ECC + DNSSEC

The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation

UNIVERSITY OF TWENTE.

DACS
Design and Analysis of Communication Systems

SURF NET

Introduction

- In earlier research, we showed:
  - What fraction of DNS resolvers have **problems with fragmentation** [ComMag14]
  - To what extent the then current population of DNSSEC-signed domains can be abused in **amplification attacks** [IMC14]
- Problems are linked to **DNSSEC response size**, due to the **inclusion of signatures** and **keys**
- Arguably, the **root cause** is use of **RSA** as **'default' algorithm**

Solution: ECC?

- Using signature schemes based on **Elliptic Curve Cryptography solves both issues** [CCR15]
    - ECC schemes generally have (much) **smaller keys and signatures**
- So **why not switch** to ECC **immediately**?

Solution: ECC?

As we will see later, this is optimistic…

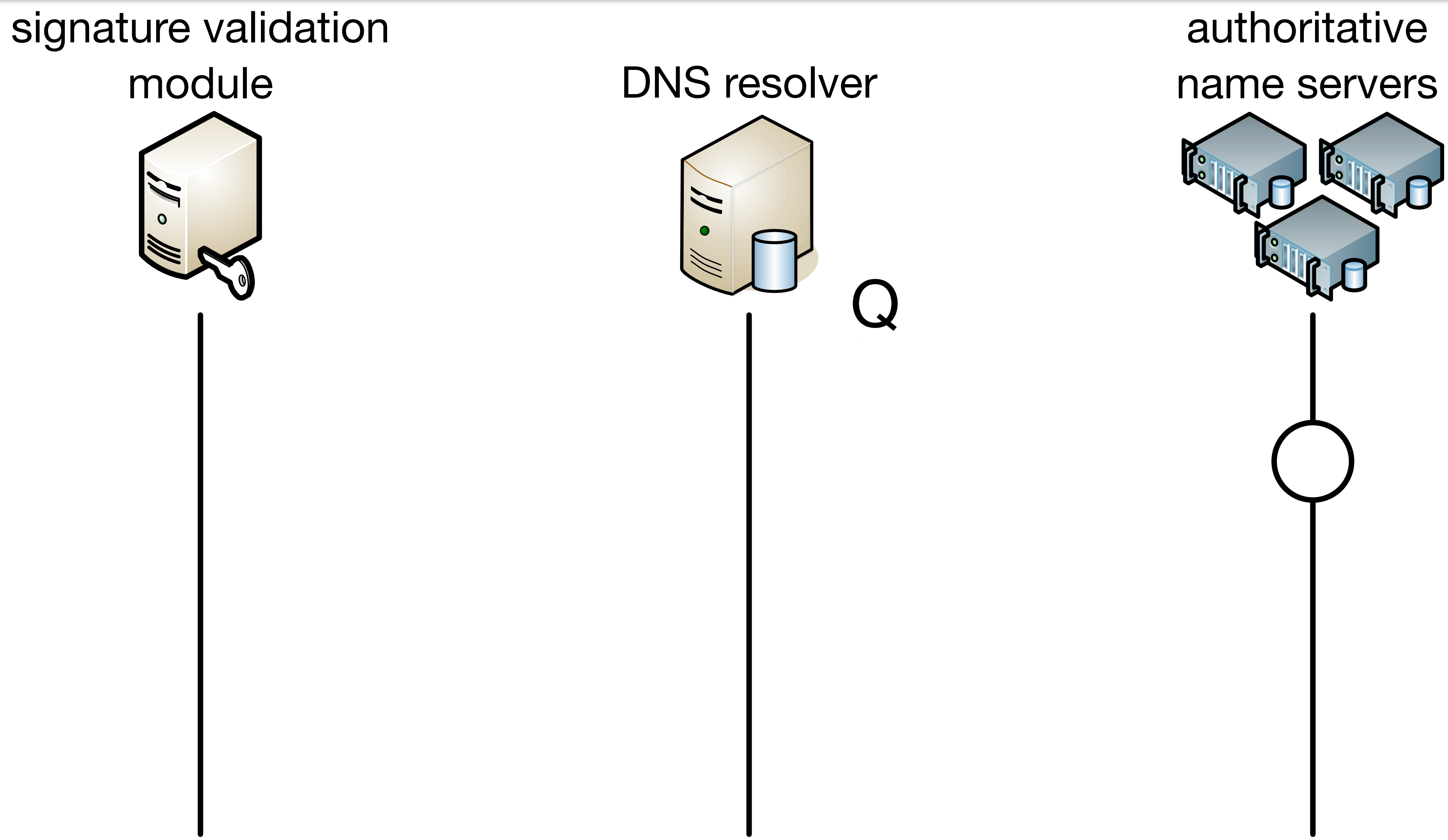
- To quote RFC 6605:

