Let a Thousand Filters Bloom:

privacy-preserving long-term collection of DNS queries

Roland van Rijswijk-Deij*, Matthijs Bomhoff†, Ralph Dolmans‡

*SURFnet / †Quarantainenet / ‡NLnet Labs

Introduction

- **Privacy of DNS traffic** between client and resolver currently has **a lot of attention** in the Internet community, e.g.:
  - **DPRIVE** working group in the **IETF**, standardised **DNS-over-TLS**
  - Deployment of **DNS-over-TLS** by e.g. **1.1.1.1**, **8.8.8.8, 9.9.9.9** and others (including SURFnet)
  - Upcoming **DoH standard** (**DNS-over-HTTPS**), which has a big **push from** the **browser community**

- Note the **focus** is **on privacy** of traffic **in-flight**

Elephant in the room

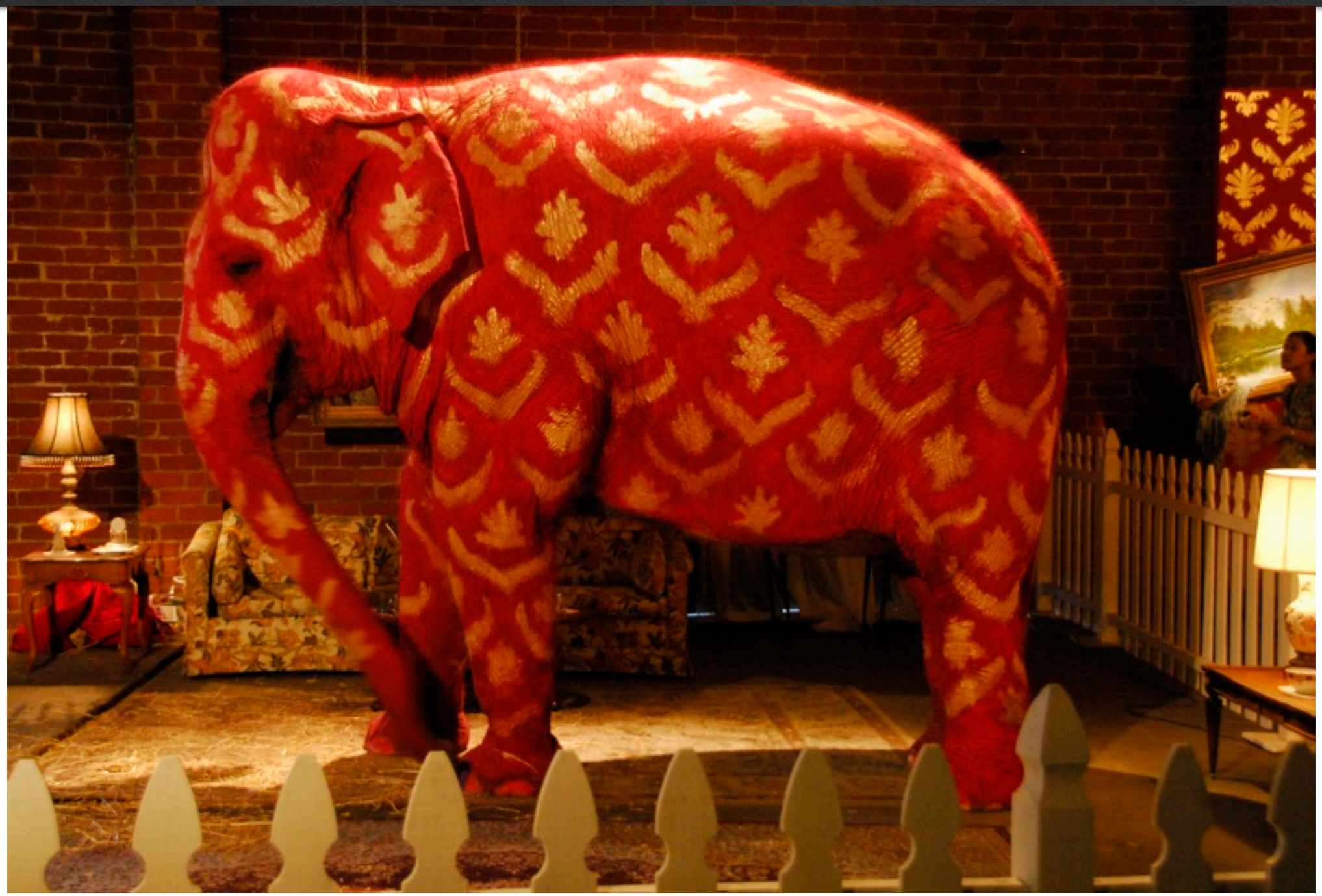


image © Jdcollins13@Flickr

