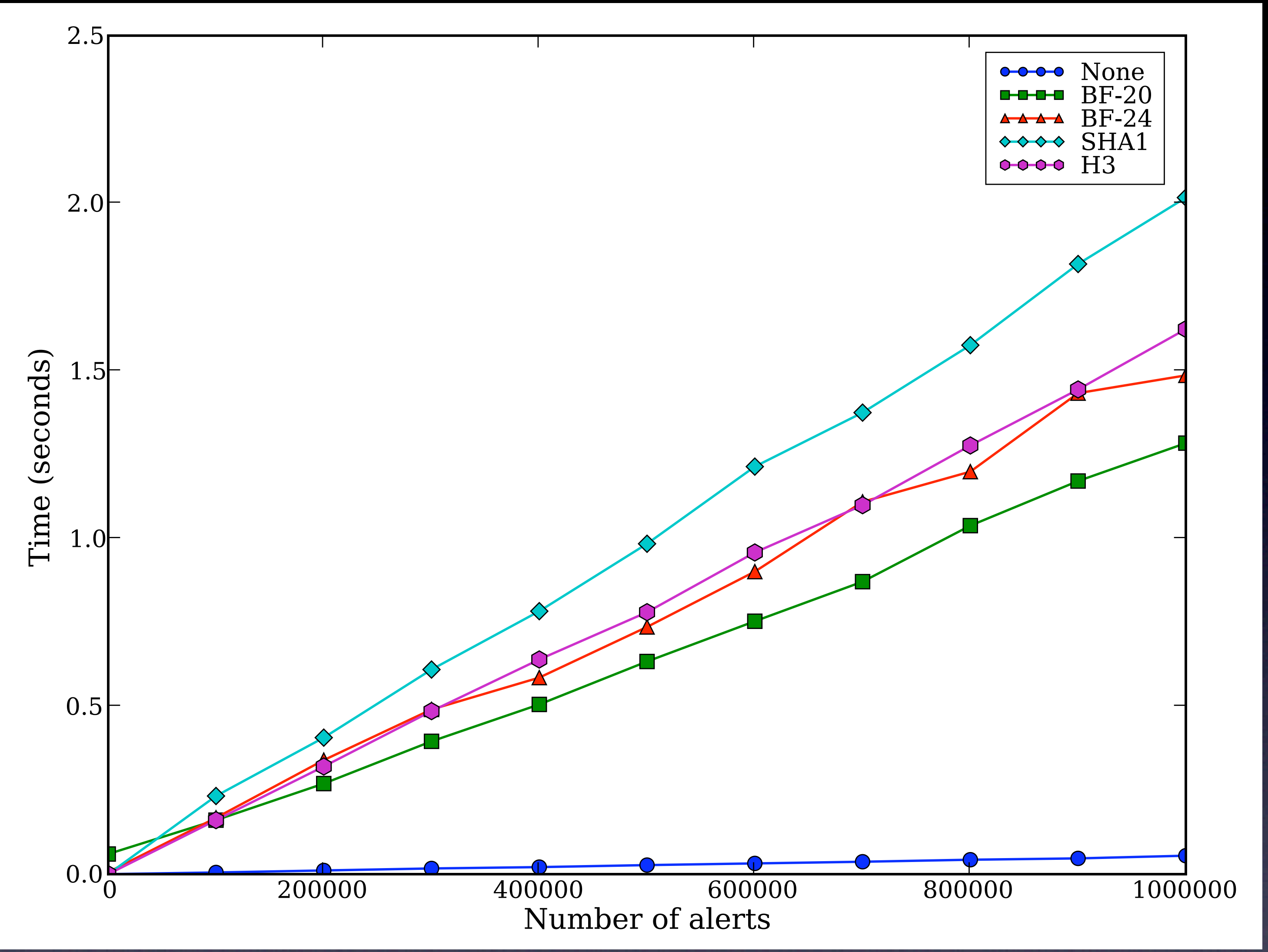
2-way corroboration



4-way corroboration

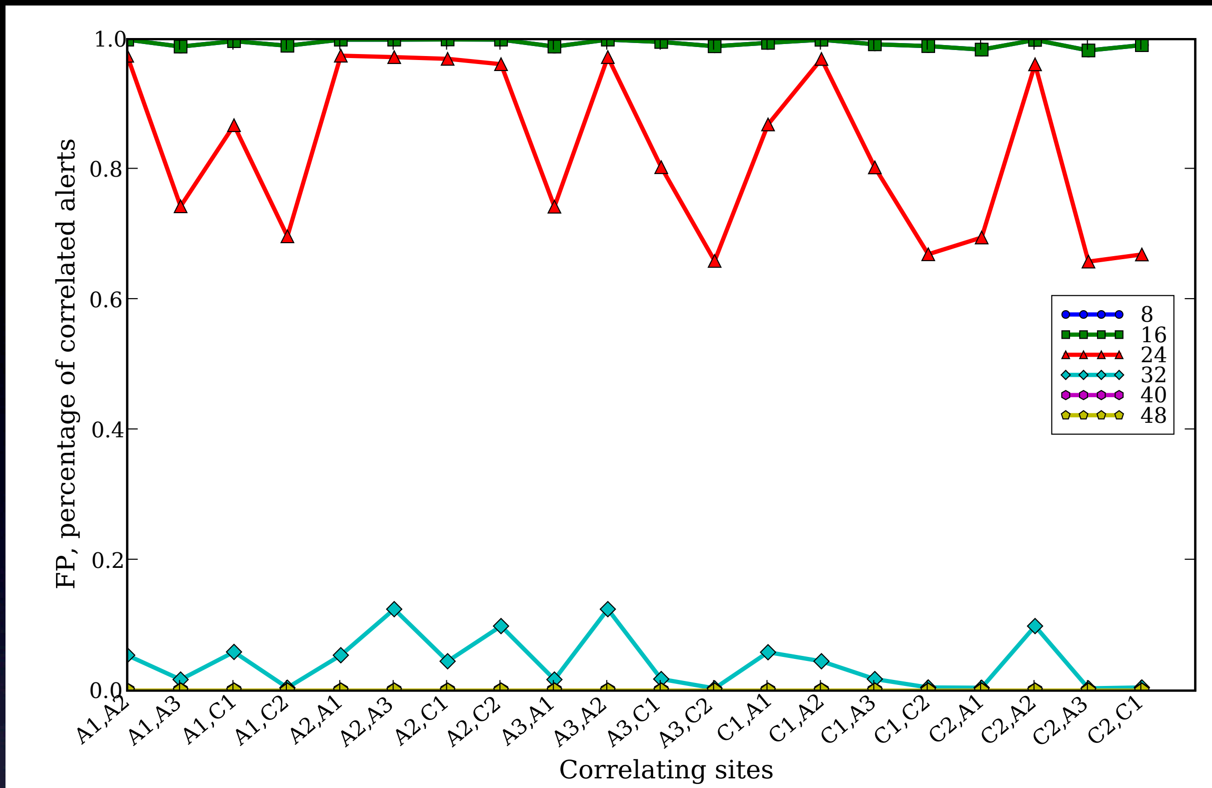
Overheads, IP/port

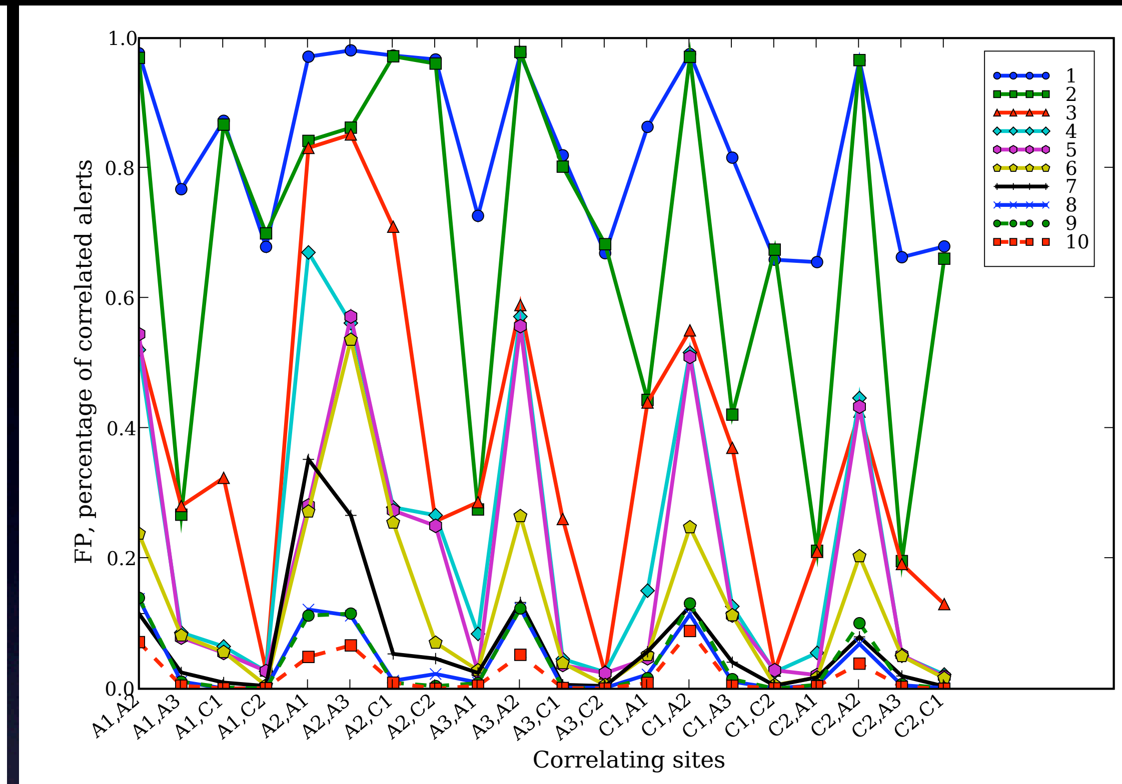
- Techniques all scale well computationally
- Hash functions usually use a fixed number of bits per alert, e.g., *160n*
- Bloom filter memory use is significantly less



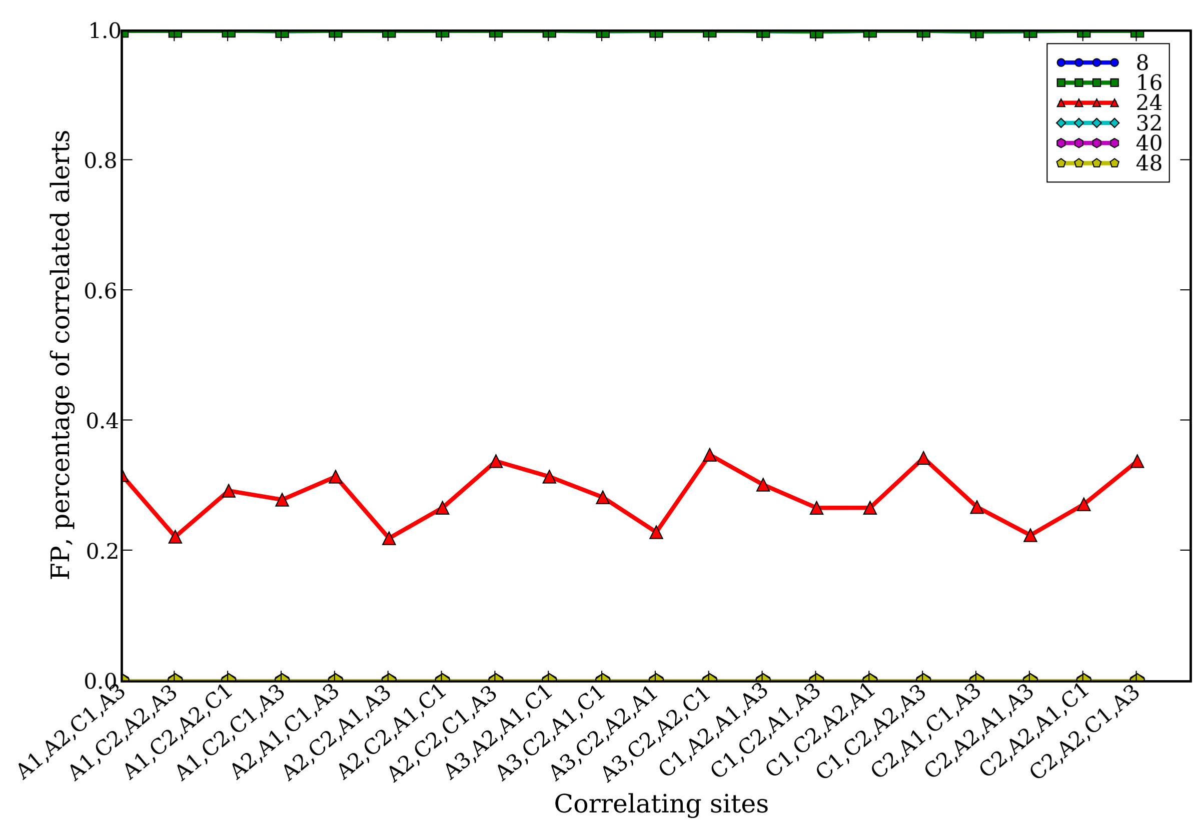
# entries	# hash functions	Uncompressed # bits		Compressed # bits	
		Size	Per Alert	Size	Per Alert
1	5	131072	131072	182	182
2	10	131072	65536	212	106
100000	5	131072	1.31	96361	.96

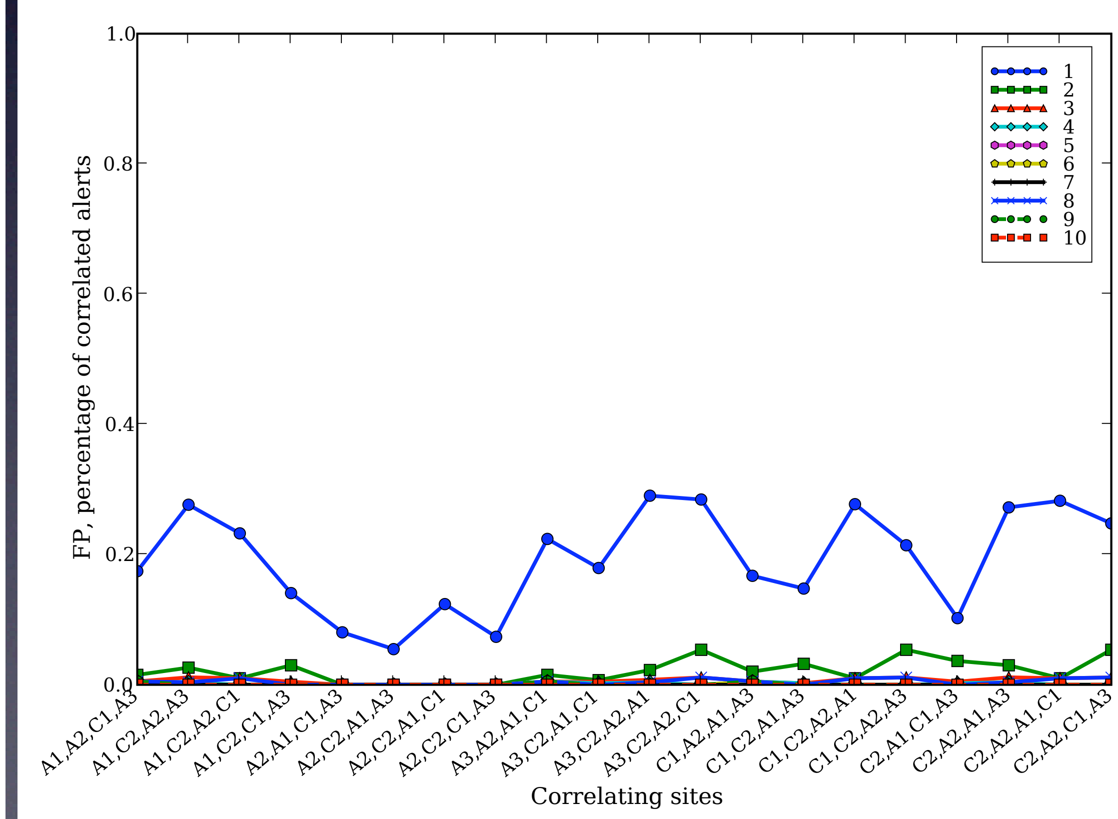
Corroboration FP, IP/port





2-way



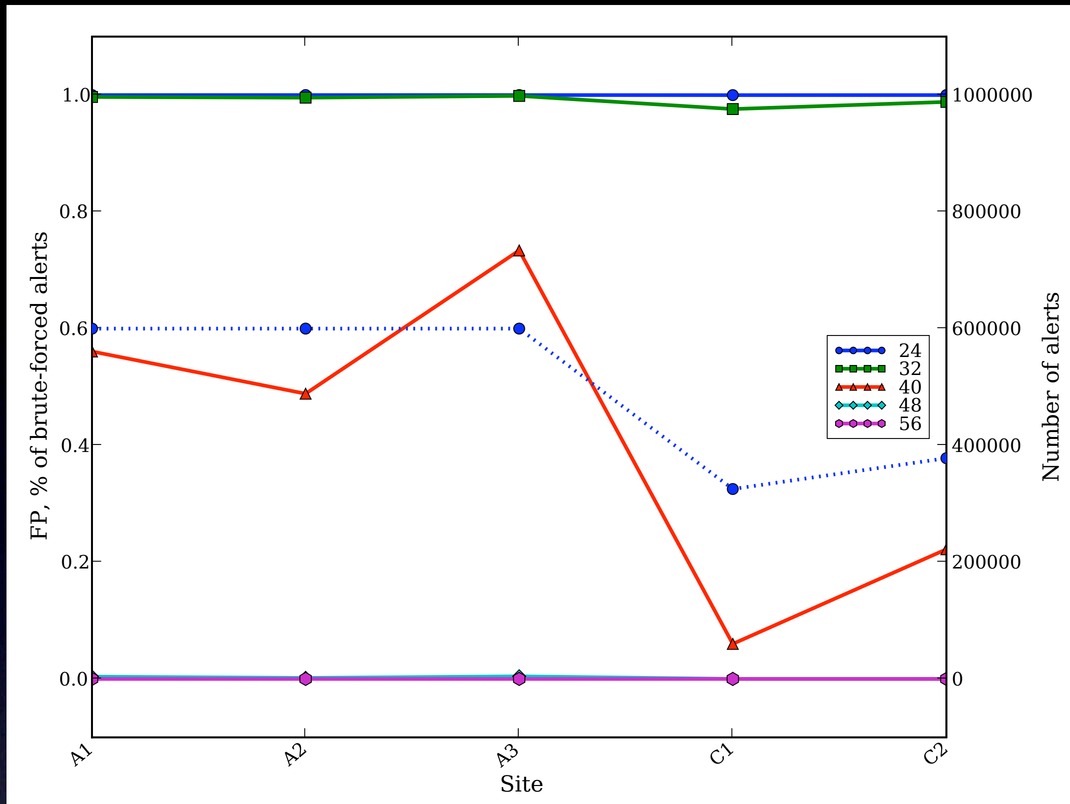


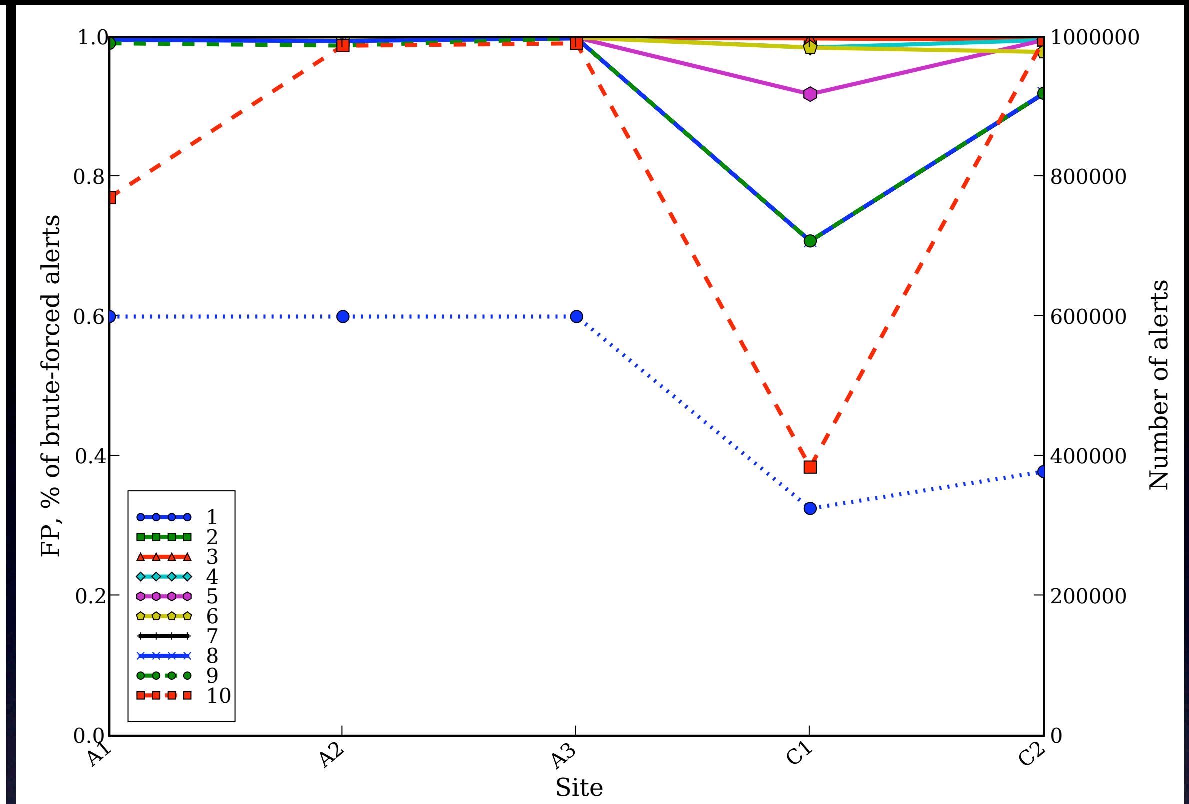
4-way

Hash set (H_3)