  “[…] validating RSA signatures is significantly faster than validating ECDSA signatures (about 5 times faster in some implementations)”

- This potentially means switching to ECC **pushes problems to the edge** of the network!

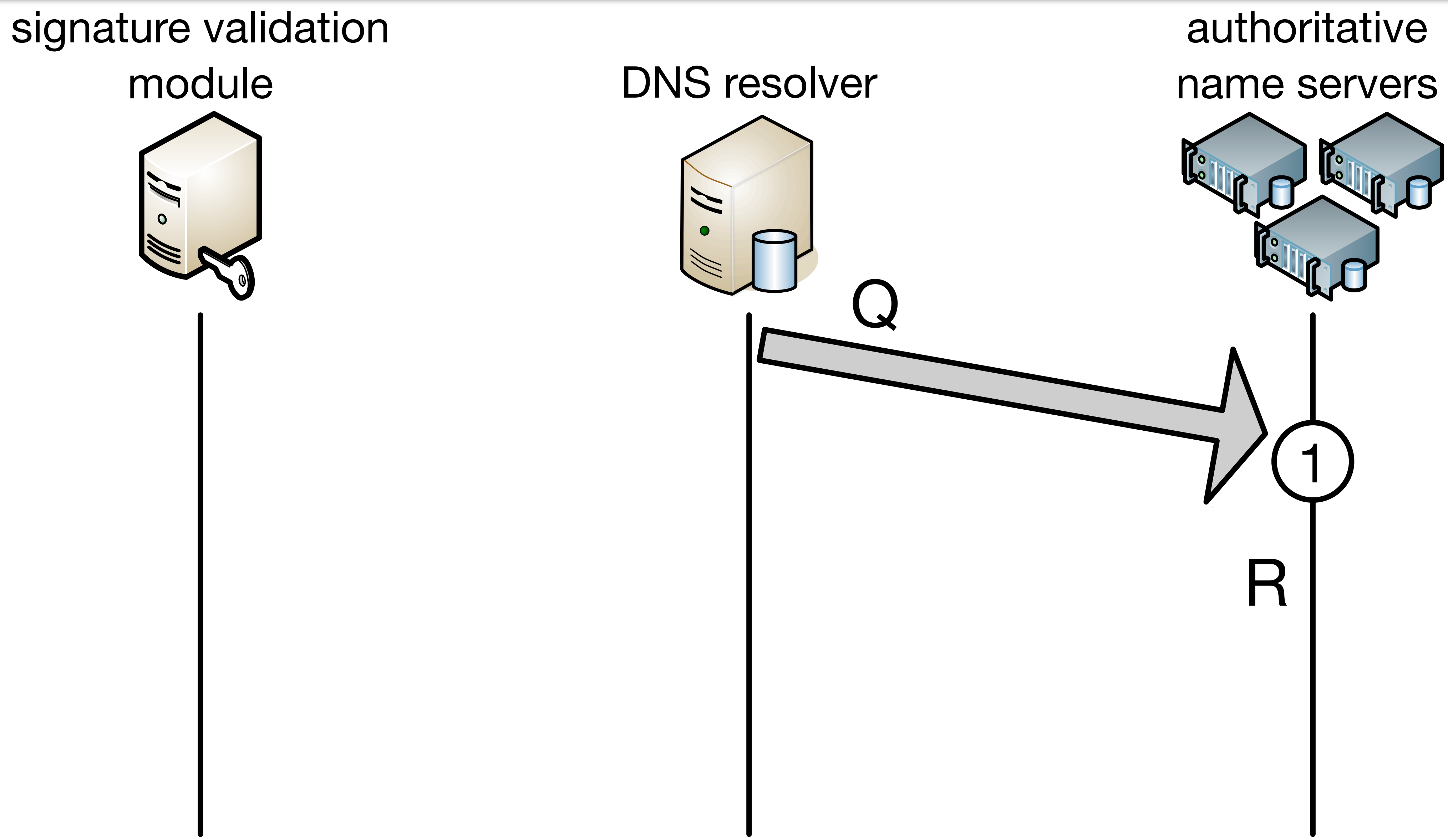
Goal of this study:
How does this impact validating DNS resolvers?