Elephant in the room

- **Resolver operators** can **still observe and collect** DNS query **traffic**

- And they have **legitimate reasons** to do so

- For example: to **detect indicators of compromise** in DNS traffic

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**
- We want to **take strategic and tactical decisions** based on the presence of **DNS queries associated with indicators of compromise**, so we are **not interested in queries per user**
- 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-enhancing technology** to record all DNS queries

- **This talk explains** what **Bloom filters** are, **how we** intend to **use them**, and **what we have learned** so far

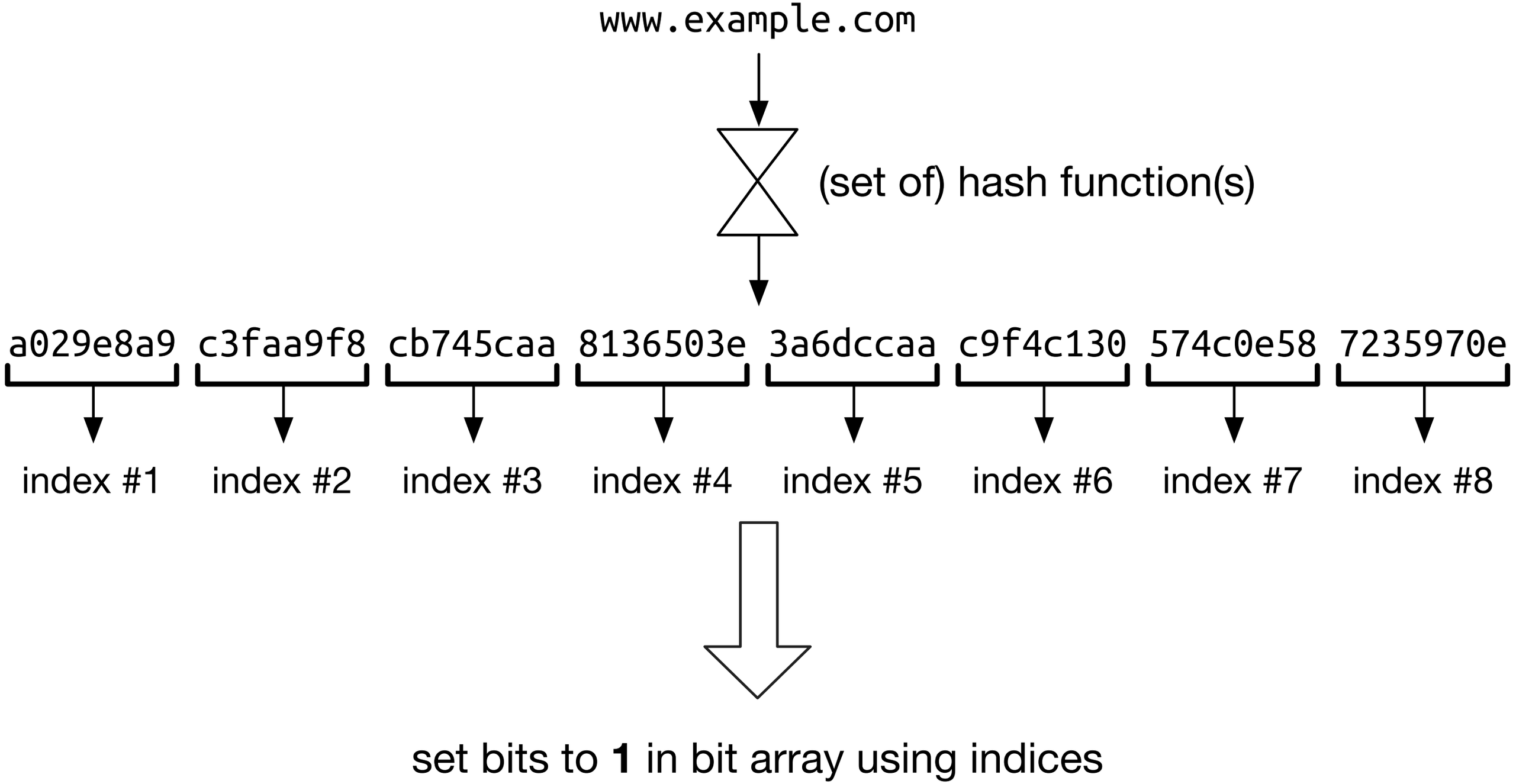
What is a Bloom filter?