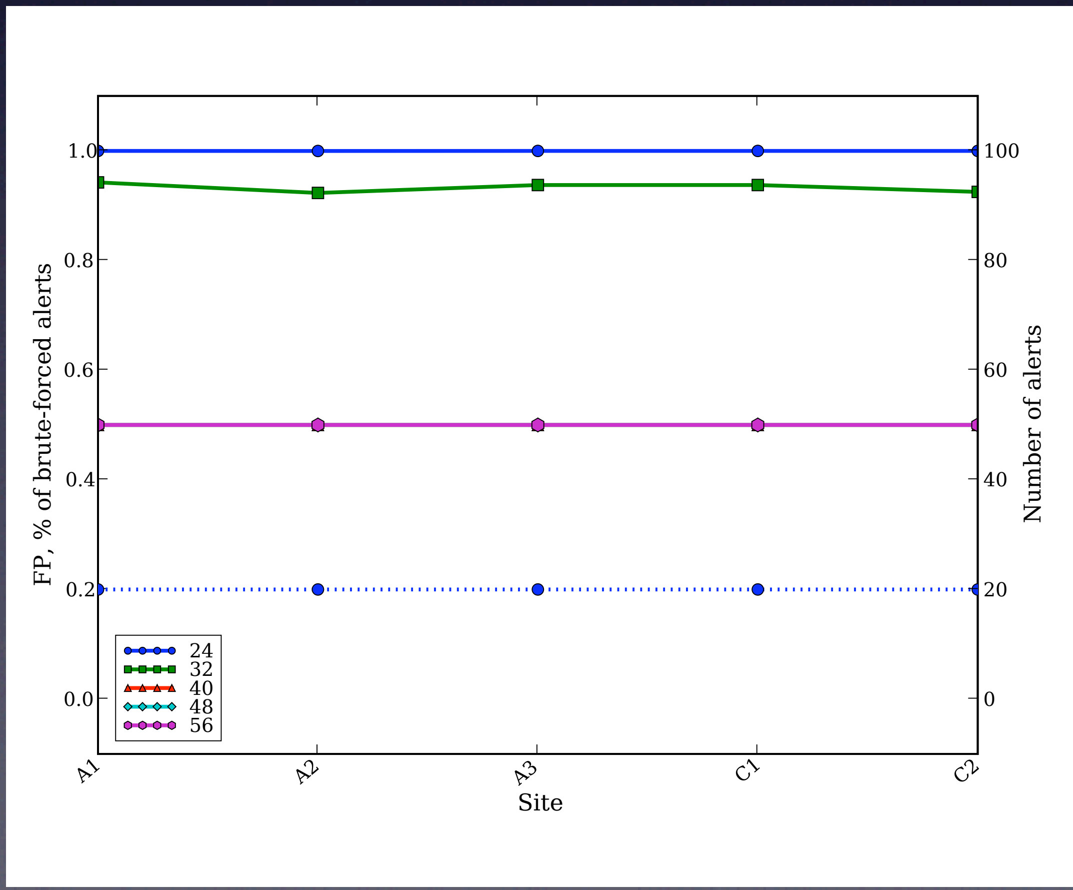
Bloom filter

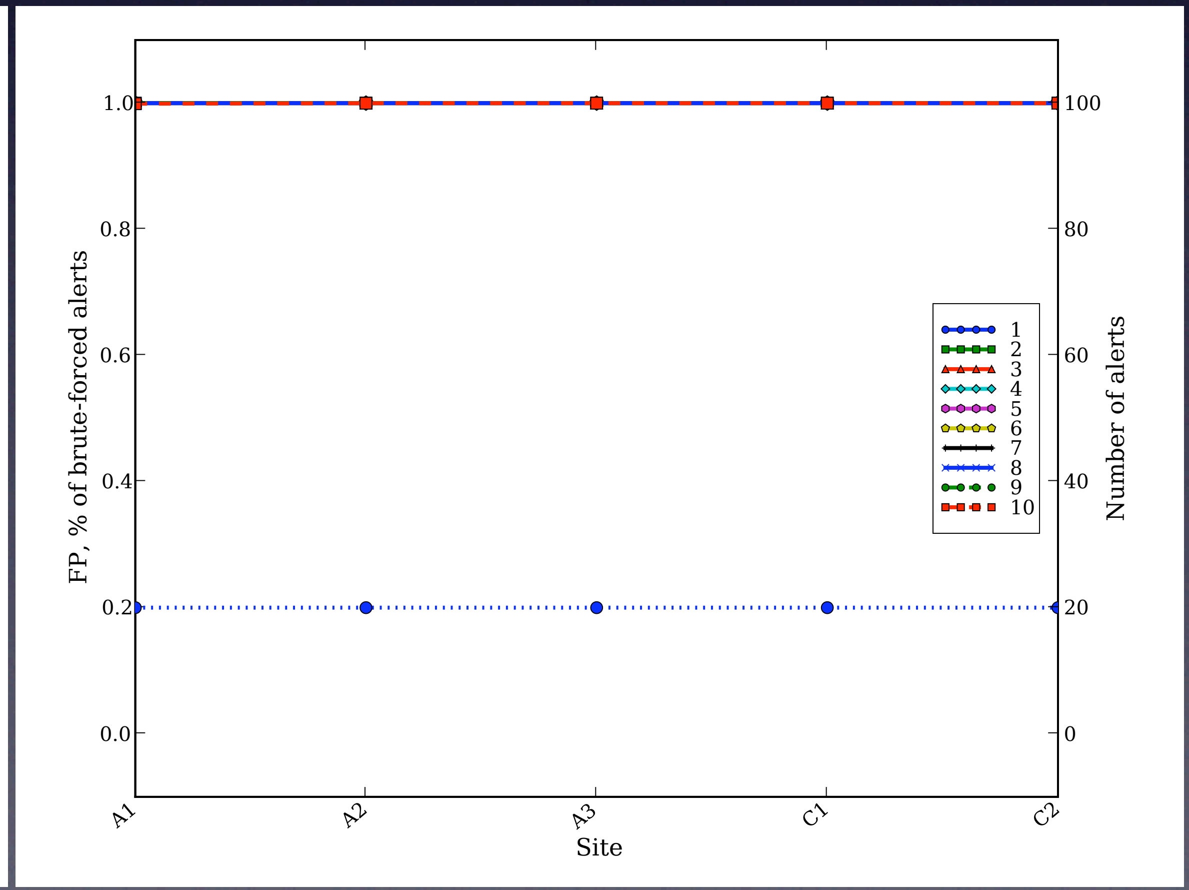
Brute-force FP, IP/port





~ 600k alerts



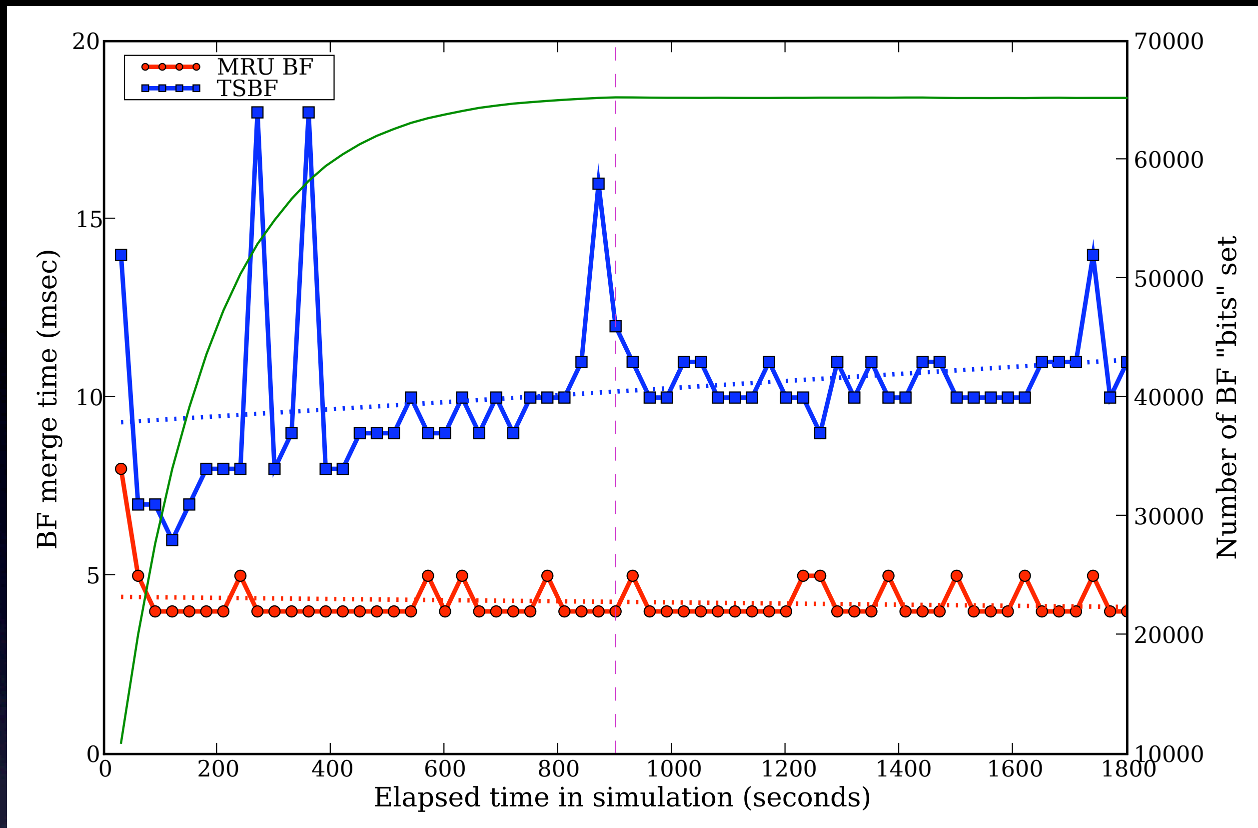


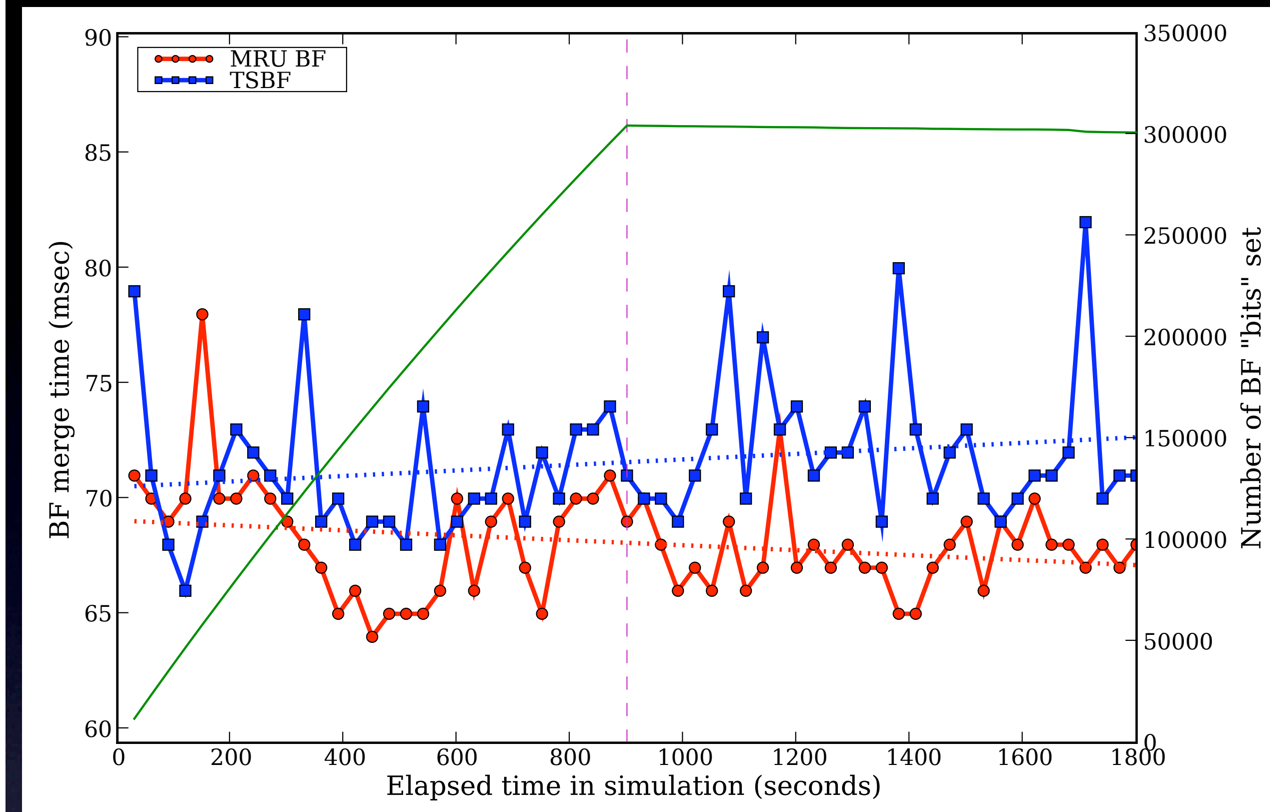
~ 20 alerts *(sparse, <u>noisy</u>)*

Hash set

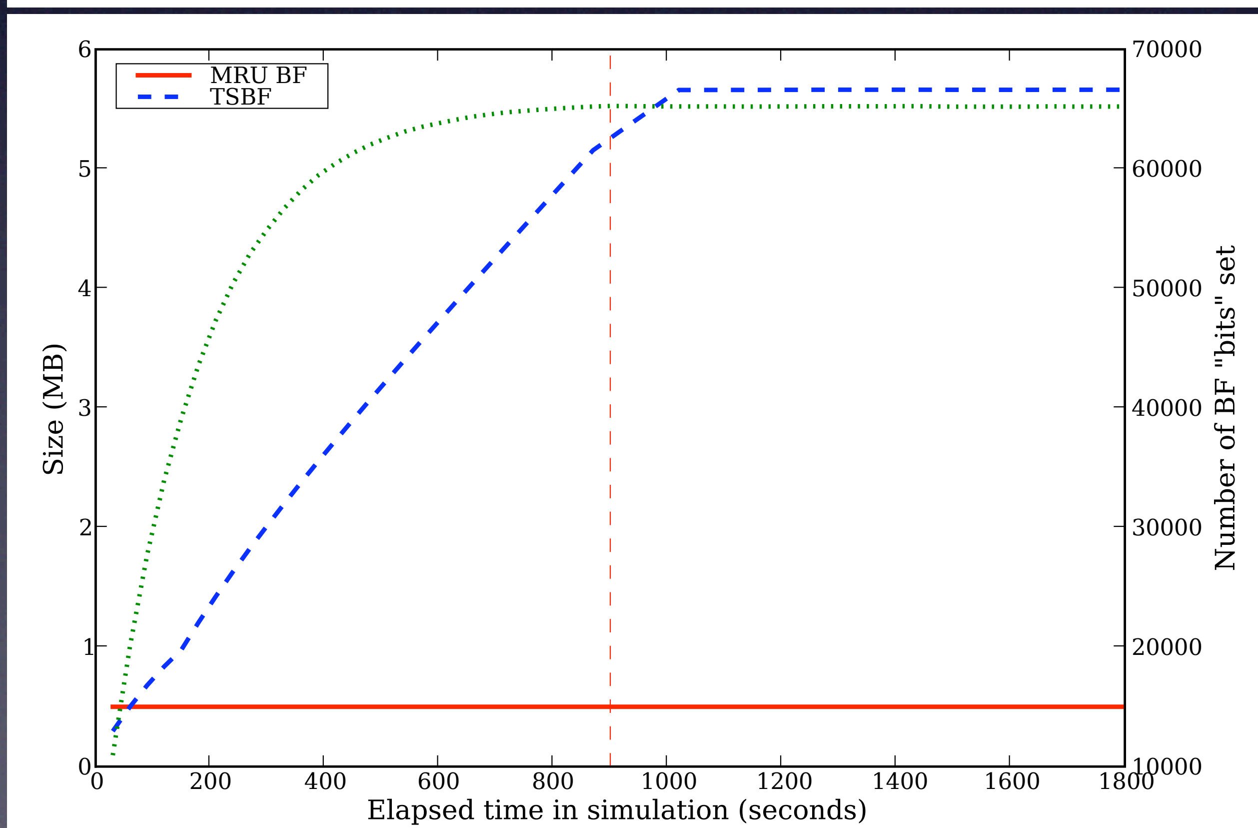
Bloom filter

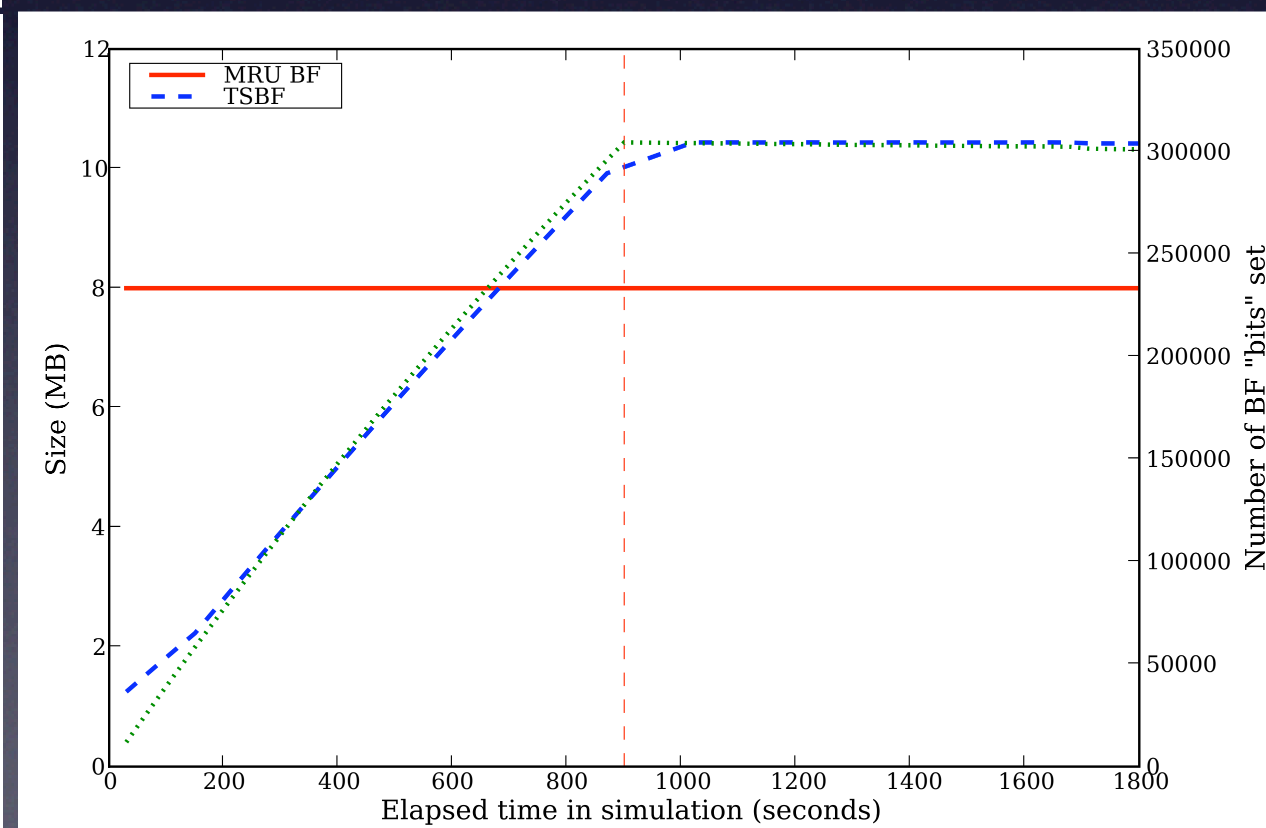
Temporal corroboration, IP/port





Merge time





Space usage

16-bit BFs

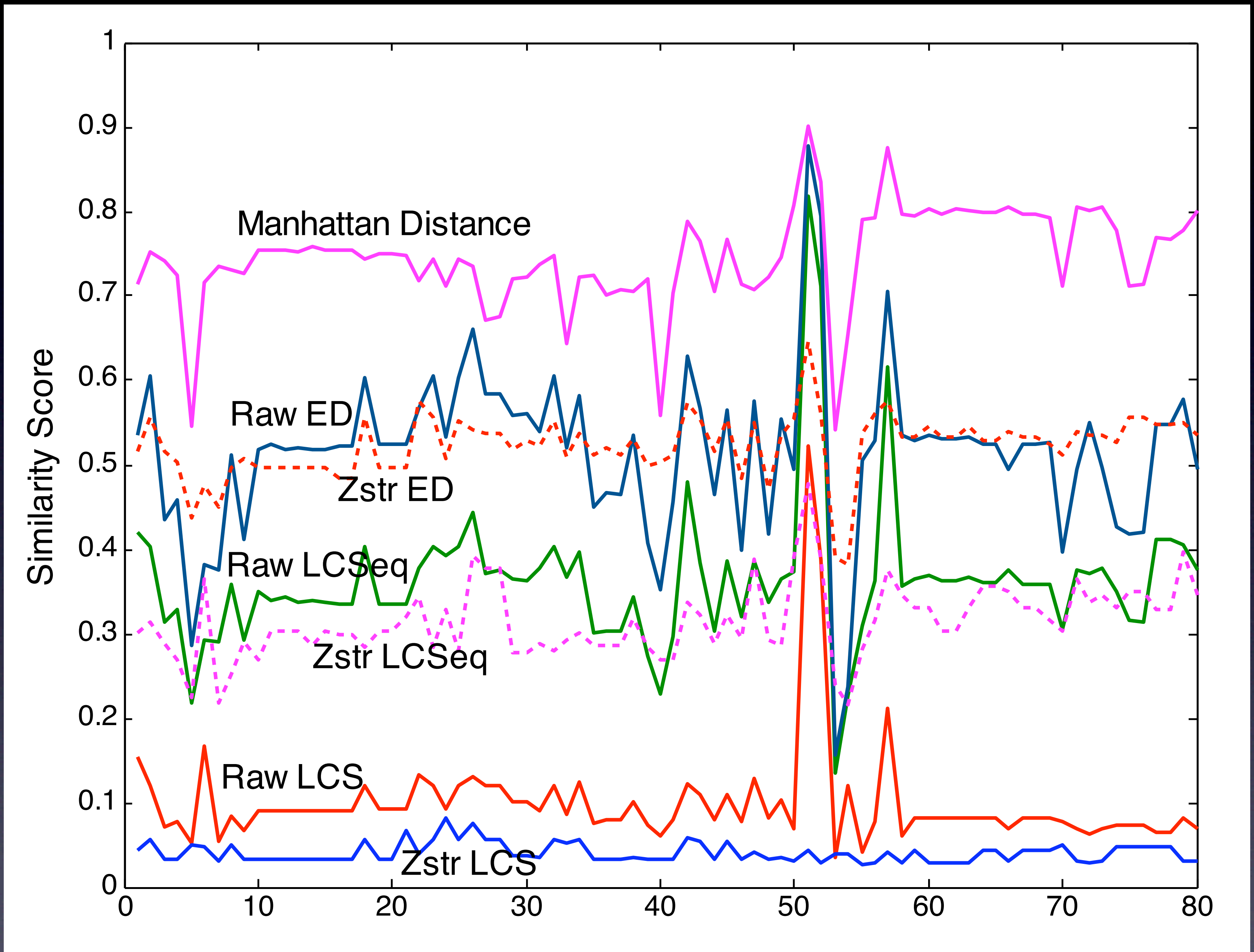
20-bit BFs

Expiry

Payload corroboration

- Techniques used: frequency distributions, Z-Strings, and n-gram Bloom filters
- Major questions:
  - How *efficient* are privacy transforms with payloads?
  - How *similar* are the different techniques at comparing packet content?
  - How well do the techniques *corroborate* alerts?
  - What kind of *signatures* can we generate?
  - What's the comparative *privacy gain*?

Payload similarity



Similarity score, 80 random pairs of "good vs. good"

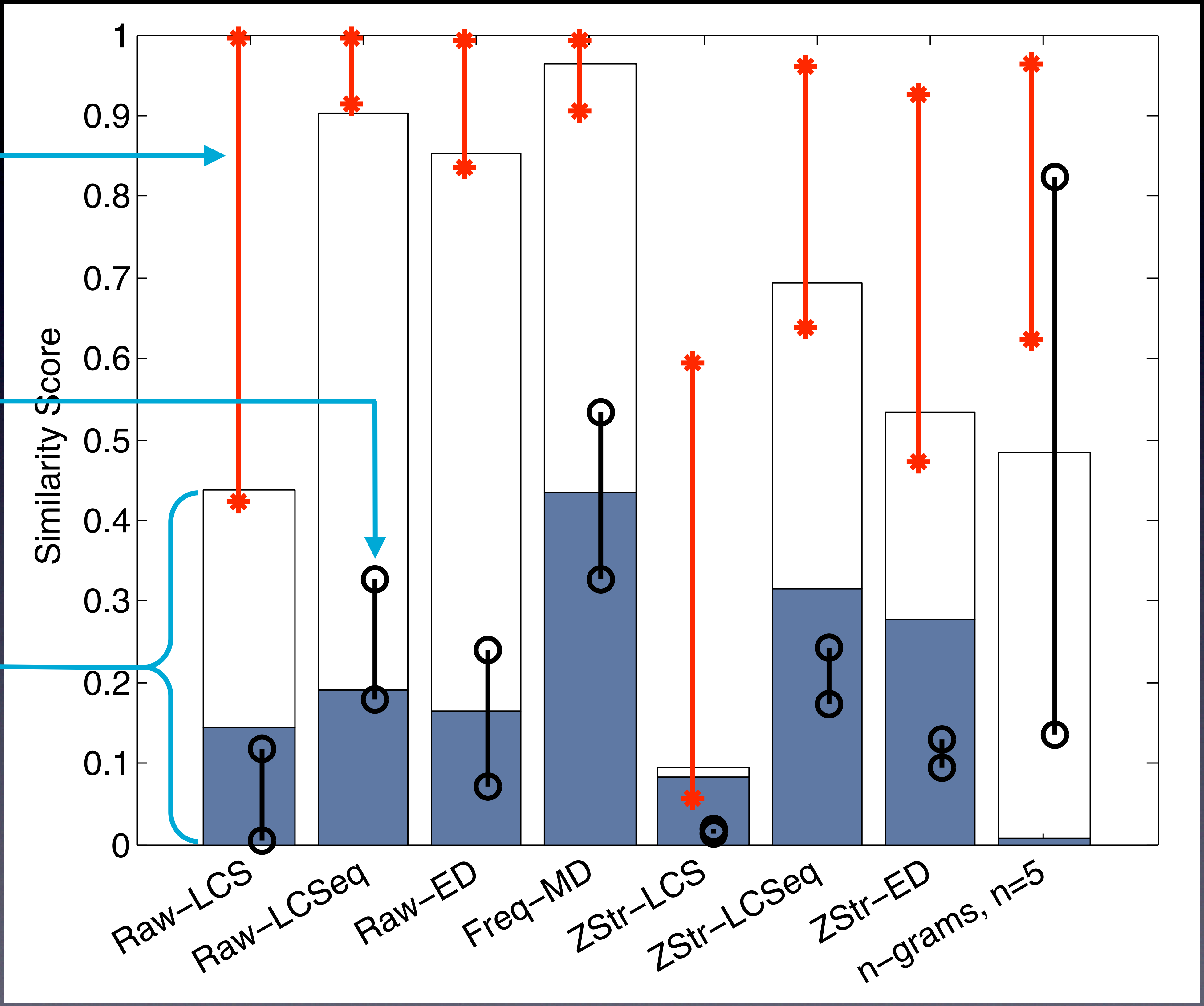
- High-level view of score similarities
- Most of the techniques are similar, except LCS (vulnerable to slight differences)
- ED and LCSeq very similar
- N-gram techniques not included (doesn't compute similarity over entire packet datagram)

Cross-domain corroboration

Range of scores across multiple instances of the same worm (CR or CRII)

Range of scores across instances of different worms (CR vs. CRII), e.g., polymorphism

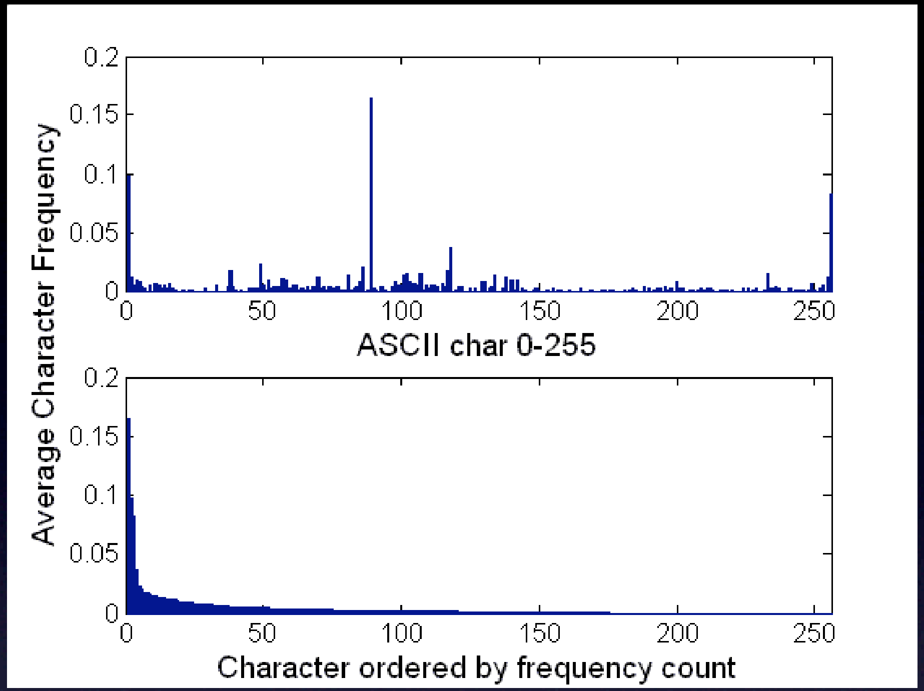
False positive score range; blue bar represents 99.9% percentile; white represents maximum score



Signature generation

```
GET./default.ida?XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%u
cbd3%u7801%u9090%u6858%ucbd3%u7801%u
9090%u6858%ucbd3%u7801%u9090%u9090%u
8190%u00c3%u0003%u8b00%u531b%u53ff%u
0078%u0000%u0
```

Original CRII packet (first 300 bytes)



Byte frequency distribution

```
* /def*ult.ida?XXXX*XXXX%u9090%
u6858%ucbd3%u7801%u9090%u6858%u
cbd3%u7801%u9090%u6858%ucbd3%u7
801%u9090%u9090%u8190%u00c3%u00
03%u8b00%u531b%u53ff%u0078%u000
0%u00=a HT*: 3379
```

Flattened 5-grams (first 172 bytes; "" implies wildcard)*

```
88 0 255 117 48 85 116 37
232 100 100 106 69 133 137
80 254 1 56 51
```

Z-String (first 20 bytes, ASCII values)

Payload privacy gain

- Frequency-based approaches
  - Characterize recovery likelihood R as the probability that someone can correctly guess the original content given the frequency distribution; for CRII, $R \approx 1/2^{8208}$
  - Even smaller (intractable) for a Z-String
- N-gram Bloom filter
  - For a 2^{12}-bit BF and 5-grams, $R = (2^{12}/2^{565})m$, where m is the number of distinct n-grams recovered
  - Surprisingly, a BF's FPs do not measurably affect correlation; "unlucky coincidence rate" = $(1/2^{12})m$, where m, the number of incorrectly verified n-grams, grows small very quickly

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

Related Work

- Event correlation (Rapide, DECS)
- Event distribution (Chord, Onion routing, Elvin, Siena, Gryphon, Astrolabe)
- Software monitoring (AProbe, Codebook, NESTOR)
- DIDS systems (EMERALD, GrIDS, DShield, DOMINO)
- Signature generation (Honeycomb, Earlybird, Autograph, Polygraph)
  - Vulnerability signatures (VSEF, Nemean, Shield, Vigilante)

Related Work (II)

- Existing privacy-preserving collaboration approaches (Lincoln, Kissner, FTN, Concept Hierarchies)
  - Focus primarily on IPs/"entity matches", as opposed to our more generic approach
  - No temporal corroboration
  - Scalability and practicality vary
- Model sharing (JAM, BARTER)
- Privacy-preserving data mining, secure computation, ZKP
- Bloom filter-based indices, search keys

Outline

- Motivation and Problem Statement
- Model
- Privacy Preservation Techniques
- Privacy Preservation & Intrusion Detection
- Related Work
- Conclusion

Contributions

- Typed event-driven privacy-preserving corroboration framework, written in Java
