Let a Thousand Filters Bloom:

privacy-preserving long-term collection of DNS queries

IETF 101, London, UK

Roland van Rijswijk-Deij

SURF NET

Introduction

- Privacy of DNS traffic between client and resolver rightly is a concern
- DNS-over-TLS goes a long way to protecting this privacy for queries in flight
- Resolver operators, however, can still observe and log traffic
- And may have legitimate reasons to do so, for example, for security reasons (detecting indicators of compromise)

Goal

- Privacy is a strongly held value at SURFnet
- Yet we also need to ensure the security of our network and the users on it
- Simply logging DNS queries on our resolvers is unacceptable
- So we asked ourselves:

How can we detect if certain DNS queries were performed, while respecting the privacy of users?

Approach

- We worked with Dutch security company Quarantainenet to develop a possible solution
- We want to use Bloom filters as a privacy-preserving means to record all DNS queries
- The rest of this talk explains what Bloom filters are, how we plan to use them, and what we are currently doing to study this

What is a Bloom filter?

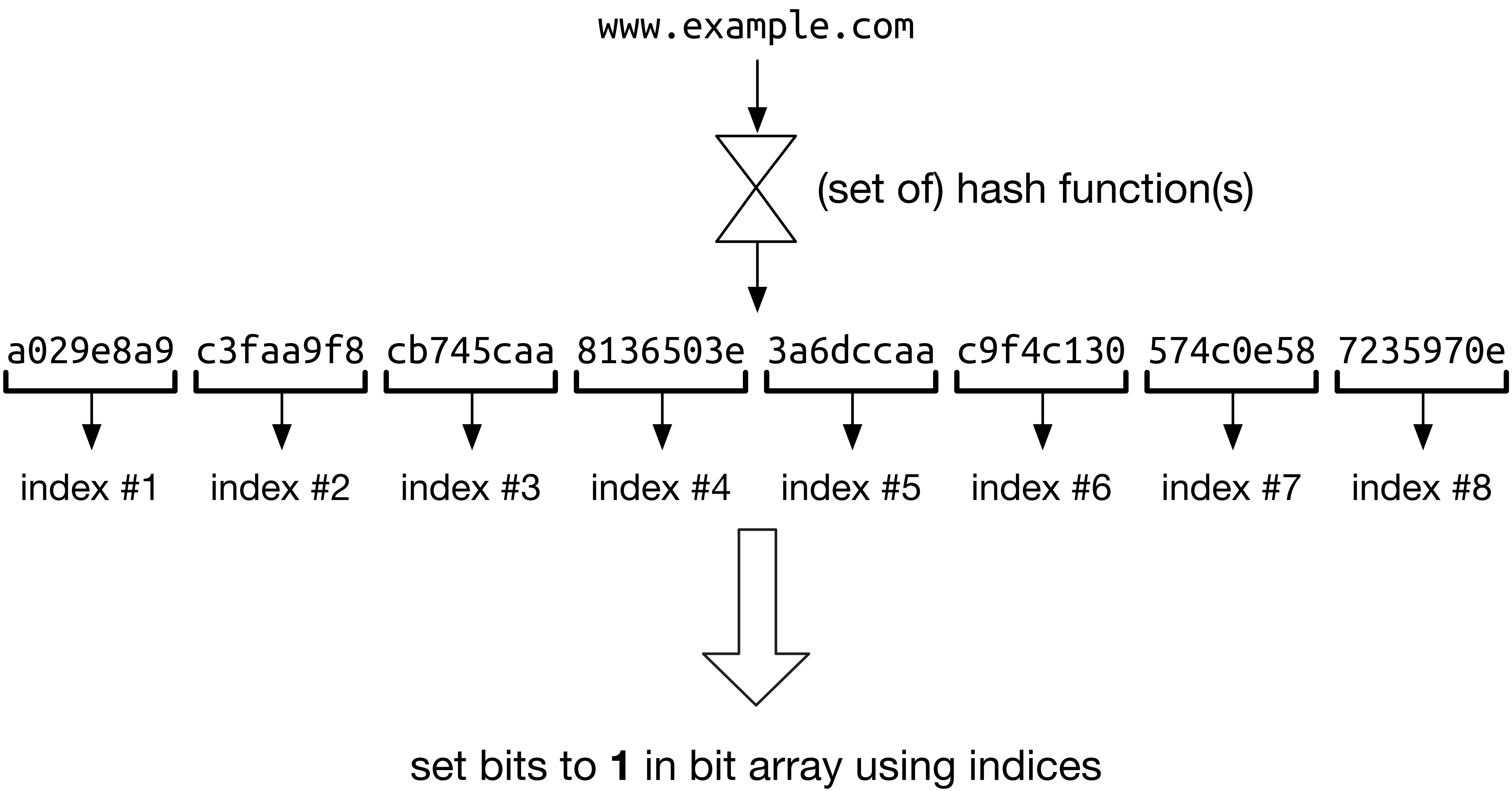
- Bloom filters were originally designed in 1970 as a space-efficient way to optimise indexing of data
- Think of Bloom filters as an unordered set of unique elements with probabilistic membership tests
- For a Bloom filter B and an element n, if we test membership:

$n \in B?$

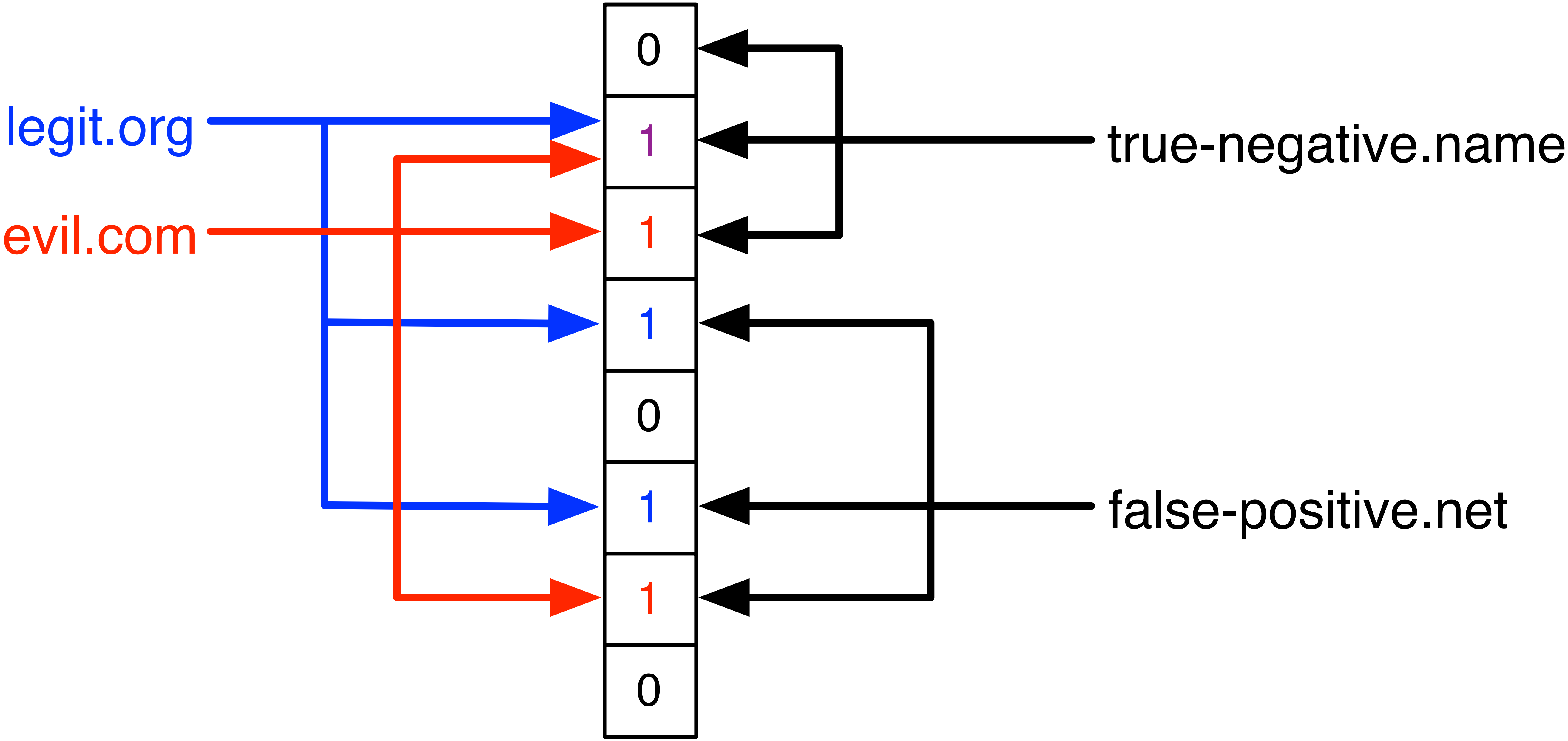
no → n is guaranteed not to be in B

yes → n is highly likely in B, with a small probability p_ε of this being a false positive

Bloom filter in pictures



Bloom filter in pictures



(image courtesy of Quarantainenet)

Bloom filter parameters

- Tune to achieve a certain (low) false positive rate
- Parameters:
  - Size of bit array
  - Number of hash functions → number of indexes
  - Expected number of distinct elements
- Performance influenced by number of distinct items entered into the filter; the formula below approximates the probability of a false positive p_ε:

$$p_\epsilon \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$$

Privacy properties

- Filters do not store original query names and are non-enumerable; lookup only possible if you know exactly what you are looking for

- By mixing queries from multiple users in a single filter, tracking individual users becomes even hard(er)

- We can combine that state of filters with the same parameters into a new, aggregated filter (with possibly a higher false positive, but also more users in the same filter)

Other considerations

- Privacy risk: if I know a query that unambiguously identifies a certain user (e.g. name of personal server), I can still track them, but hard to correlate with other queries if more than one user in the filter
- Bloom filters have additional benefits:
  - Space efficient (filters have a fixed, reasonable size)
  - Time efficient (lookups are fast)