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

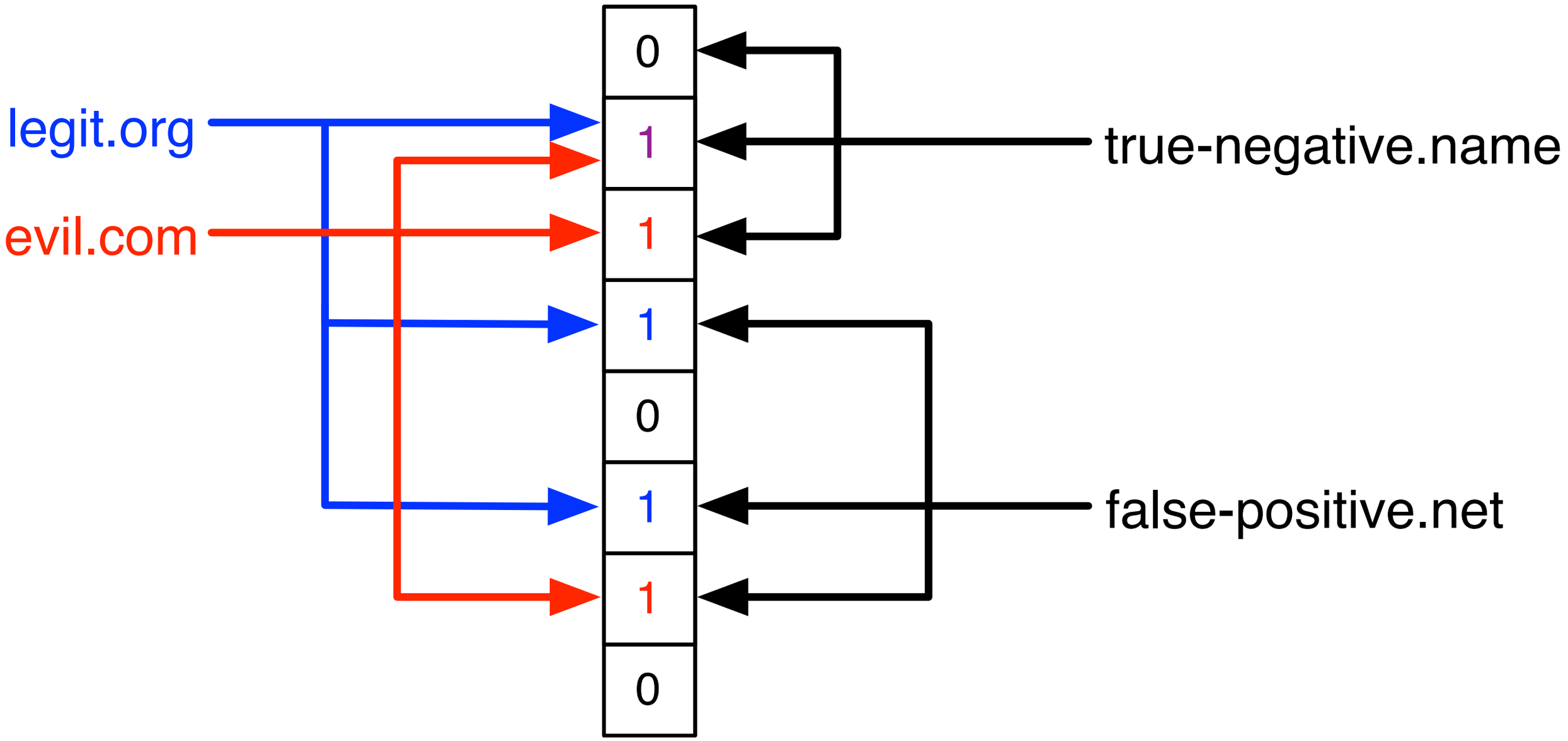
$n \in B?$

- **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