- The use of a diverse array of existing data structures to support corroboration
- New data structures and strategies for temporal corroboration: MRU BF, TSBF and Z-String clustering
- Extensive evaluation of these techniques with *real data*

Accomplishments

- Publications (so far): [Parekh06], [Wang06], [Parekh05], [Locasto05], [Gross04], [Keromytis03], [Kaiser03], [Kaiser02], [Gross01]
- KX/XUES demoed, deployed in 3+ applications, Worminator demoed, deployed at 5+ sites
  - http://worminator.cs.columbia.edu
- Grant support, successful presentations and demos to DARPA, NSA, DHS, ARO
- Patent application filed on aspects of Worminator work

Future Work

- Wider-scale deployment, evaluation
- Polymorphic/obfuscated worm detection, mimicry attacks
- *Posture-based* [Knight02] aggregation/exchange policies
- Privacy-preserving language and matching capabilities
- "Application communities" peer-to-peer application monitoring
- Privacy-preserving model-based authentication

✳ Malicious insider/watermarking problem
✳ Evaluation of event distribution strategies
✳ Automated IDS attacker profiling
✳ Automatic event schema discovery/generation/processing
✳ Automatic event processing rule generation

Conclusions

- Effective privacy-preserving event corroboration is *practical*, and for a broad variety of applications
- Event corroboration in the intrusion domain can provide a useful global picture of threats, exploits, and trustworthy peers
- A *typed* framework provides access to a heterogeneous set of corroboration tools depending on the preferred scenario

(the end)

Two Good Ideas

- Demonstrably effective techniques to enable privacy-preserving event sharing, *including* temporal constraints, even when original alerts aren't exchanged
- A framework to convince organizations to actually *share* information for distributed applications

Service failure detection

```
<state name="Start" timebound="-1" children="End" actions=""
       fail_actions="">
 <attribute name="Service" value="*service"/>
 <attribute name="Status" value="Started"/>
 <attribute name="ipAddr" value="*ipaddr"/>
 <attribute name="ipPort" value="*ipport"/>
 <attribute name="time" value="*time"/>
</state>

<state name="End" timebound="15000" children="" actions="Debug"
       fail_actions="Crash">
 <attribute name="Service" value="*service"/>
 <attribute name="State" value="FINISHED_STATE"/>
 <attribute name="ipAddr" value="*ipaddr"/>
 <attribute name="ipPort" value="*ipport"/>
 <attribute name="time" value="*time2"/>
</state>
```

Spam detection

```
<state name="a" timebound="-1" children="b">
 <attribute name="from" value="*1"/>
 <attribute name="messageID" value="*2"/>
</state>
<state name="b" timebound="100" count="1" children="" actions="A,B"
       fail_actions="F" absorb="true">
 <attribute name="from" value="*1"/>
 <attribute name="messageID" value="*2"/>
</state>
```

Privacy

- Many different forms; those explored in this thesis include
  - Data privacy: privacy of data semantics
  - Source anonymity: privacy of producer
- Physical privacy *not* covered
- Time privacy "optional"

Timestamping

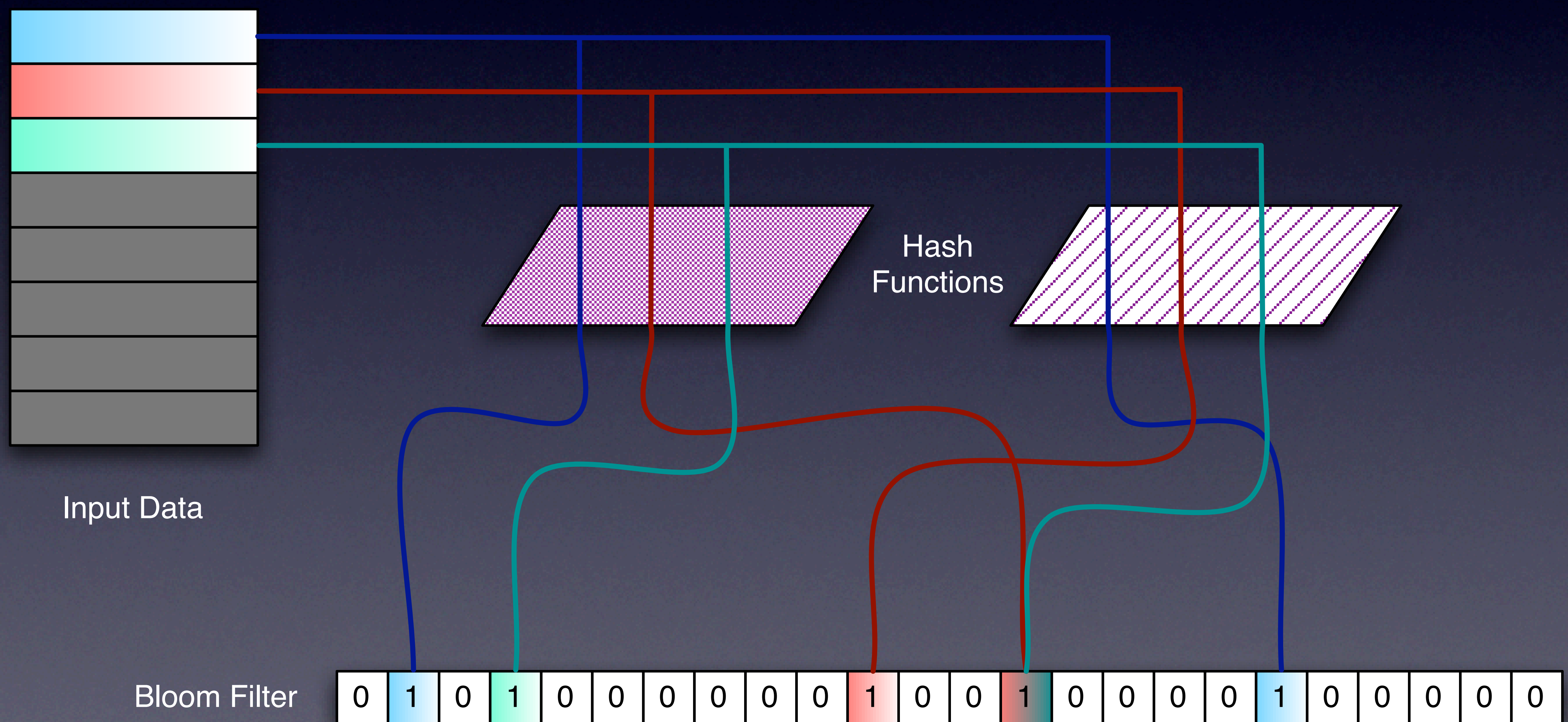
- Ideal: created by producer upon event creation
  - Upper bound, upper/lower bound,exact
- Implicit timestamp
  - At publication
  - At receipt; can lead to ordering errors
- No timestamping: pure intersection

Levels of anonymity

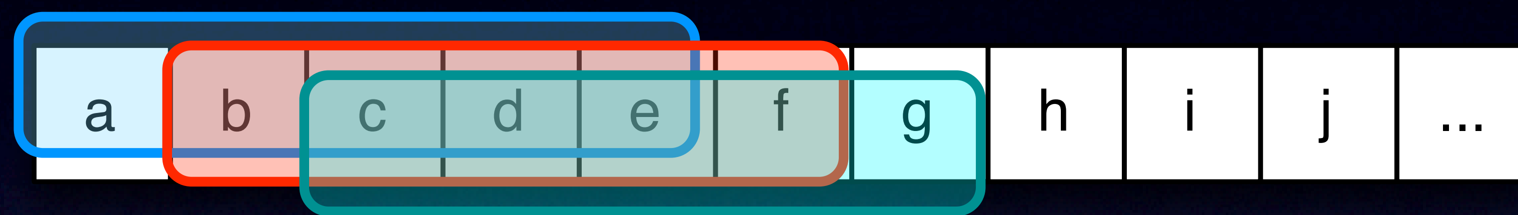
- Non-anonymous
- Anonymous but differentiable
- Anonymous but categorizable
- Fully anonymous (not supported; very difficult problem, e.g., Sybil attacks)

Bloom filters

- Classic hash-based data structure [Bloom60]



Incremental analysis



5-grams.

$$\mathcal{S}(e,\mathcal{E}') = \begin{cases} \dfrac{\sum_{i=0}^{k} f(g_i)}{k} & : \quad \mathcal{E}' \text{ is frequency-modeled} \\ \dfrac{\sum_{i=0}^{k} \mathcal{F}(g_i)}{k} & : \quad \mathcal{E}' \text{ is binary-modeled} \end{cases}$$

Similarity metric for a set of n-grams.