Methodology



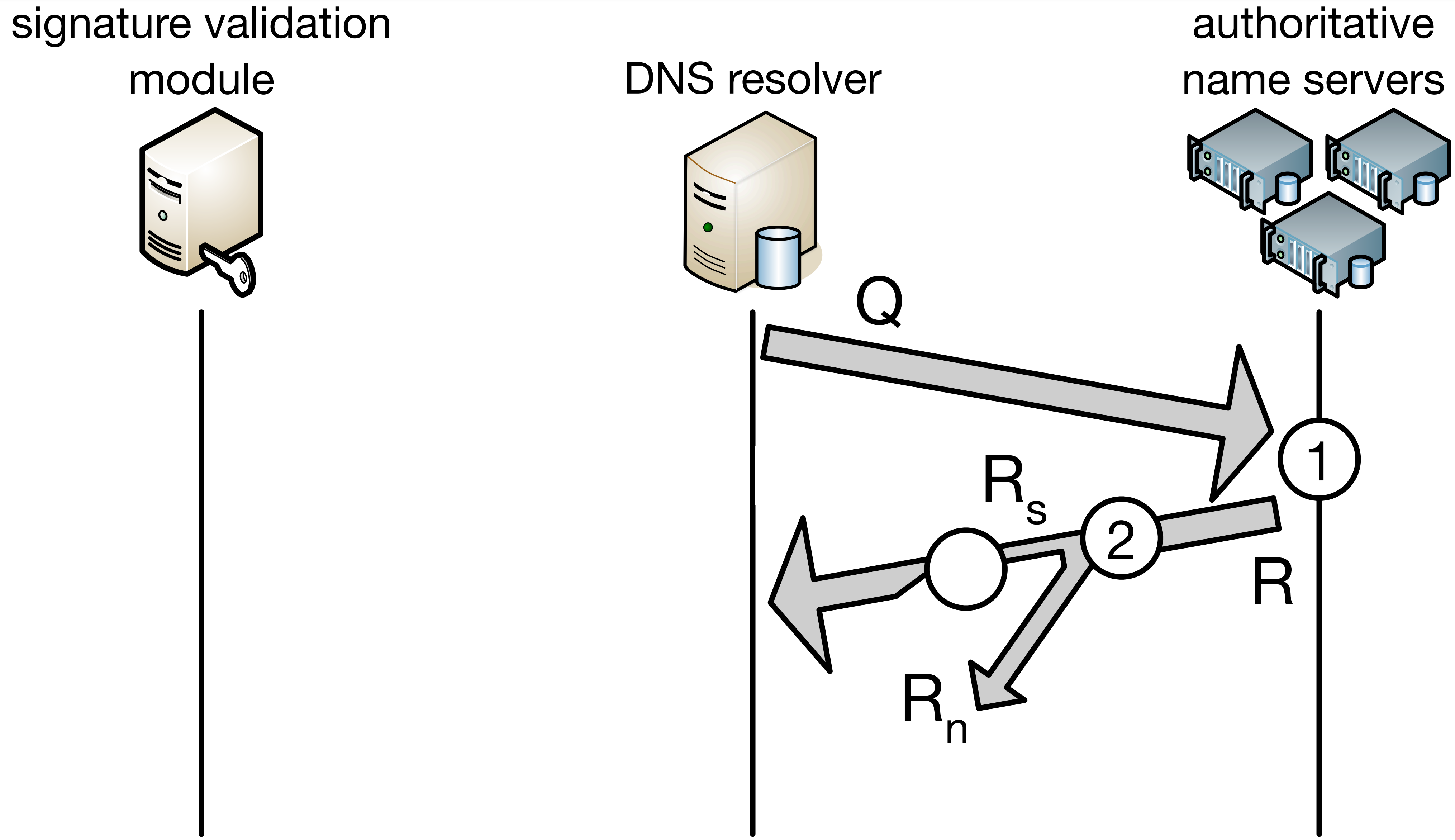
- **Intuition**: we can **predict** the **number of signature validations (S_v) based on** the **number of outgoing queries** from a resolver **(Q)**