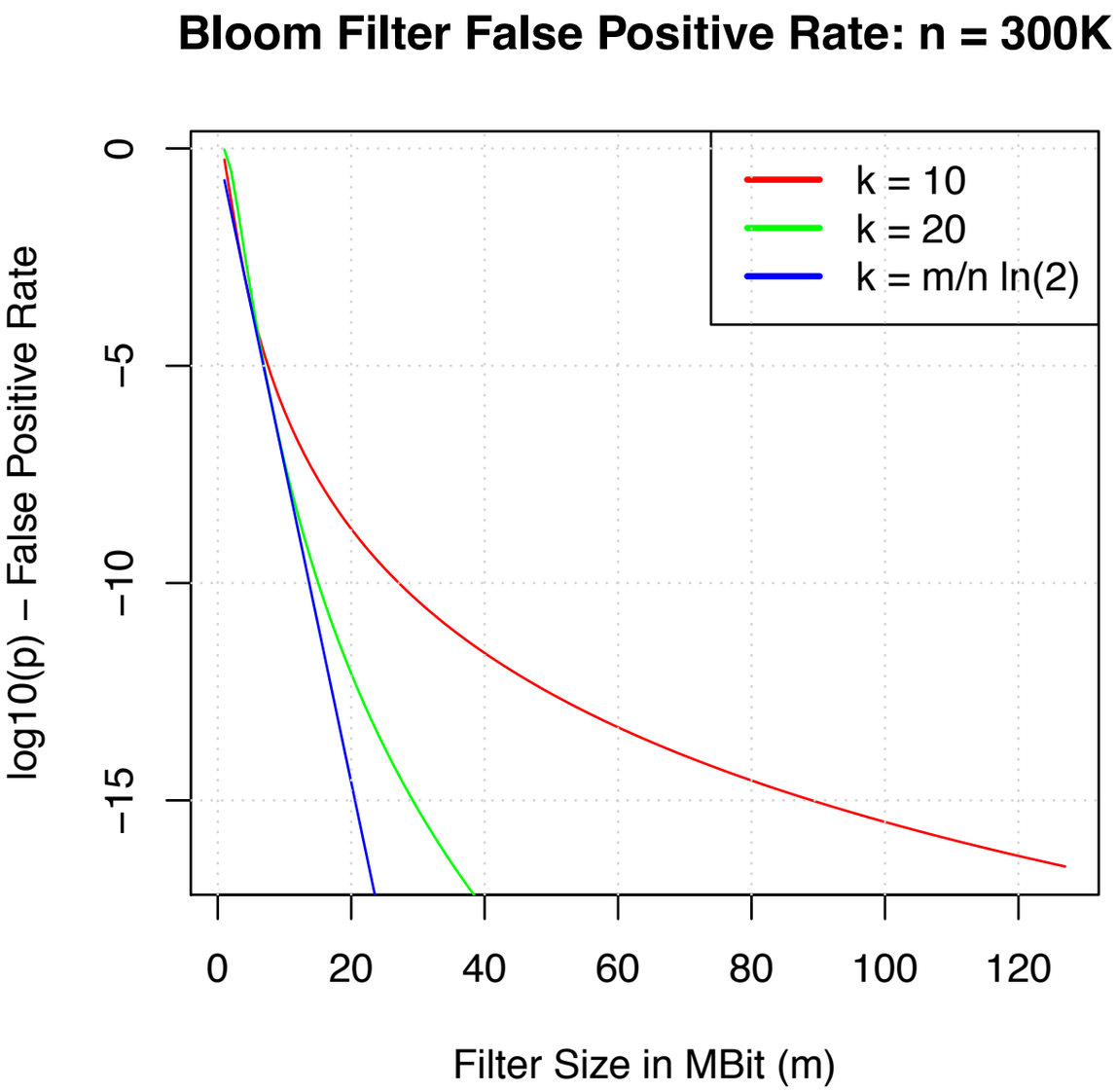
- **Tune** to achieve a certain (low) **false positive rate** at a **reasonable filter size**