Frequency model distance metrics

- Event against model: simplified Mahalanobis distance

$$D'_{Mah}(x, \mu) = \sum_{i=0}^{n-1} (|x_i - \mu_i| / (\sigma_i + \alpha))$$

- Model vs. model: Manhattan distance

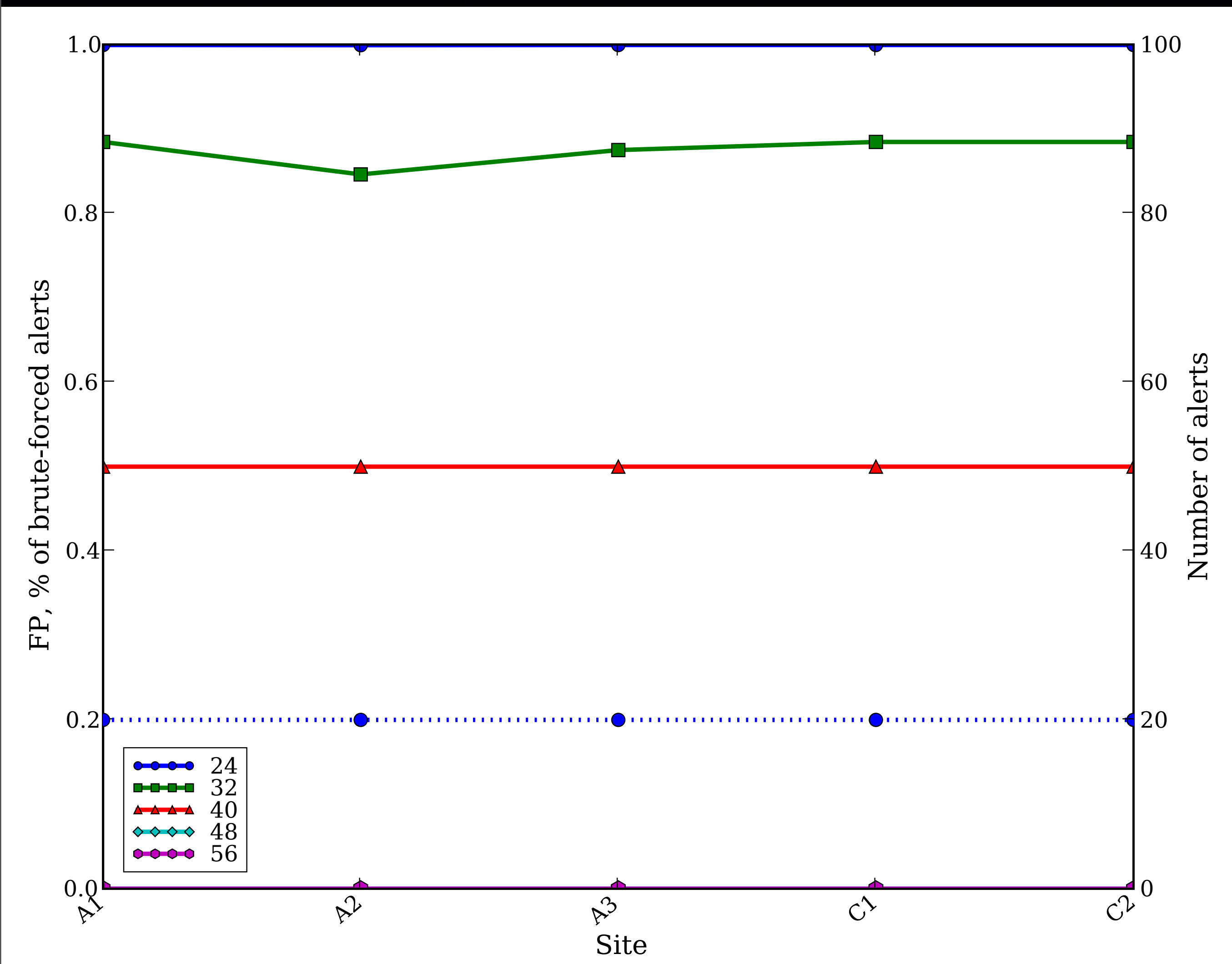
$$D_{Man} = \sum_{i=0}^{n-1} |x_i - y_i|$$

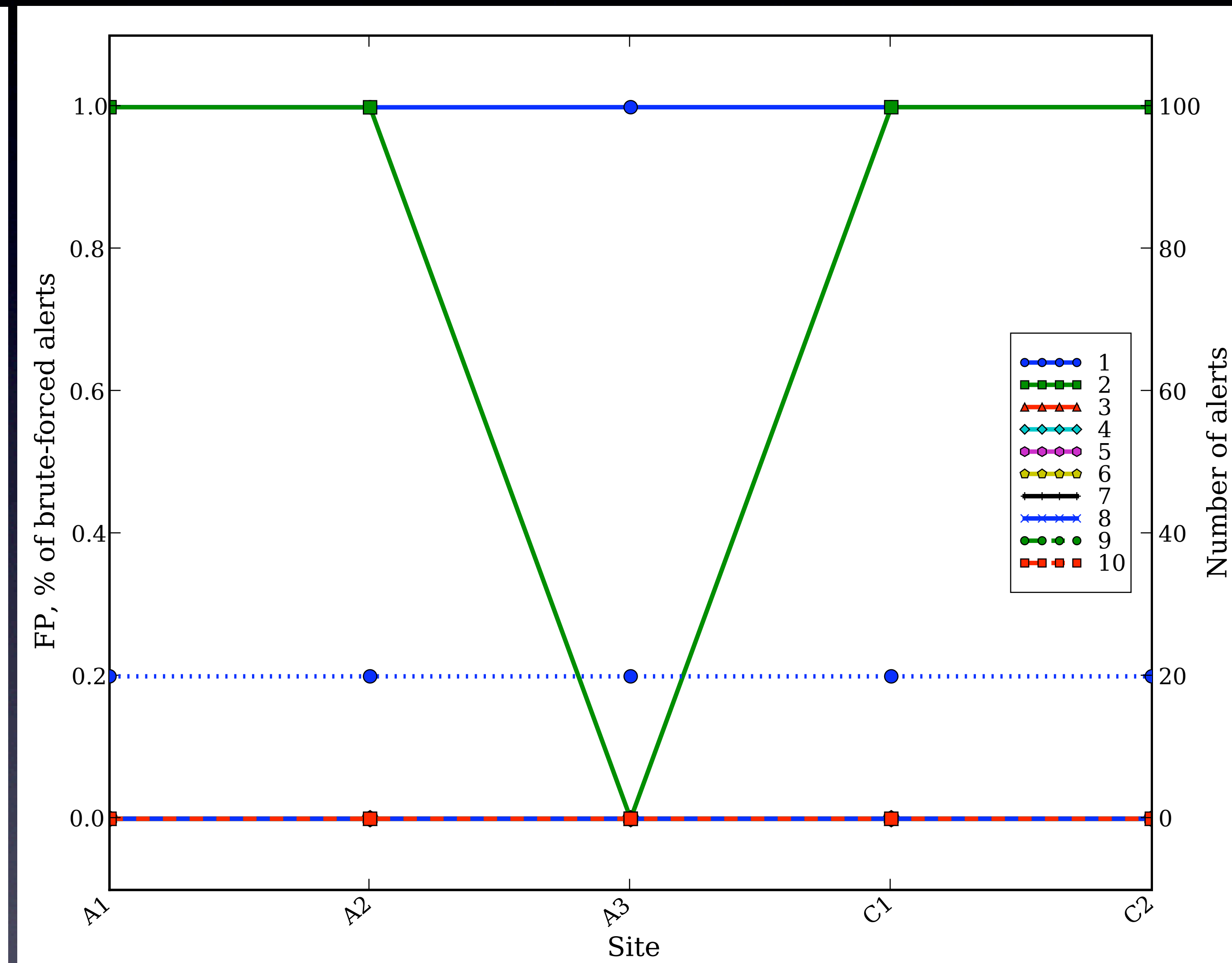
IP Data Collected

Site	Time (days)	# Alerts	# Alerts/Min.	# Distinct IPs	# Distinct IP/port pairs
Academic 1	314.87	3919604	8.64	86108	4576155
Academic 2	28.53	823631	20.04	28838	844288
Academic 3	164.56	2811553	11.86	45255	3605271
Academic 4	14.95	54518	2.53	2398	2541
Commercial 1	242.52	923482	2.64	119675	325283
Commercial 2	373.68	543979	1.01	60585	378062

Brute-forcing sparse BFs

(20 alerts per BF)

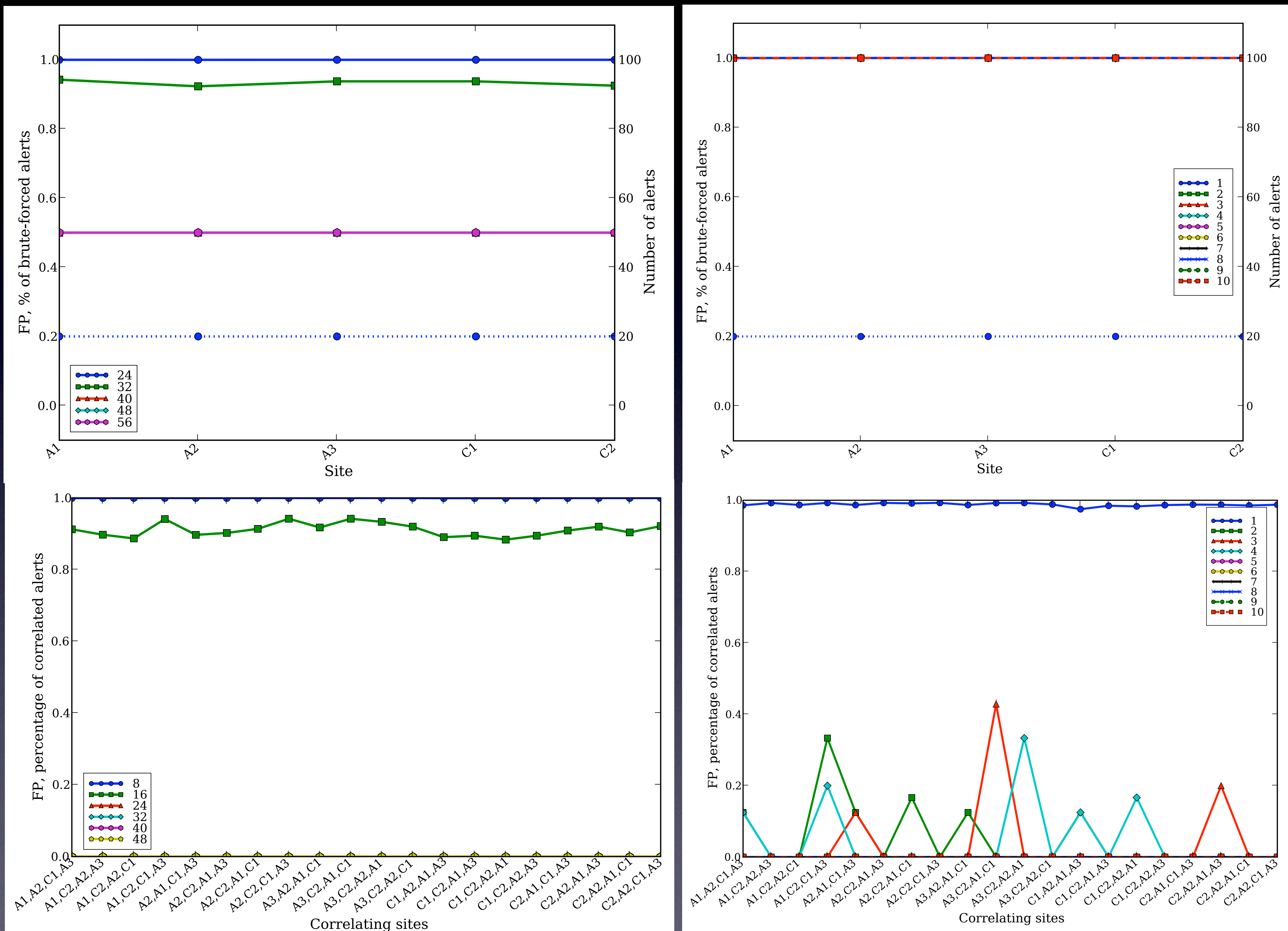




Hash set

Bloom filter

Sparse **noisy** sets/BFs, IP/port



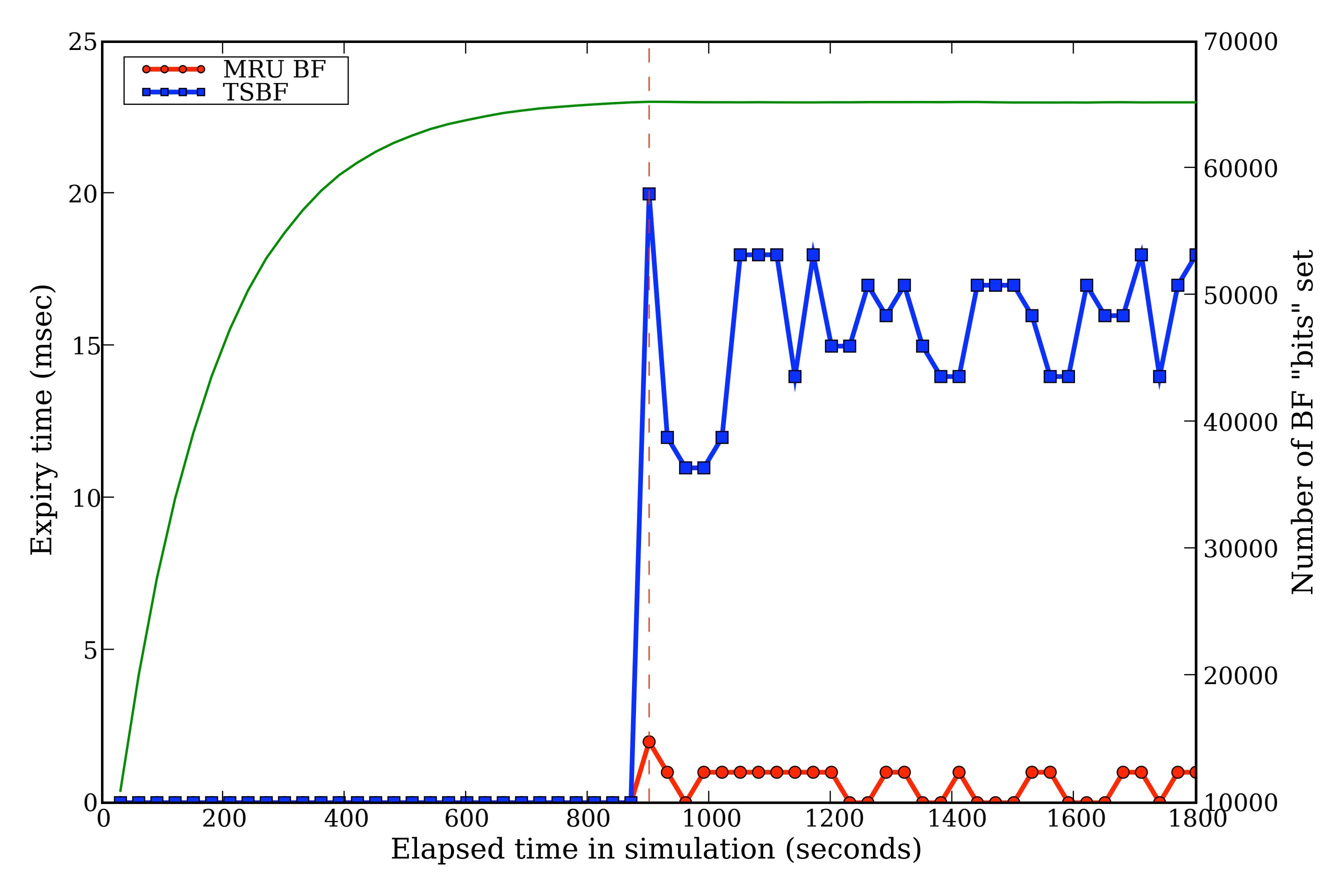
Privacy

Corroboration

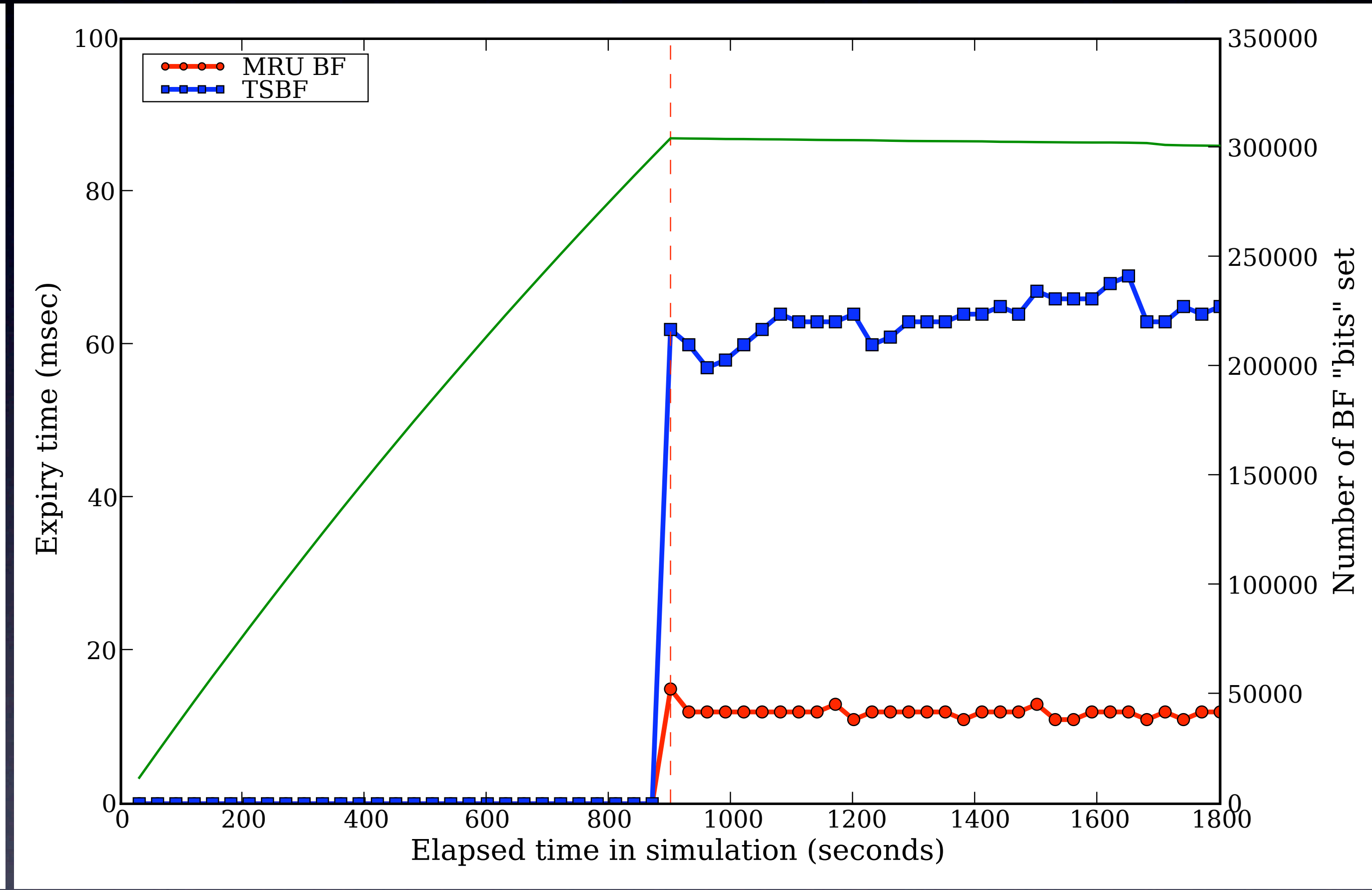
Hash set (100% noise)