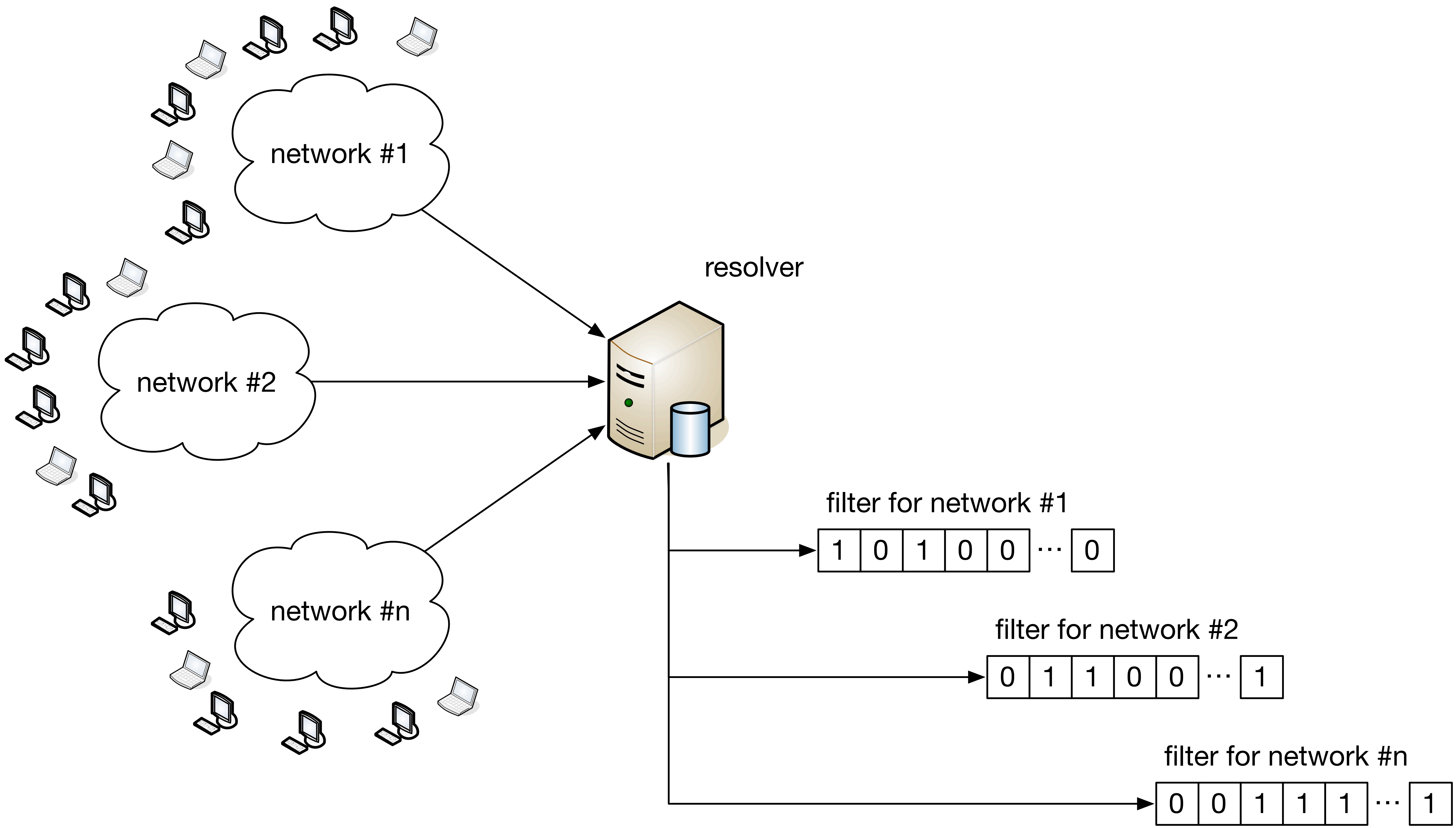
Grouping users into filters

- One of our challenges is how we will group users into filters

- Current thinking: group all users in certain prefixes that belong to specific connected institutions on our network (e.g. universities, teaching hospitals, ...)

- Open question: how many users should we combine in a filter? Does that matter very much for privacy? (since filters cannot be enumerated)

Grouping users into filters



Work in Progress

- We have a master student working on testing the use of Bloom filters for detection of indicators-of-compromise (IoCs) in DNS queries

- His main focus:

  - What IoCs can we detect using this approach, but also: what can't we detect?

  - Designing an architecture for filling and querying filters (e.g. how do we group users, how do we store and query filters?)

Work in Progress

- We will deploy Unbound with Bloom filter integration on SURFnet's production resolver infrastructure

- Relatively busy resolvers (order of 5-10k queries per second), that between them see roughly 150-200k unique client IPs per day

- Ideally, we want to group by customer, challenge: we have ±200 customers

- Goal is also to see how well all of this scales

Use Cases

- The master student will look at three use cases in particular:

1. Detection of (high value) IoCs that we receive from the Dutch National Detection Network (IoCs received from, a.o., intelligence agencies)

2. Detection of queries for "DDoS-as-a-Service" providers (aka Booters/Stressers)

3. Analysis of blacklist hits from our e-mail filtering service

Open source

- Bloom filter library we use developed as open source by Quarantainenet, funded by SURFnet (BSD 3-clause license)
- SURFnet also provided funding for integration in Unbound (will be DNSTAP), NLnet Labs is working on this
- Expecting to release code somewhere this year, no definitive data yet

Conclusions

- We set out to find a privacy-conscious way to collect information on DNS queries, with the goal of looking for certain queries for security purposes

- In collaboration with Quarantainenet and NLnet Labs, we are implementing a solution based on Bloom filters, that will be released in open source

- We currently have a master student integrating this and testing this in our DNS production environment

Thank you for your attention!
Questions?

nl.linkedin.com/in/rolandvanrijswijk

@reseauxsansfil

roland.vanrijswijk@surfnet.nl

SURF NET