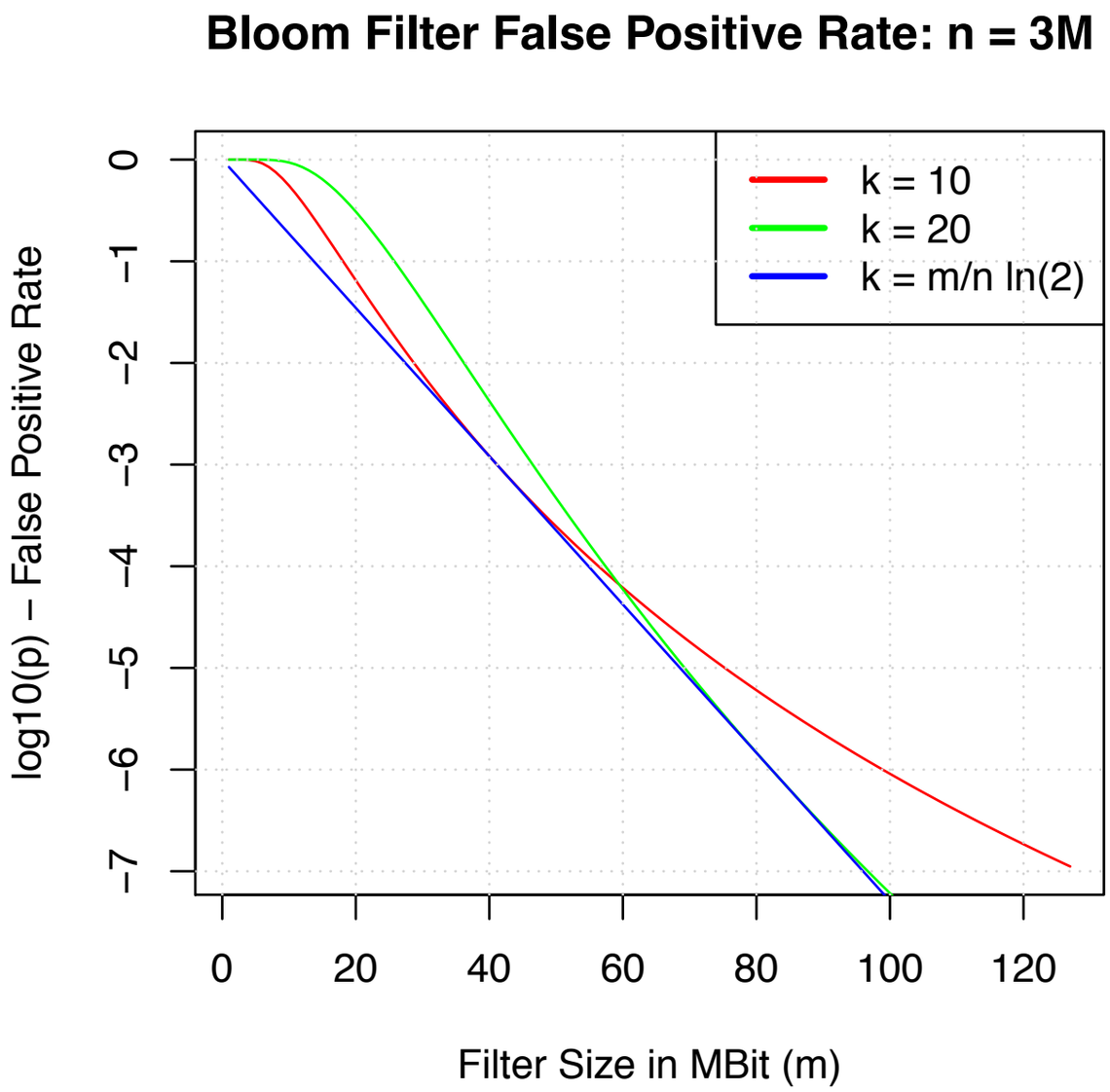
- **Parameters:**
  - Number of hash functions k → number of indices
  - Size of bit array m
  - Expected number of distinct elements n

- The formula below approximates the **probability of a false positive** p_ε:

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

False positive rate





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** the **state of filters** with the same parameters **into** a new, **aggregated filter** (with possibly a higher false positive probability, but data over a longer period and/or for more users combined)

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 **impossible 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)

What to store?

- The **most important** design decision is **what information from a query** to store

- We **considered** the **following** query **attributes**:
  1. **Full query name** (canonicalised)
  2. **Individual labels** in a name (e.g. 'www', 'example', 'com')
  3. Queried **type**
  4. **Response** data