Bloom filter (10% noise)

TSBF, MRU BF Expiry



16-bit BF



20-bit BF

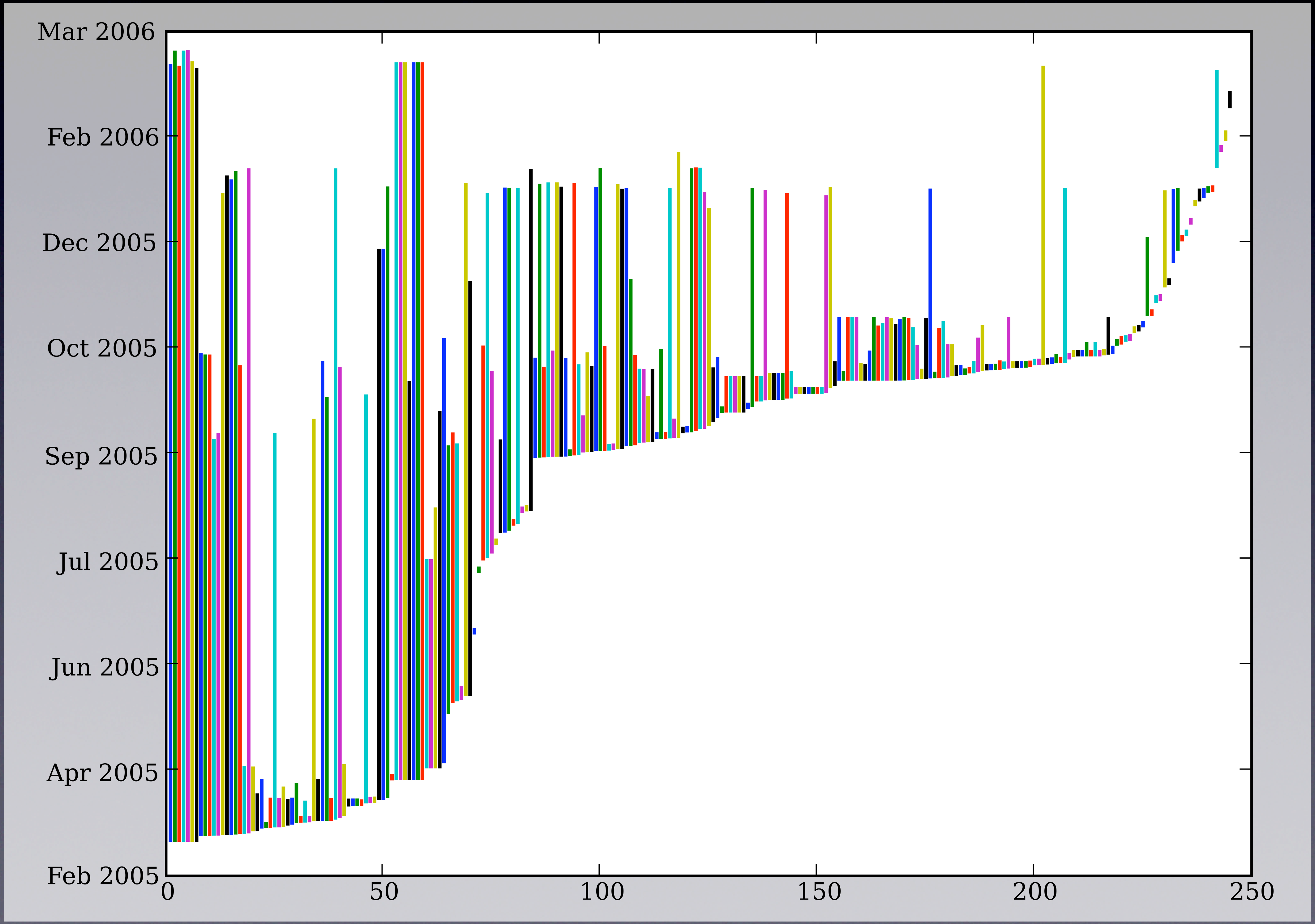
Longitudinal study of IP scans

- Worminato’s goal is to enable precisely this type of analysis
- Three key *longitudes* analyzed
  - Over time
  - Over geographical, network space
  - By target

# Sites	# Site/IPs	Avg Scan Len (days)
1	307050	7.14
2	22250	10.86
3	10074	17.20
4	3228	29.86
5	245	70.77

Scan length distribution

5-site scanners



Stealthiness

Source IP	Scan Length (days)	# Alerts	*St*
61.185.246.34	257.73	7	3.144e-07
207.218.223.98	302.96	9	3.438e-07
61.129.45.54	302.12	10	3.831e-07
207.218.223.91	270.71	9	3.848e-07
207.218.223.89	271.16	11	4.695e-07
207.218.223.93	301.50	13	4.990e-07
66.150.8.18	199.92	10	5.789e-07
62.189.244.254	287.28	17	6.849e-07
61.172.250.90	234.36	14	6.914e-07
206.253.195.10	293.14	19	7.502e-07

Table 6.5: Top 10 stealthy scanners detected at 4 sites

Source IP	Scan Length (days)	# Alerts	*St*
207.218.223.92	300.14	12	4.628e-07
207.218.223.103	302.52	17	6.504e-07
69.7.175.21	293.50	41	1.617e-06
69.25.27.10	225.52	33	1.694e-06
161.170.254.232	299.29	51	1.972e-06
219.148.119.199	227.03	45	2.294e-06
66.151.55.10	303.12	62	2.367e-06
62.73.174.150	338.39	90	3.078e-06
64.41.241.171	338.39	90	3.078e-06
64.56.168.66	338.39	96	3.283e-06

Table 6.6: Top 10 stealthy scanners detected at 5 sites

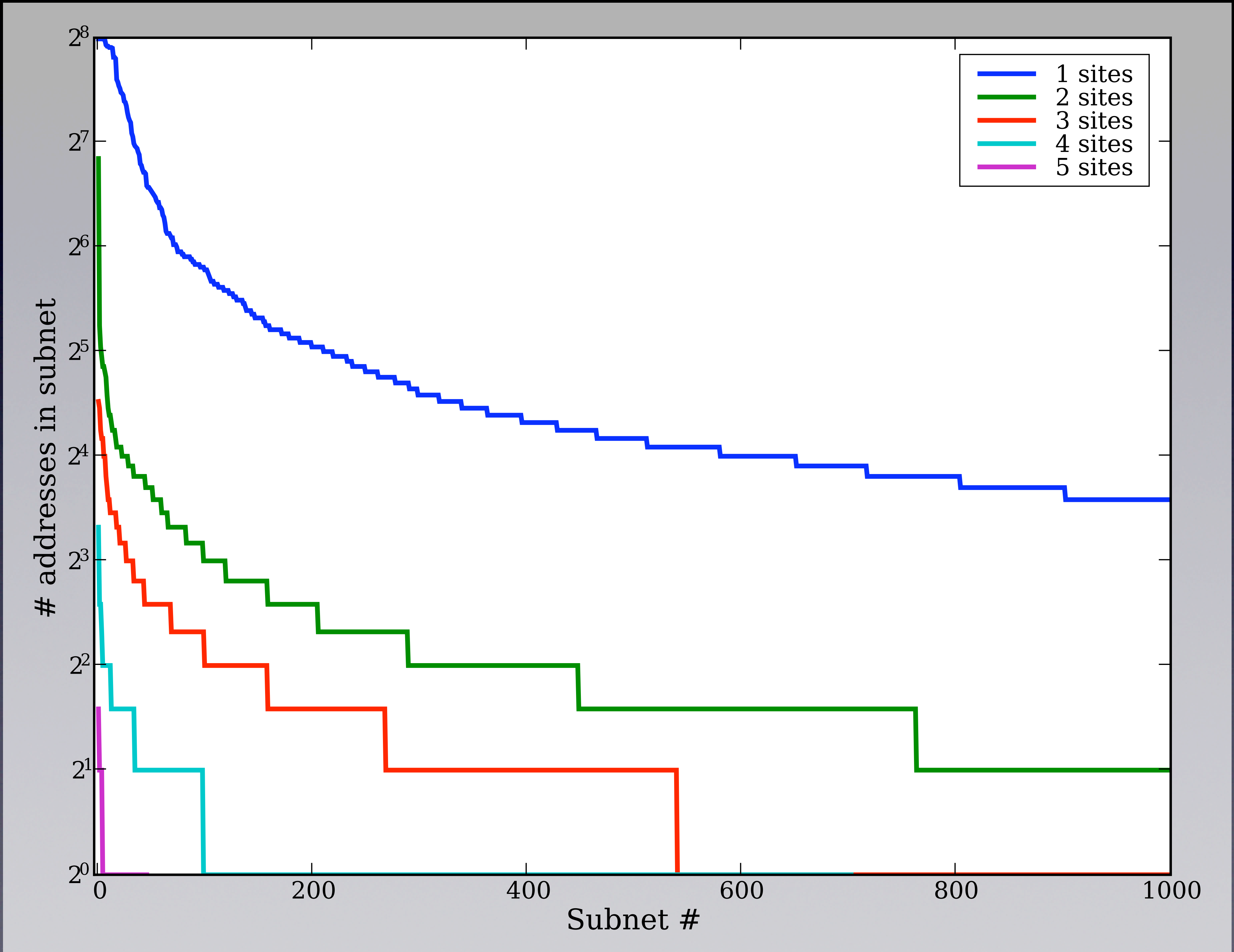
Interestingly, the items *in italics* are all from the same subnet

So, which addresses from 207.218.*?

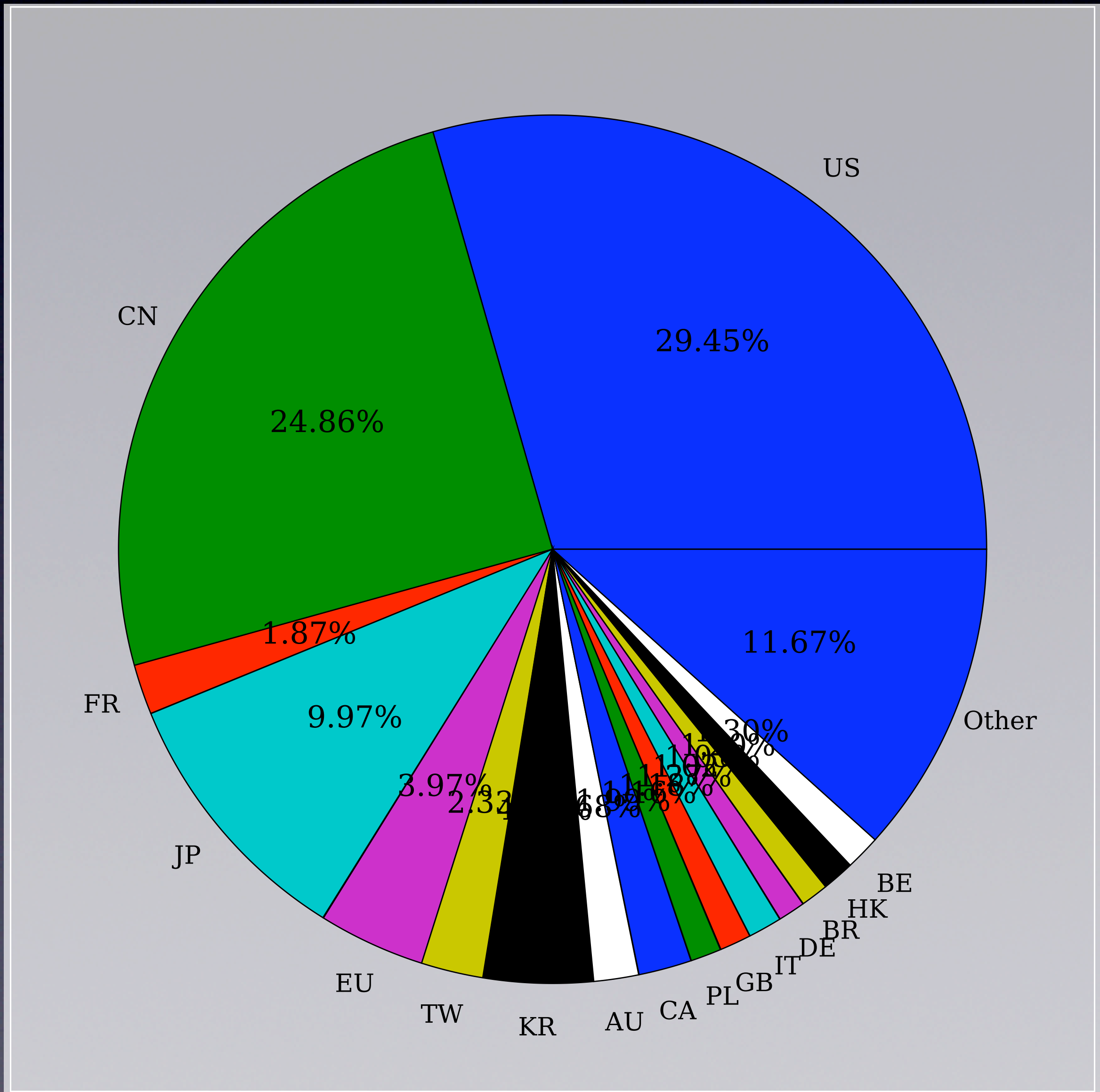
Source IP	#sites	#alerts	Scan len	Hostname
207.218.223.92	5	12	300.14	ivhou-207-218-223-92.ev1servers.net
207.218.223.103	5	17	302.52	ivhou-207-218-223-103.ev1servers.net
207.218.223.89	4	11	271.16	ivhou-207-218-223-89.ev1servers.net
207.218.223.91	4	9	270.71	ivhou-207-218-223-91.ev1servers.net
207.218.223.93	4	13	301.50	ivhou-207-218-223-93.ev1servers.net
207.218.223.98	4	9	302.96	ivhou-207-218-223-98.ev1servers.net
207.218.223.94	3	10	300.44	ivhou-207-218-223-94.ev1servers.net
207.218.223.95	3	8	301.51	ivhou-207-218-223-95.ev1servers.net
207.218.223.97	3	8	63.06	ivhou-207-218-223-97.ev1servers.net
207.218.223.99	3	10	271.10	ivhou-207-218-223-99.ev1servers.net
207.218.223.102	3	10	297.12	ivhou-207-218-223-102.ev1servers.net
207.218.223.90	2	9	20.04	ivhou-207-218-223-90.ev1servers.net
207.218.223.101	2	5	270.55	ivhou-207-218-223-101.ev1servers.net
207.218.223.100	1	1	3.99	ivhou-207-218-223-100.ev1servers.net
207.218.223.132	1	4	2.12	ns1.rackshack.net
207.218.223.162	1	6	1.05	ns2.rackshack.net