- For the moment, we are **focussing on 1 and 2**

Distributing users over filters

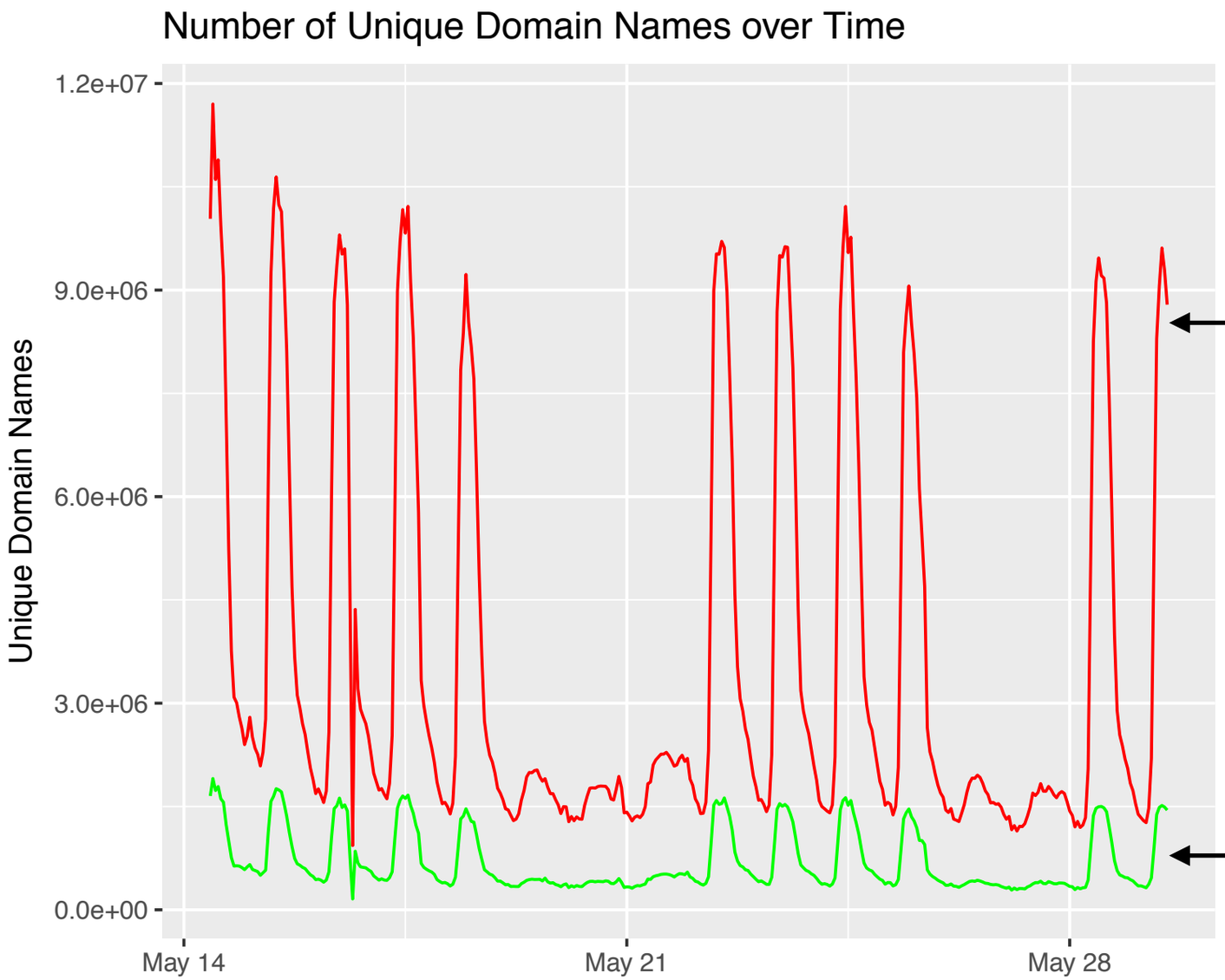
- The **second** important **design decision** is **how to distribute users over filters**

- We want to learn if queries were made by **users from certain institutions** (again, not interested in individuals)

- **Two options:**
  1. **Separate filter** for **each institution**
  2. **One big filter**, and **prepend institution name** to data inserted into filter

Distributing users -- numbers (1)

- Ideally, we want to **collect queries per hour**; so **how many distinct queries** do we get?

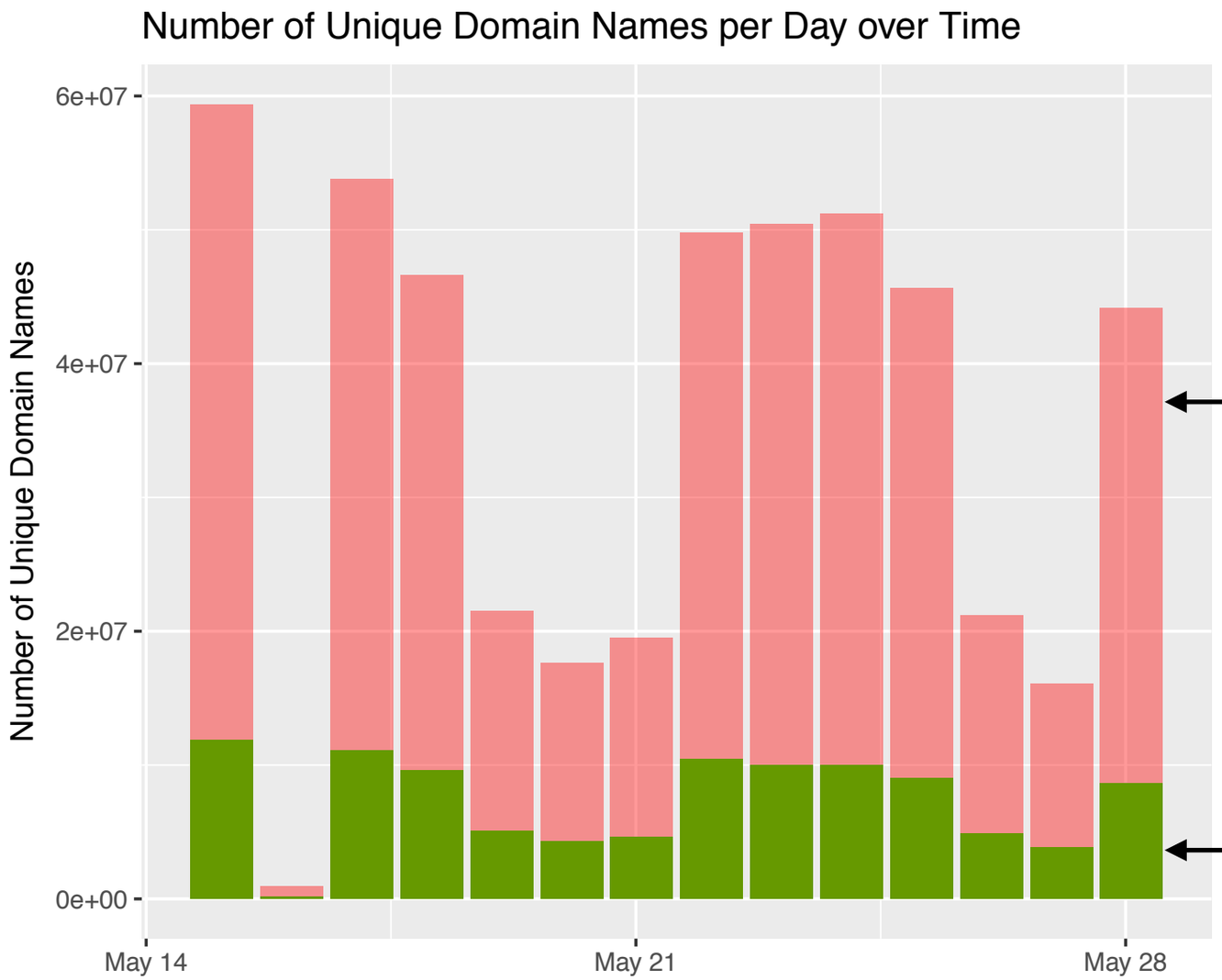


#distinct query names with /16, /20 and /24 prefix prepended

#distinct query names

Distributing users -- numbers (2)

- Also, we would like to **aggregate from hourly to daily** filters, while **maintaining a reasonable false positive rate**



#distinct query names with /16, /20 and /24 prefix prepended

#distinct query names

Work in Progress

- We have a **master student** who is **building** a working **prototype** to test 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?)

Prototype design

- Based on **measurements and experiments**, we decided to use a **single large(r) Bloom filter** to **store all** query **information**

- **Advantages:** (very) **space efficient**, and single set of parameters so **filters can be combined** for aggregation