Table 6.7: Subnet search results for 207.218.223.0/24

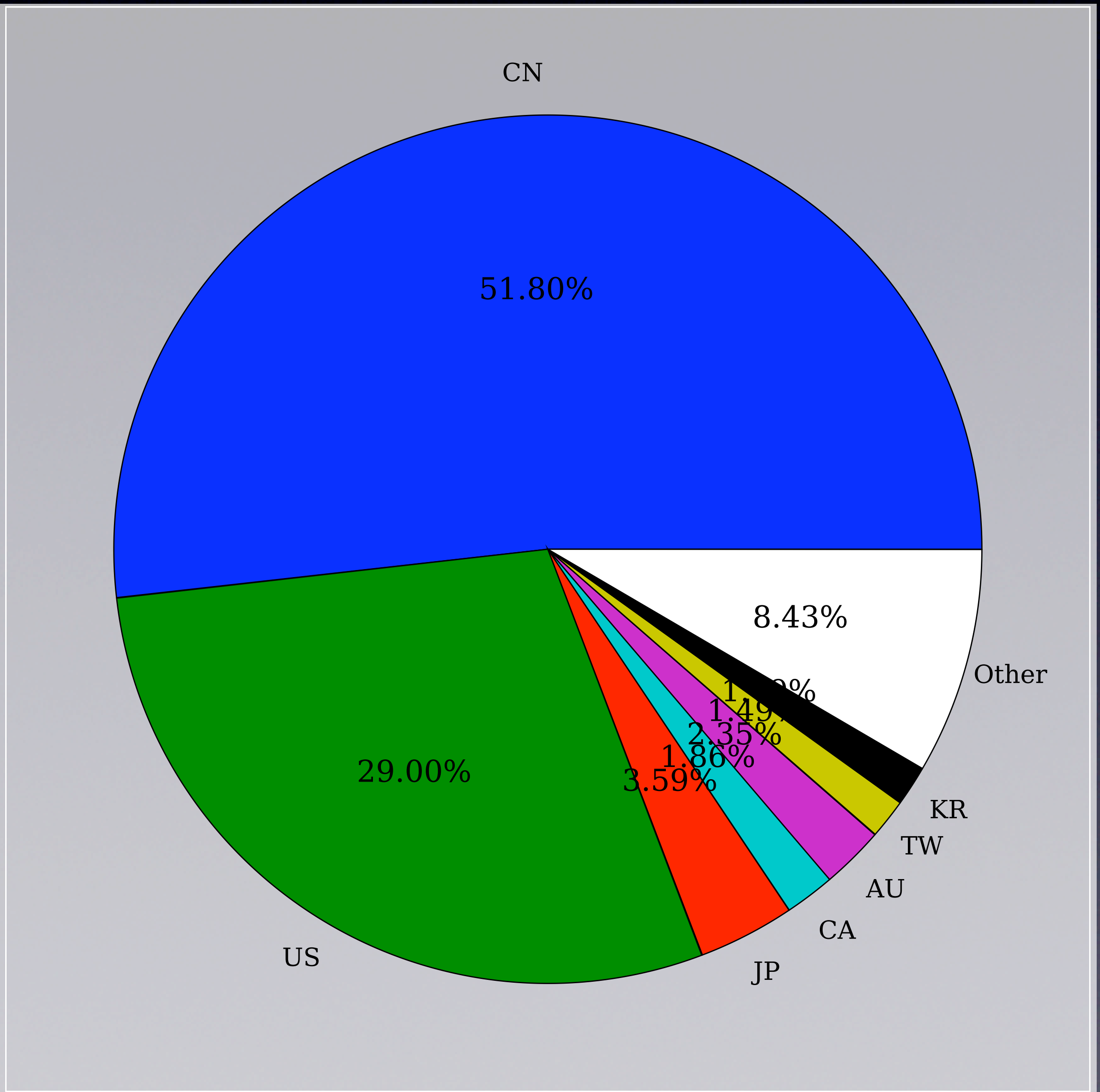
Subnet scanners



Source geography

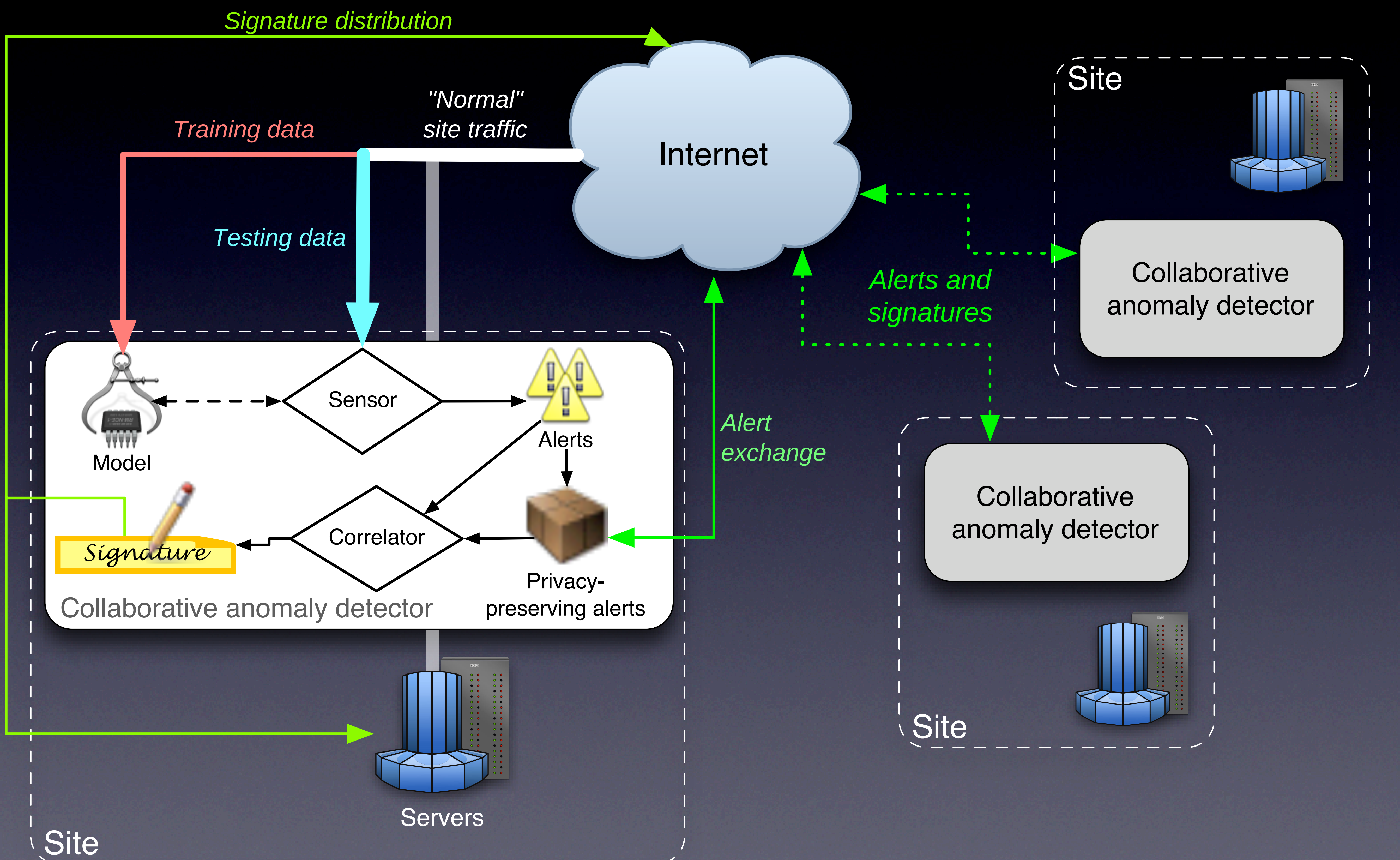


2-site, by IP

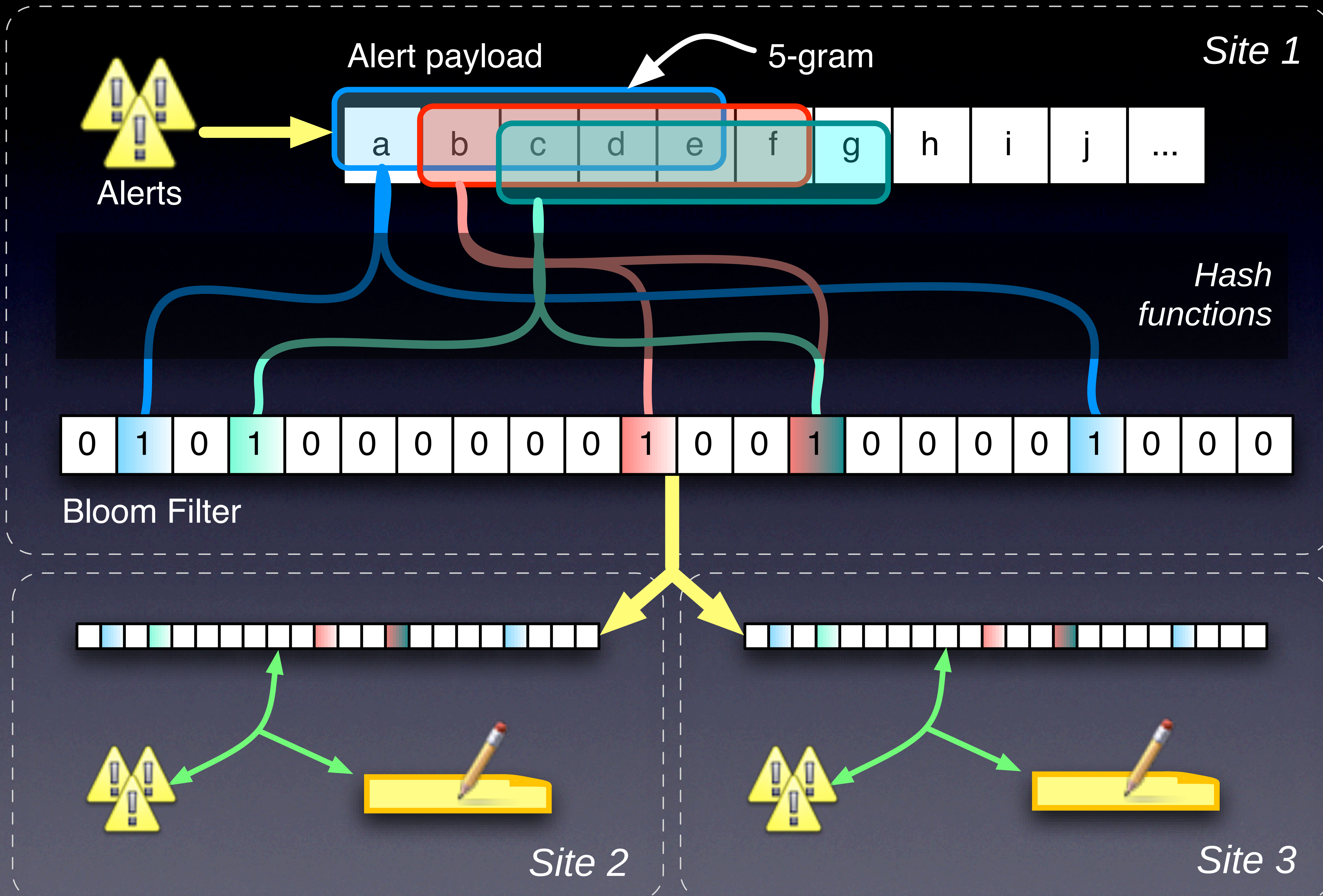


4-site, by IP

Payload: Big picture

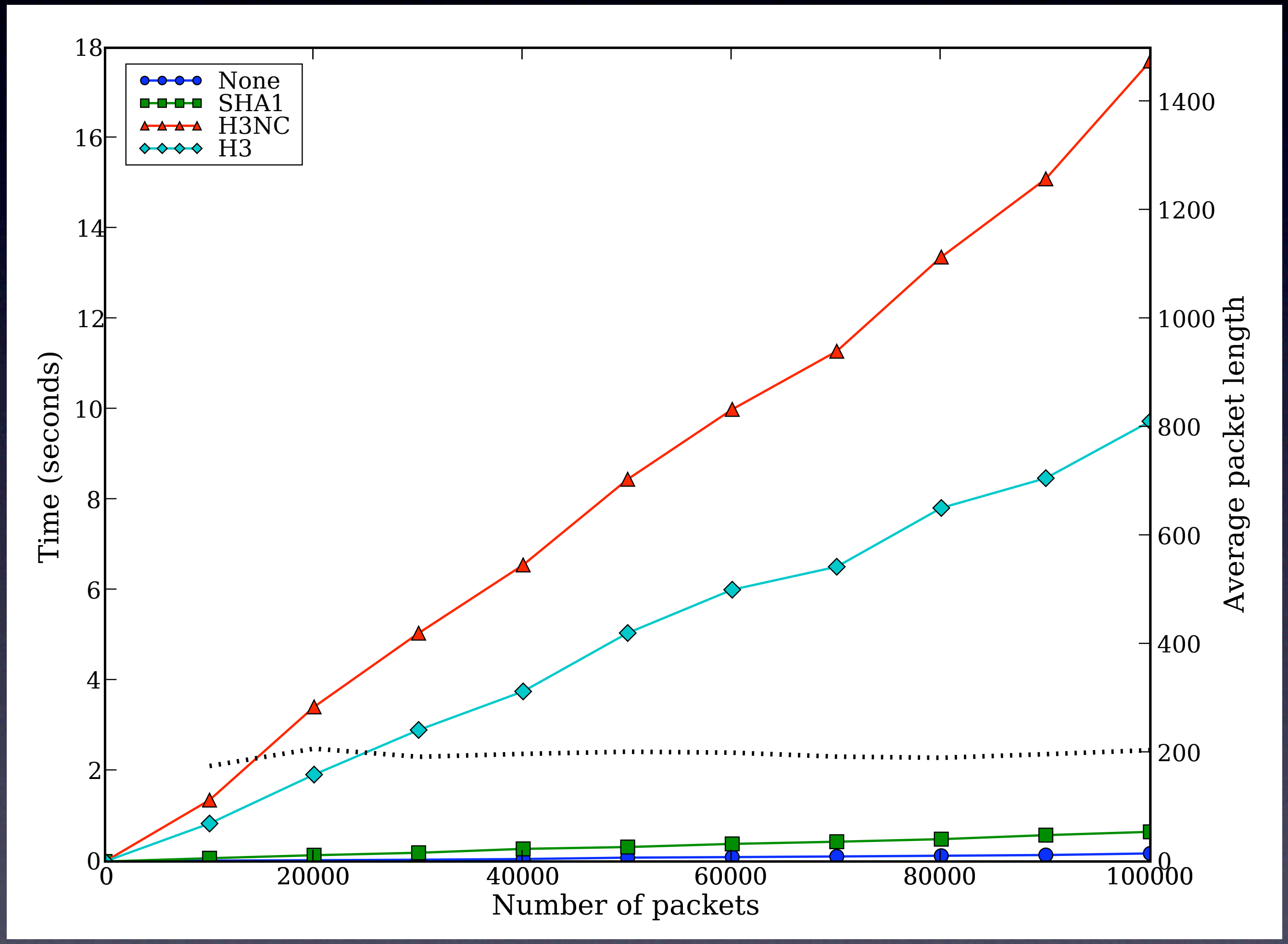


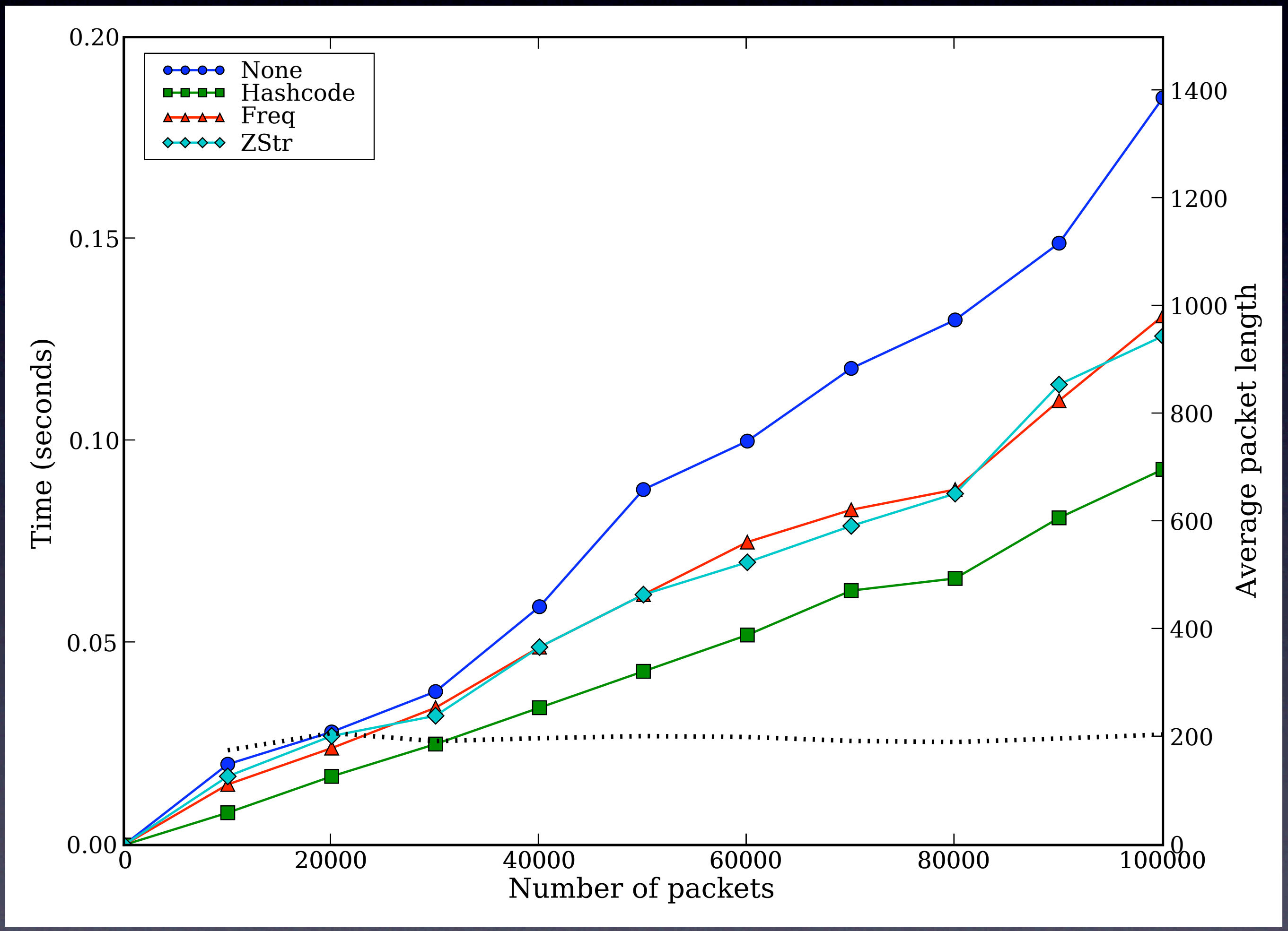
Bloom filter n-gram analysis



Hash, freq performance

(full payloads)



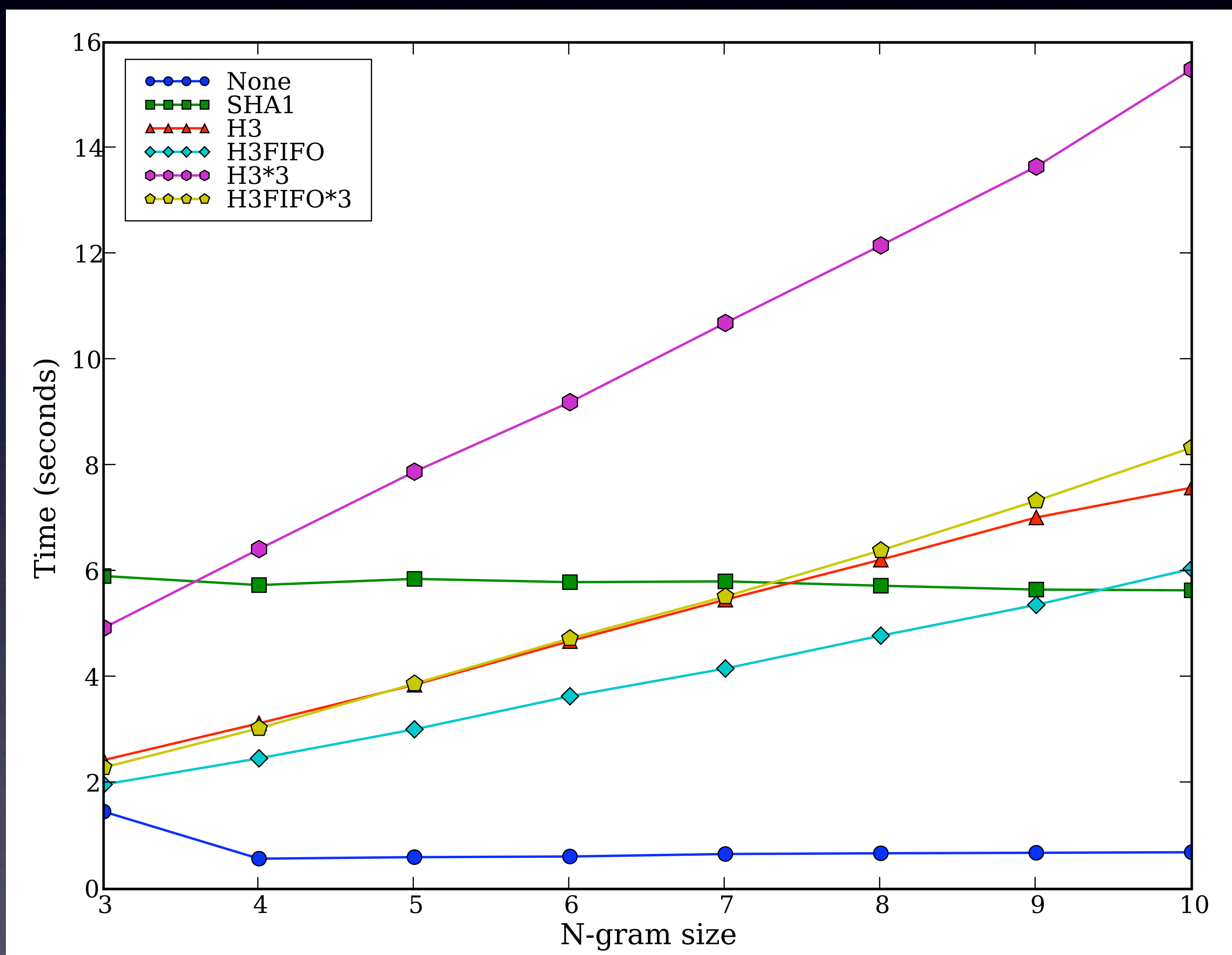


Hashing, entire packet

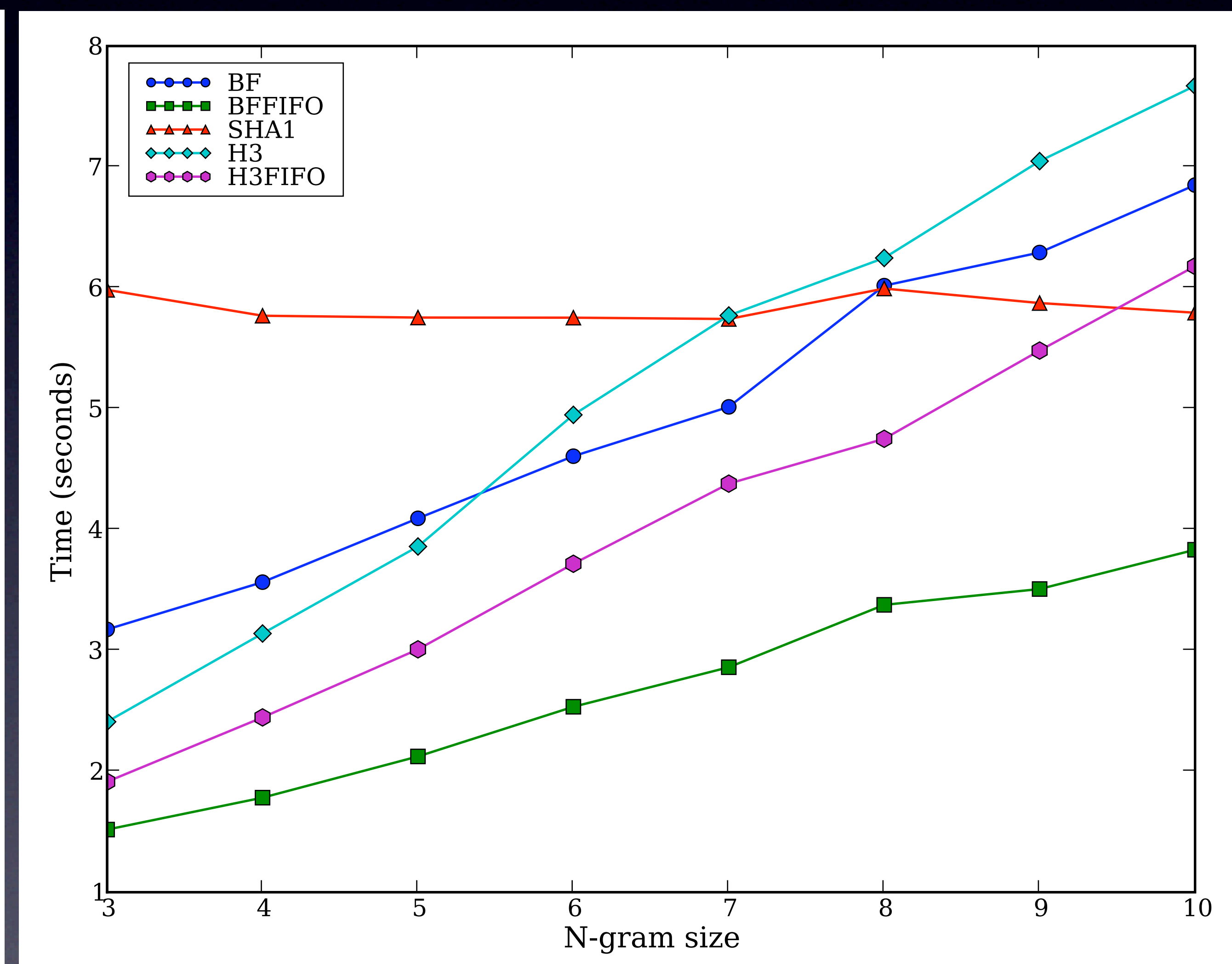
Frequency transform, entire packet

N-gram performance

HTTP traffic, 60,000 packets



Hash set



Bloom filter

Payload corroboration

Site A

```
Example
malicious code
```

```
Exa, xam, amp, mpl, ple,
le□, e□m, □ma, mal, ali,
lic, ici, cio, iou, ous,
us□, s□c, □co, cod, ode
```

```
0000011010101101001101100110
```

```
Exa, xam, amp, mpl,
ple, le□, e□m, □ma,
mal, cod, ode
```

```
Example mal*code
```

Site B

```
Example
malcode
```

```
Exa, xam, amp, mpl, ple,
le□, e□m, □ma, mal, alc,
lco, cod, ode
```

```
0000001011011101000100110000
```

```
Exa, xam, amp, mpl,
ple, le□, e□m, □ma,
mal, cod, ode
```

```
Example malcode
```

Evaluating payload corroboration

- Three sets of randomly-sampled traffic
  - *www1* and *www2*: Columbia webservers, 100 packets each
  - Malicious packet dataset, 56 packets
  - "Known ground truth"
- Evaluation
  - Similarity: arranged into three pairs (good vs. good, bad vs. bad, good vs. bad)
  - Corroboration: mix attack collection into real traffic, measure *separation* with 100% detection

Payload similarity: setup

- Arranged into three sets of pairs
  - 10,000 “good vs. good”
  - 1,540 “bad vs. bad”
  - 5,600 “good vs. bad” between *www1* and the malicious dataset
- To compare the difference more precisely, normalize and compare scores
  - Compute similarity score vectors V_A, V_B
  - Match their medians
  - Scale ranges proportionally so min and max values match
  - Compute *Manhattan distance* between normalized vectors
- Each privacy-enabled technique is compared against Raw-LCSeq (baseline)

Payload similarity (II)

Type	Raw-LCseq	Raw-LCS	Raw-ED	MD	ZStr-LCS	ZStr-LCSeq	ZStr-ED
G-G	0	.0948	.0336	.0669	.2079	.0794	.0667
B-B	0	.0508	.0441	.0653	.0399	.0263	.0669
G-B	0	.0251	.0241	.0110	.0310	.0191	.0233

Normalized similarity scores (lower is better)

- Unsurprisingly, Raw-ED closest to Raw-LCSeq
- *All* privacy-preserving methods are close when correlating pairs including attack traffic; may be leveraging difference between byte distributions
  - Manhattan distance between packet freq distributions best

Cross-domain corroboration

- Goal: measure performance in identifying true alerts from false positives
  - Ideal: true positives have very high similarity scores, while false positives have very low scores
- Mix the collection of attacks into two hours of traffic from *www* and *www1*
  - Multiple, differently-fragmented instances of Code Red and Code Red II to simulate a real worm attack
- Mixed sets are run through PAYL and Anagram, with alerting threshold reduced so that 100% of attacks are detected, but with possibly higher FP rates

Cross-domain corroboration (II)

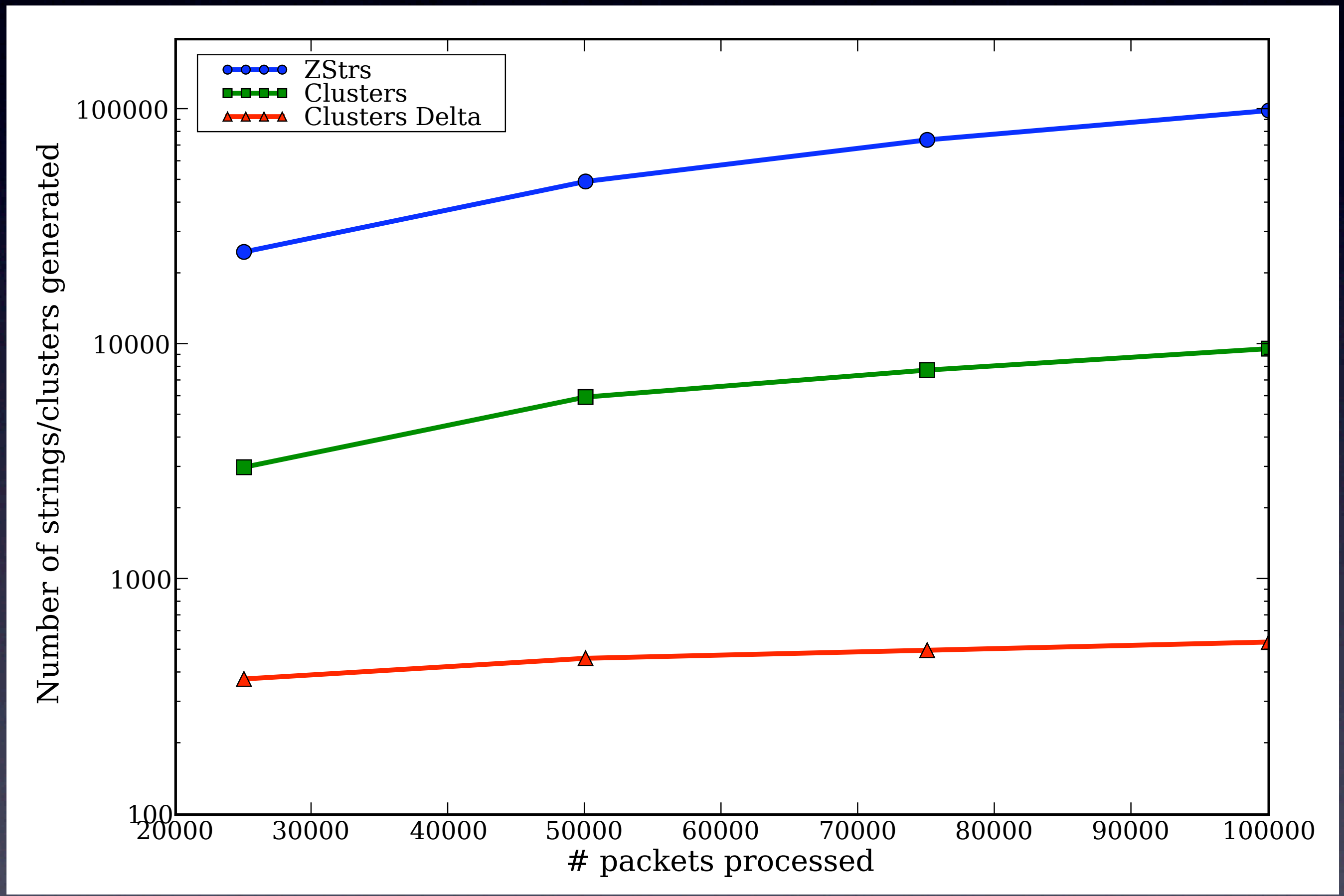
- Correlation of identical (non-polymorphic) attacks works accurately for all techniques
  - Non-fragmented attacks score near 1
  - Z-Strings (MD, LCseq, ED) and n-grams handle fragmentation well
- Polymorphism is hard to detect; only Raw-LCSeq and n-grams score well
- Overall, n-grams are particularly effective at eliminating false positives, and Bloom filters enable privacy preservation

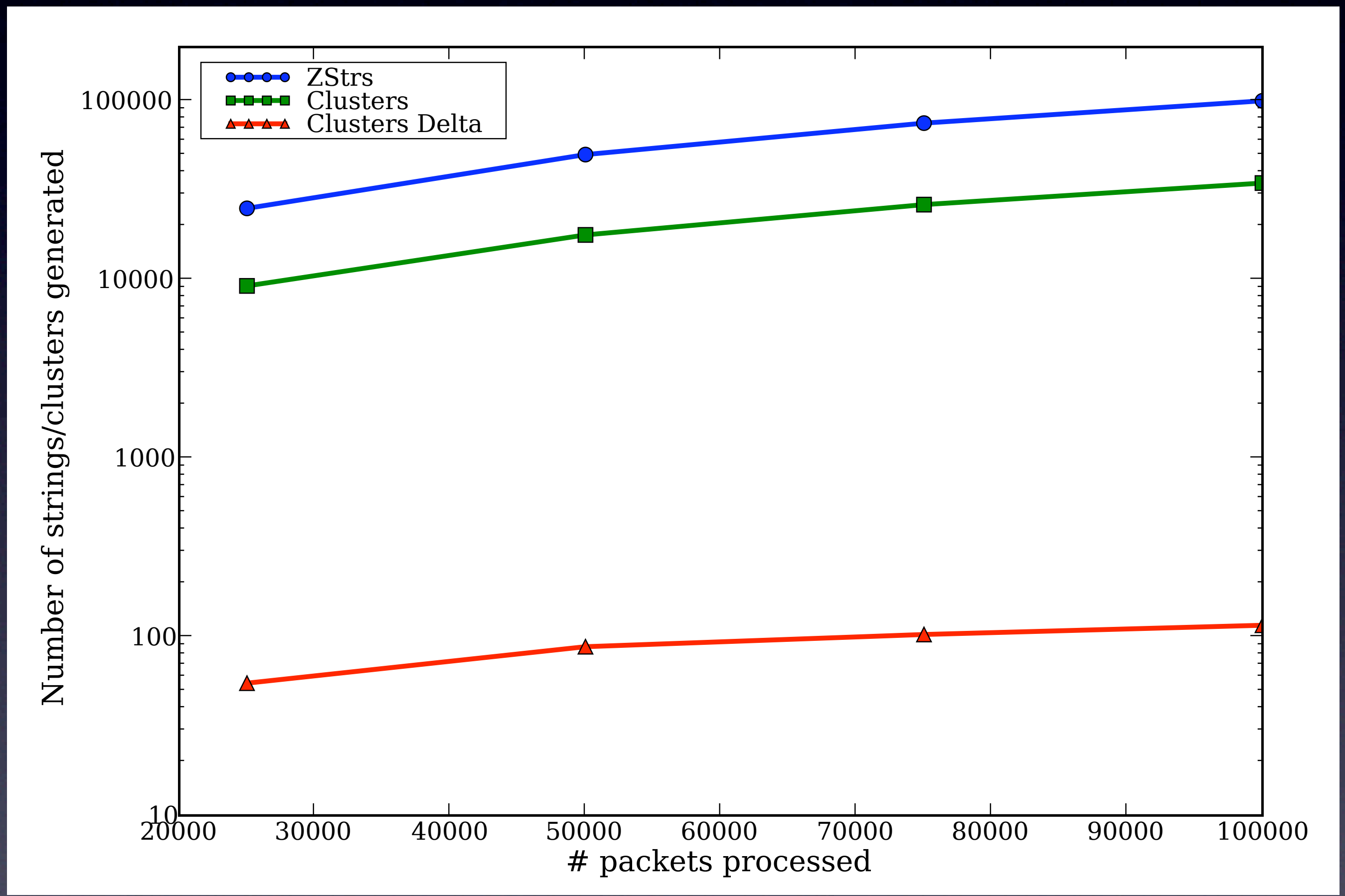
Signature generation

- Each class of techniques can generate its own signature
- Raw packets: Exchange LCS/LCSeq
  - Not privacy-preserving
- Byte frequency/Z-Strings
  - Given the frequency distribution, Z-Strings generated by ordering from most to least frequent and dropping the least frequent
- N-grams
  - Robust to reordering or fragmentation
  - If position information is available, can “flatten” into a deployable string signature