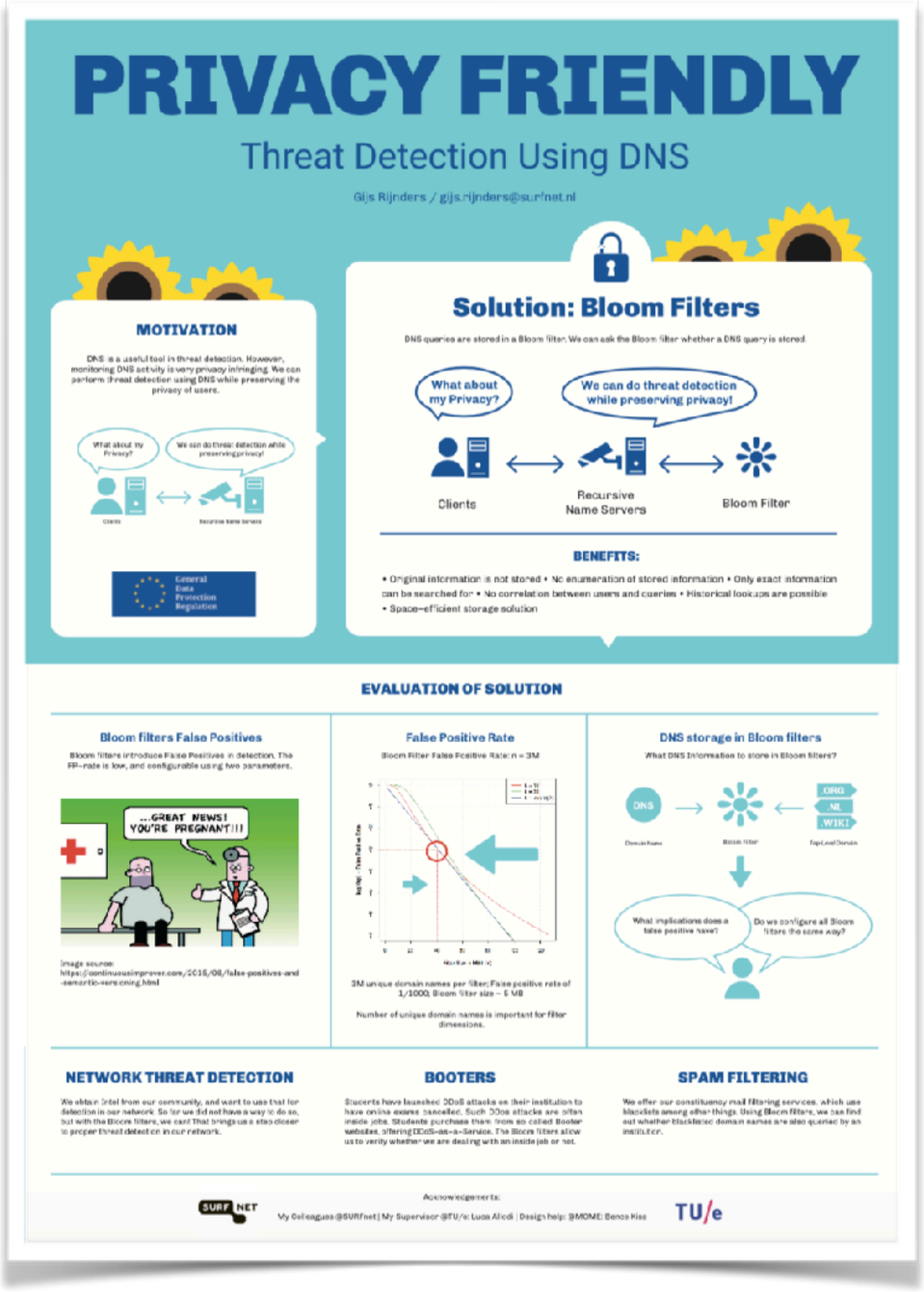
- **Disadvantage:** a **single user can pollute** the filter with random query names and **raise the false positive rate**

- We **will store:**
  - <institution>+<full query name>
  - <institution>+<individual labels from FQDN>
  - <full query name>
  - <individual labels from FQDN>
  - (possibly also <prefix>+<(parts of) FQDN>)

Prototype design

My student, Gijs Rijnders, who is working on our prototype, was one of the winners of the TNC Poster Pursuit, so go see his poster for more info on the prototype!

(also: vote for him 😉)



Testing the prototype

- 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) in **collaboration with NLnet Labs**

- Expecting to **release prototype code** somewhere **this year**, no definitive date yet
(come talk to me if you would like to play with it)

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 **expect to publish results** of our prototype experiments at the **end of this summer** (late August)

Thank you for your attention!
Questions?

acknowledgements: with many thanks to my student, Gijs Rijnders for supplying nice graphs :-)

nl.linkedin.com/in/rolandvanrijswijk

@reseauxsansfil

roland.vanrijswijk@surfnet.nl

SURF NET