Z-String Clustering

Technique comparison



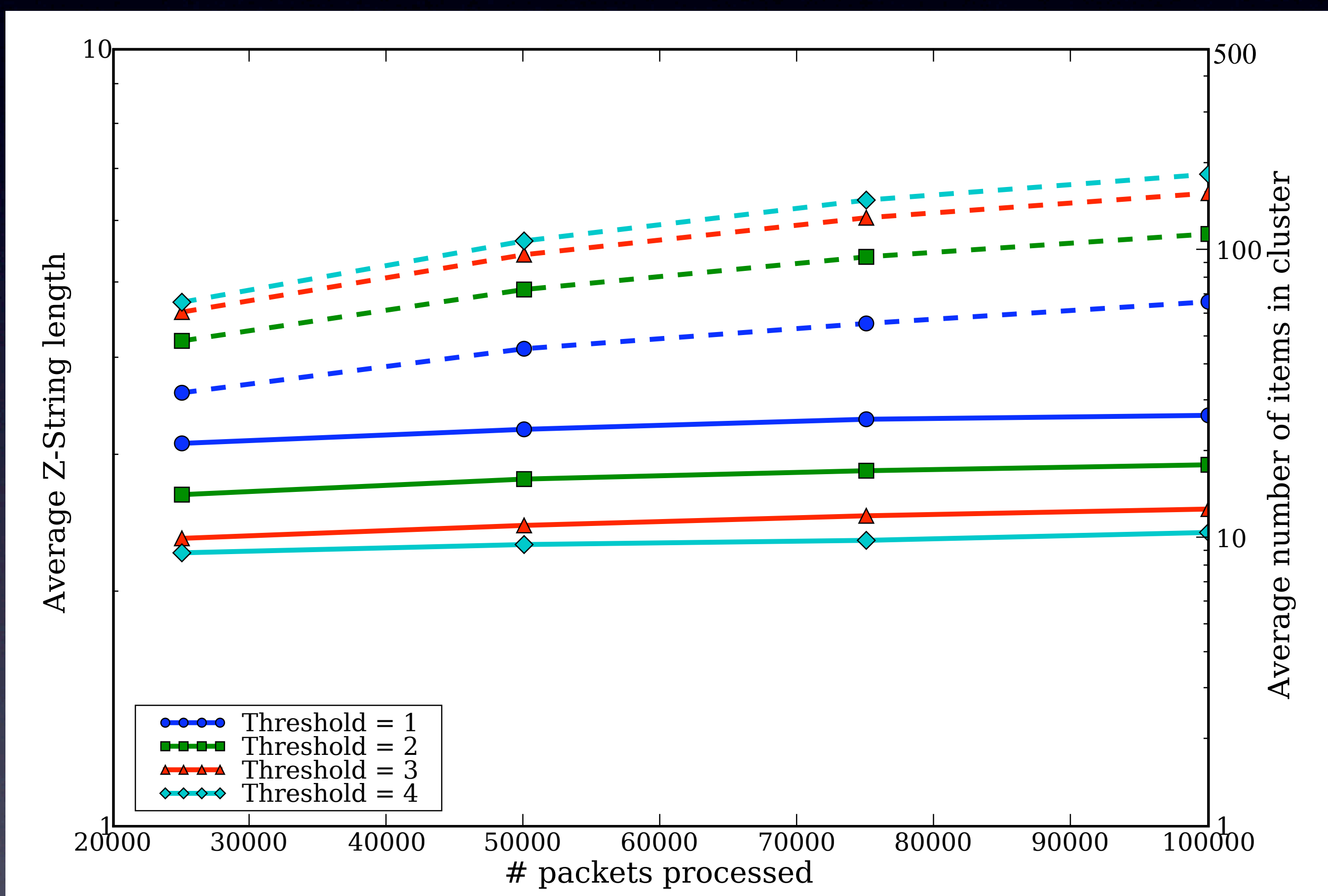


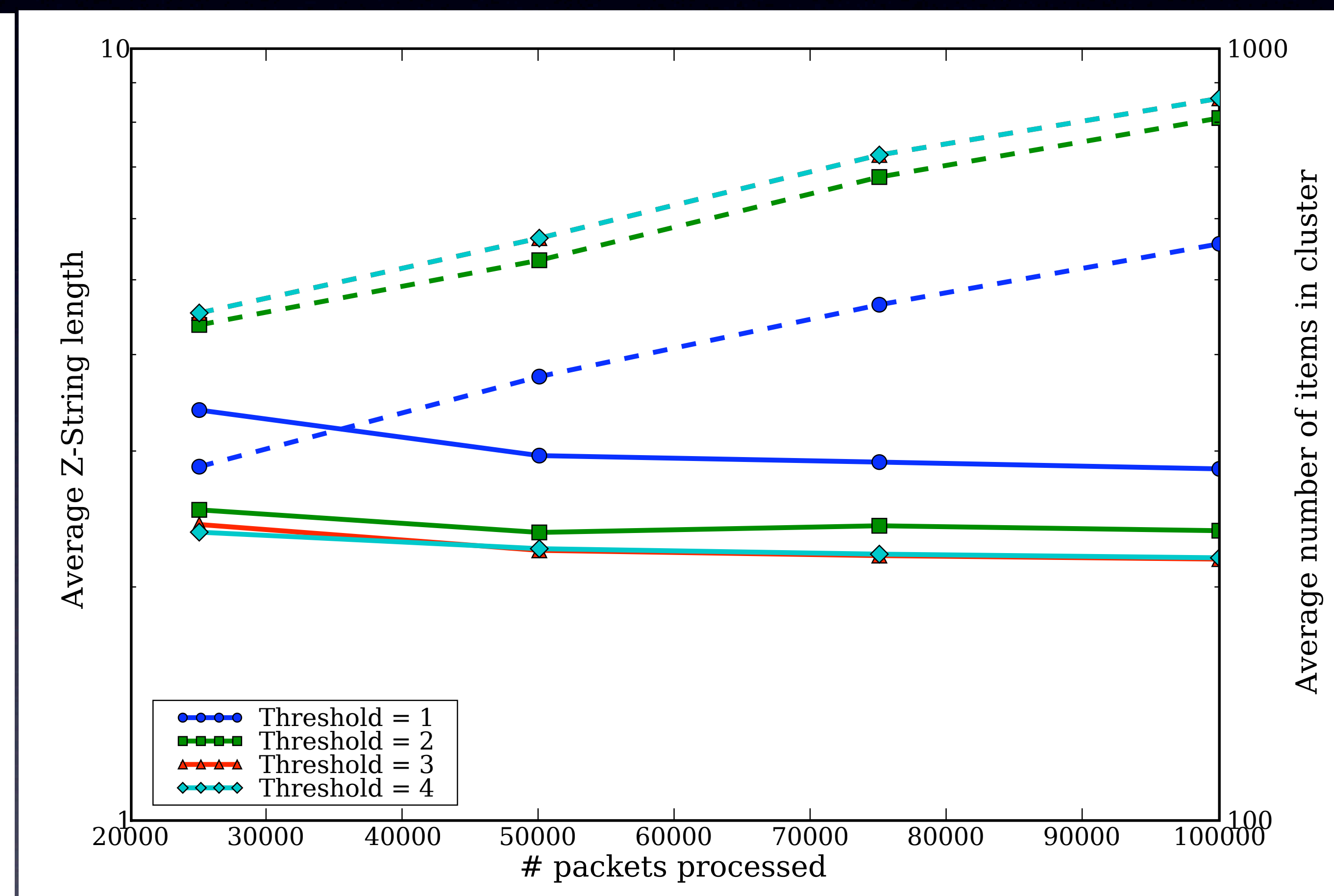
Normal HTTP traffic, threshold = 4

CRII traffic, threshold = 4

Z-String Clustering

Cluster Delta technique



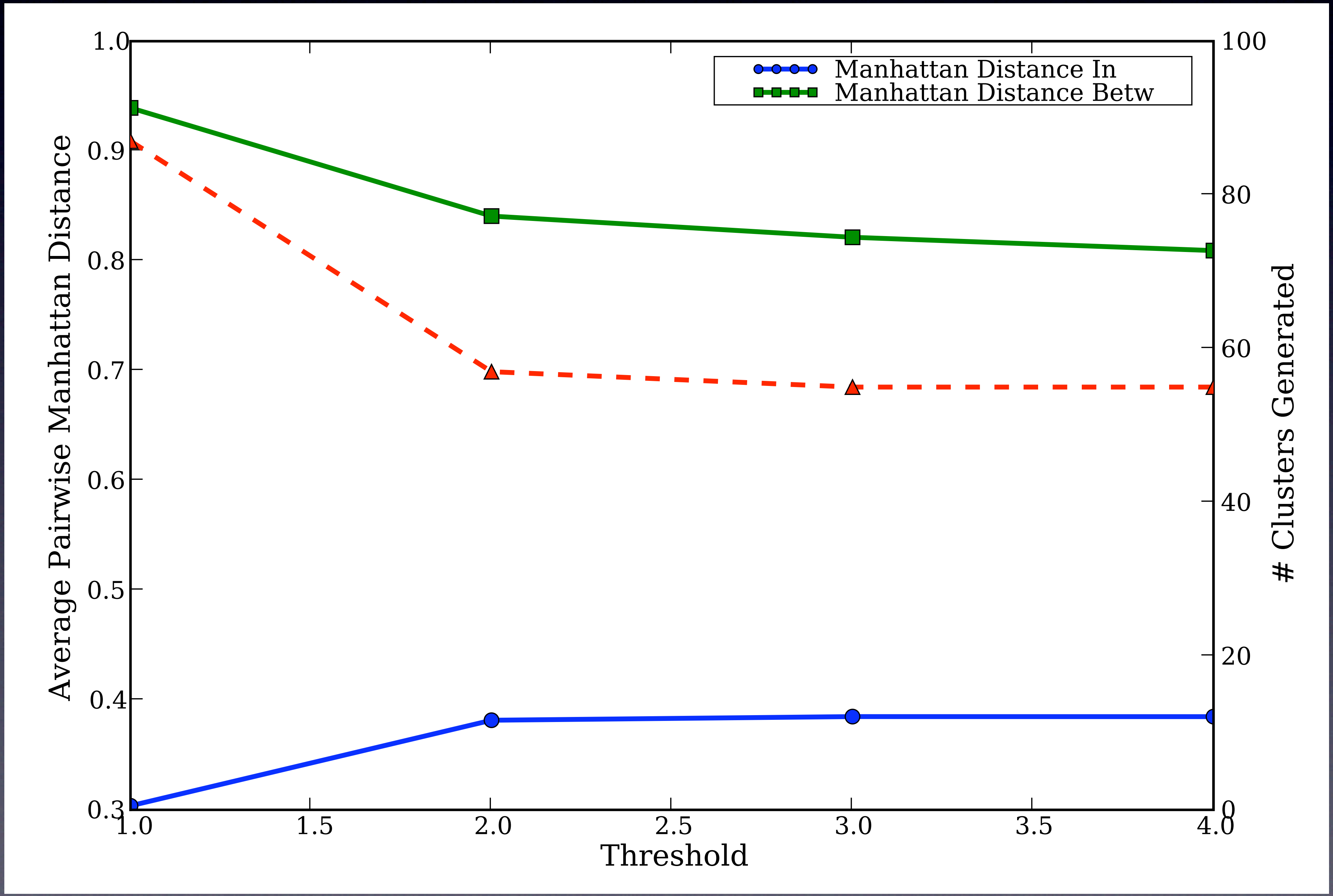


Normal HTTP traffic, threshold = 4

CRII traffic, threshold = 4

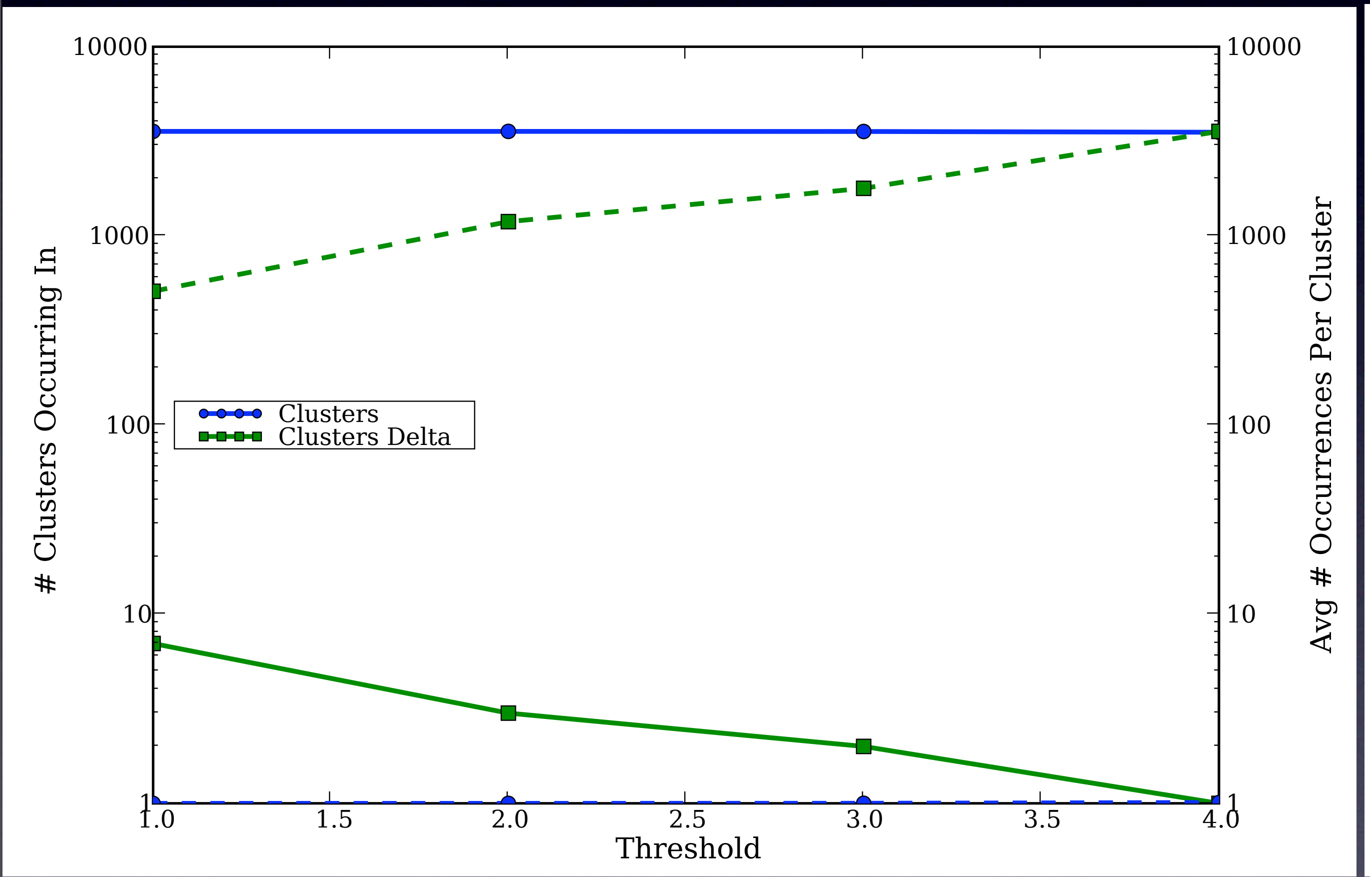
Z-String Clustering

Manhattan Distance

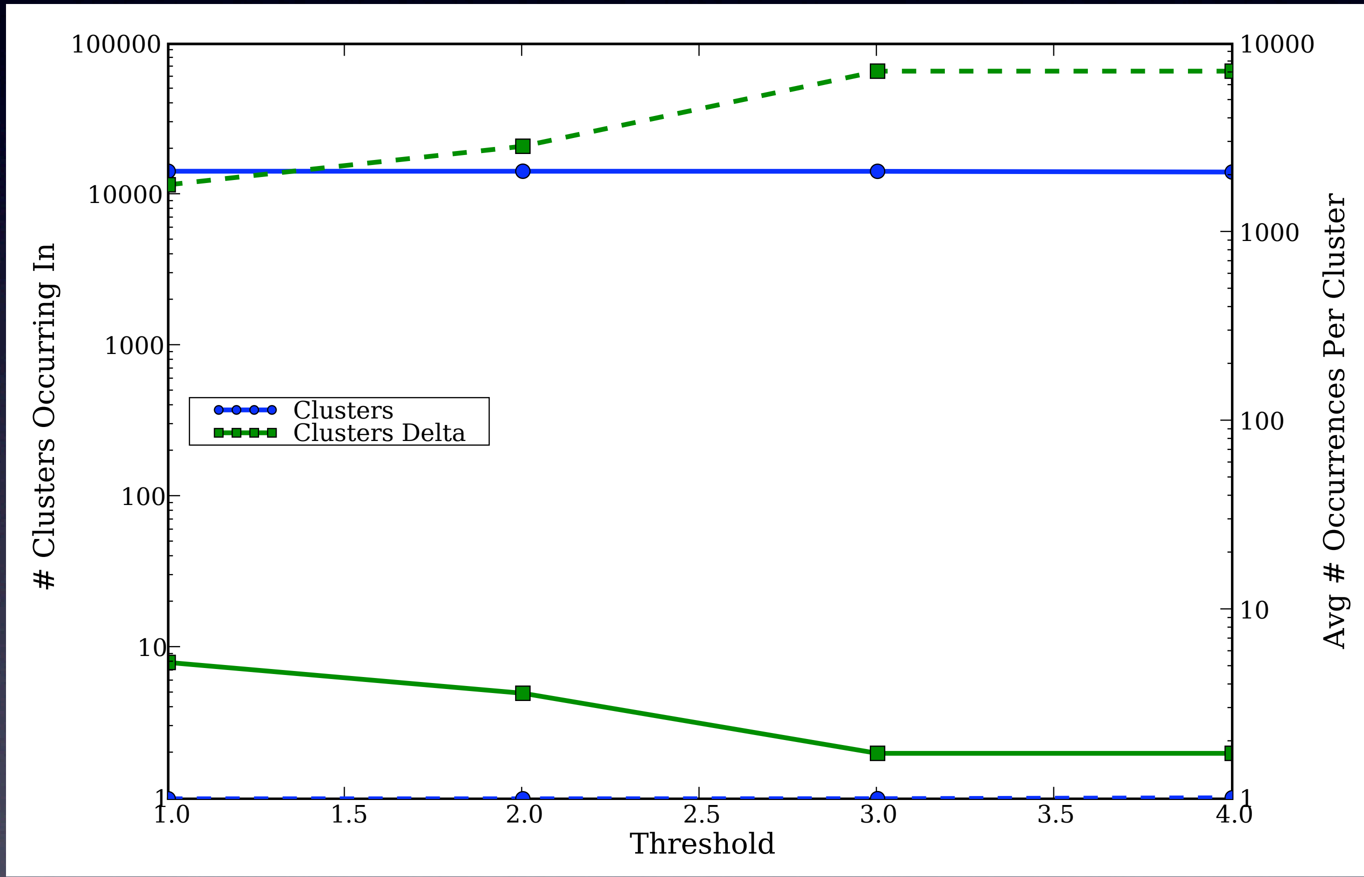


Z-String Clustering

CRII Prevalence



25,000 packets



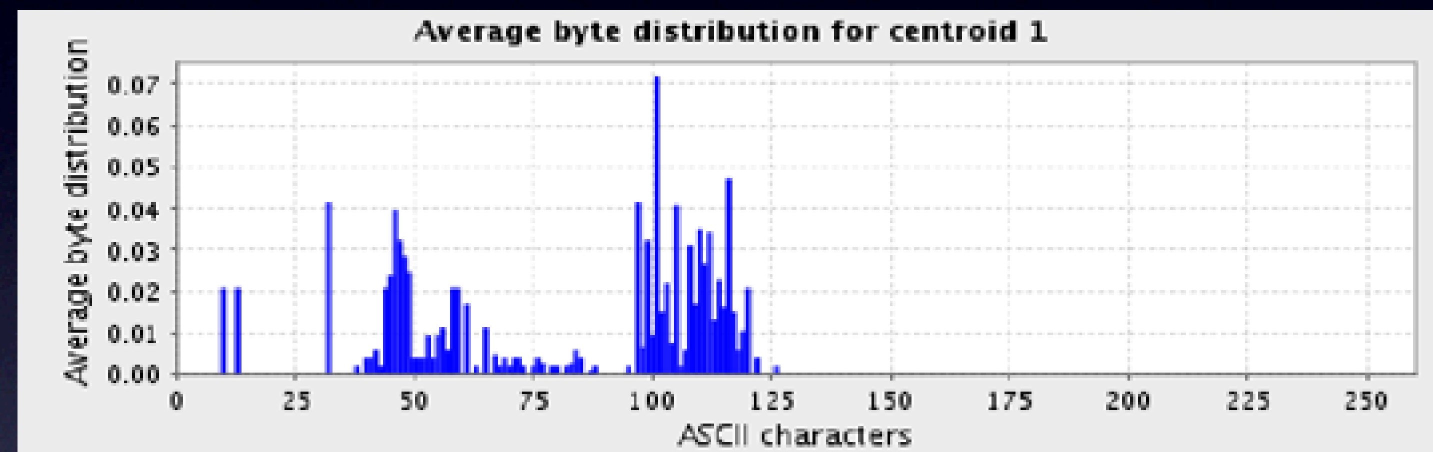
100,000 packets

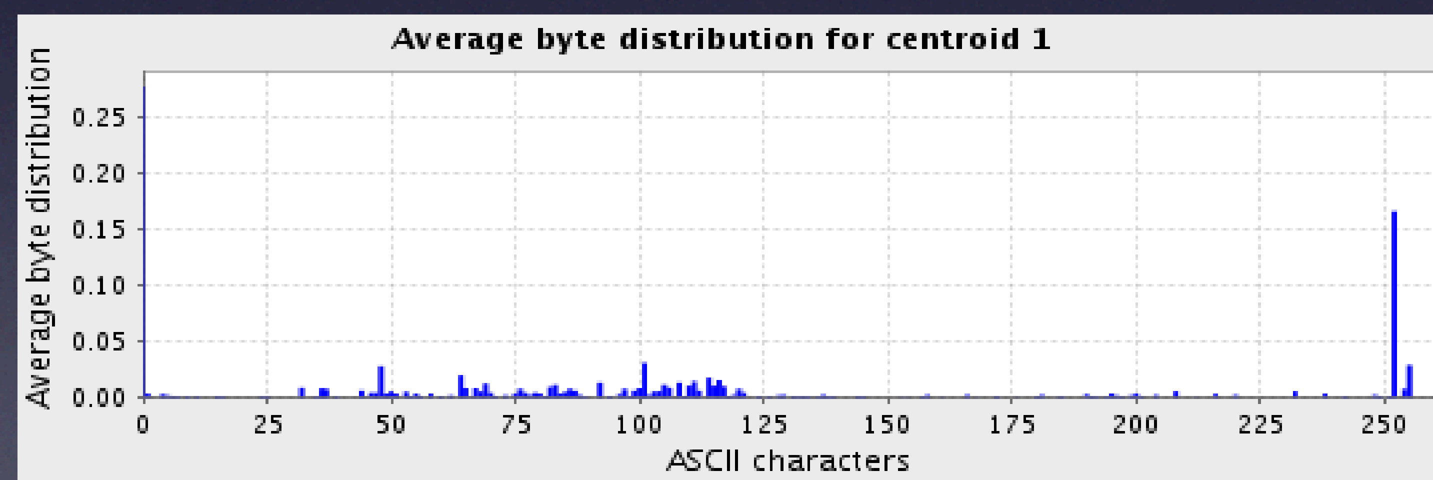
Model corroboration

- Exchange and corroborate/combine the *models* themselves, instead of individual alerts
  - Corroboration of models → comparison of *traffic patterns*
- Leverage privacy-preserving properties of models
- Useful in ad-hoc communications, e.g., MANET
- Key question: do different traffic patterns differ?

Model experiments

- Four models: #1 and #2 simple, #3 "more complex", and #4 primarily malcode
- Examine Manhattan distances, alert incidence between models





Example centroids for models #1 and #4

Model distance

Manhattan distance

	model1 model2	*model1 model3*	*model1 model4*	*model3 model4*
Dist between payload lengths	0.4210	1.5201	1.8981	0.7898
Avg dist over first centroids	0.5946	0.7400	1.6368	1.6330
Avg dist over all centroids	0.4276	0.6112	1.5220	1.5096

Alert indicidence

model1 and model2

Total # packets	# Content Packets	Model 1 #Alerts	Model 2 #Alerts	Model 1+2 #Alerts	Model 3 #Alerts	Model 1+3 #Alerts
127023	10414	149	184	149	81	148
304182	21812	2705	2829	2672	1789	2613
276332	26294	9684	11128	9669	1138	9530
353897	36780	11201	3394	2187	2919	11040

Related work: Event Correlation, Event Systems

- Temporal event correlation/aggregation supporting arbitrary event types
  - Rapide [Luckham96]: focus on software architecture simulation, monitoring
  - SMARTS InCharge/DECS [Yemini96]: primarily network, distributed application management
- Publish/subscribe content-based routing systems providing simple event filtering/covering
  - ELVIN [Segall00]: simple single-message predicate matching
  - Siena [Carzaniga00]: adds minimal support for sequence matching
  - Gryphon [Banavar99]: event stream "interpretation" to reduce transmission overhead

Related work: Distributed Intrusion Detection (DIDS)

DIDS/CIDS: Distributed/Collaborative Intrusion Detection System, multiple networks and sensor(s) at each network

- GrIDS [Staniford96]: Graph hierarchy-based aggregation, with centralized monitoring server
- EMERALD [Porras97]: Distributed, component-based intrusion monitoring
- Quicksand [Kruegel02]: Completely decentralized, specification language to specify patterns
- Indra [Janakiraman03]: Uses "pub-sub-on-P2P" infrastructure
- DShield (Ullman, http://www.dshield.org): Volunteer DIDS
- DOMINO [Yegneswaran04]: Decentralized hierarchy with summary exchange; aggregate analysis of DShield logs

Related work: Privacy-Preserving Collaboration

- Corroboration most commonly implemented using set membership algorithms/tests
  - HotItem protocols [Kissner05]: Uses a Bloom filter implicitly; discusses theoretical capability to maintain "data" and "owner" privacy amongst malicious entities
- Hybrid approaches including hashing/set membership, randomized routing
  - [Lincoln04]: Hashing to scrub sensitive data, second key-based hash algorithm adds "noise" to prevent brute-force attacks
  - Friends Troubleshooting Network [Huang05]: build a recursive lookup P2P network that maintains anonymity; uses hashing, SMC, and random-walk routing for software diagnosis

Related work: Other Privacy-Preserving Computation

- Statistical transformation: useful for larger data exchange where such "summaries" are accurate
  - PAYL [Wang05]: 1-gram and Zipf frequency distributions of packet content
  - Anagram [Wang06]: N-gram binary modeling based on BFs
- Databases and data mining
  - Statistical databases ([Agrawal00], [Lindell02): Aggregate statistics despite perturbation and individual restrictions
  - Privacy-preserving information sharing [Agrawal03]: Two-party equijoin, intersection, counts via commutative encryption
  - K-anonymity [Sweeney02]: Privacy via redundancy
  - Privacy-preserving BF-enabled queries [Bellovin04], secure indices [Bawa03, Goh04]
  - "Hippocratic databases" [Agrawal02]

Related work: Other Privacy-Preserving Computation

- Secure multiparty communication [Yao82]
  - [Du01] proposes general transformation architecture, including intrusion detection information; too slow to handle near real-time alert streams
- Zero-Knowledge Proofs [Goldwasser89, Goldreich94]
  - Like data-mining, traditionally between two parties; scaling up is extremely hard, and may leak information
  - [Dwork04] proposes a model for scaling, but requires clever timing constraints
  - Like SMC, doesn't scale to the event volumes discussed here