Methodology



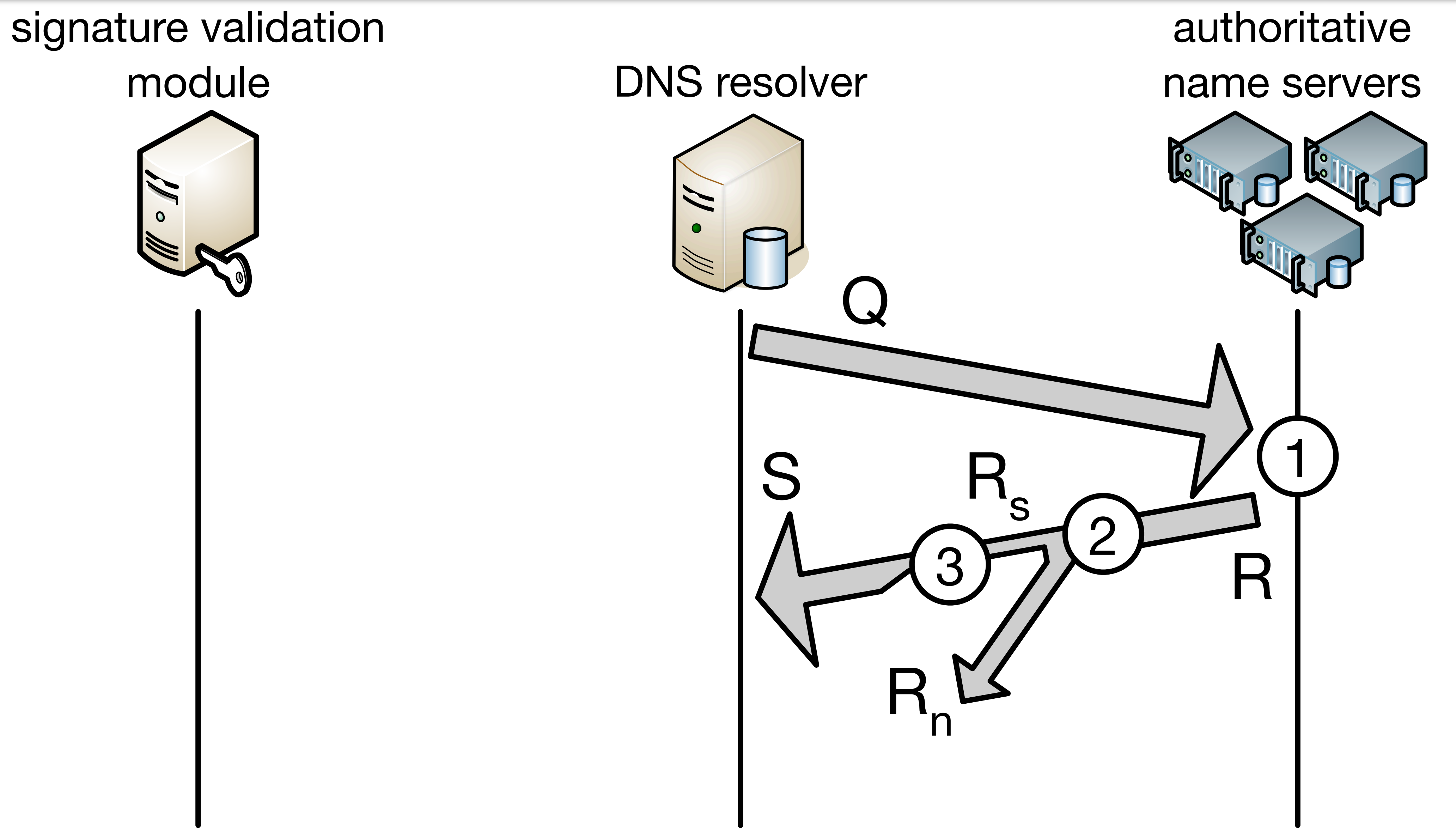
- The resolver will likely not receive a response to every query it sends, therefore we record **(1)** the **the number of queries (Q) and responses (R)**

Methodology



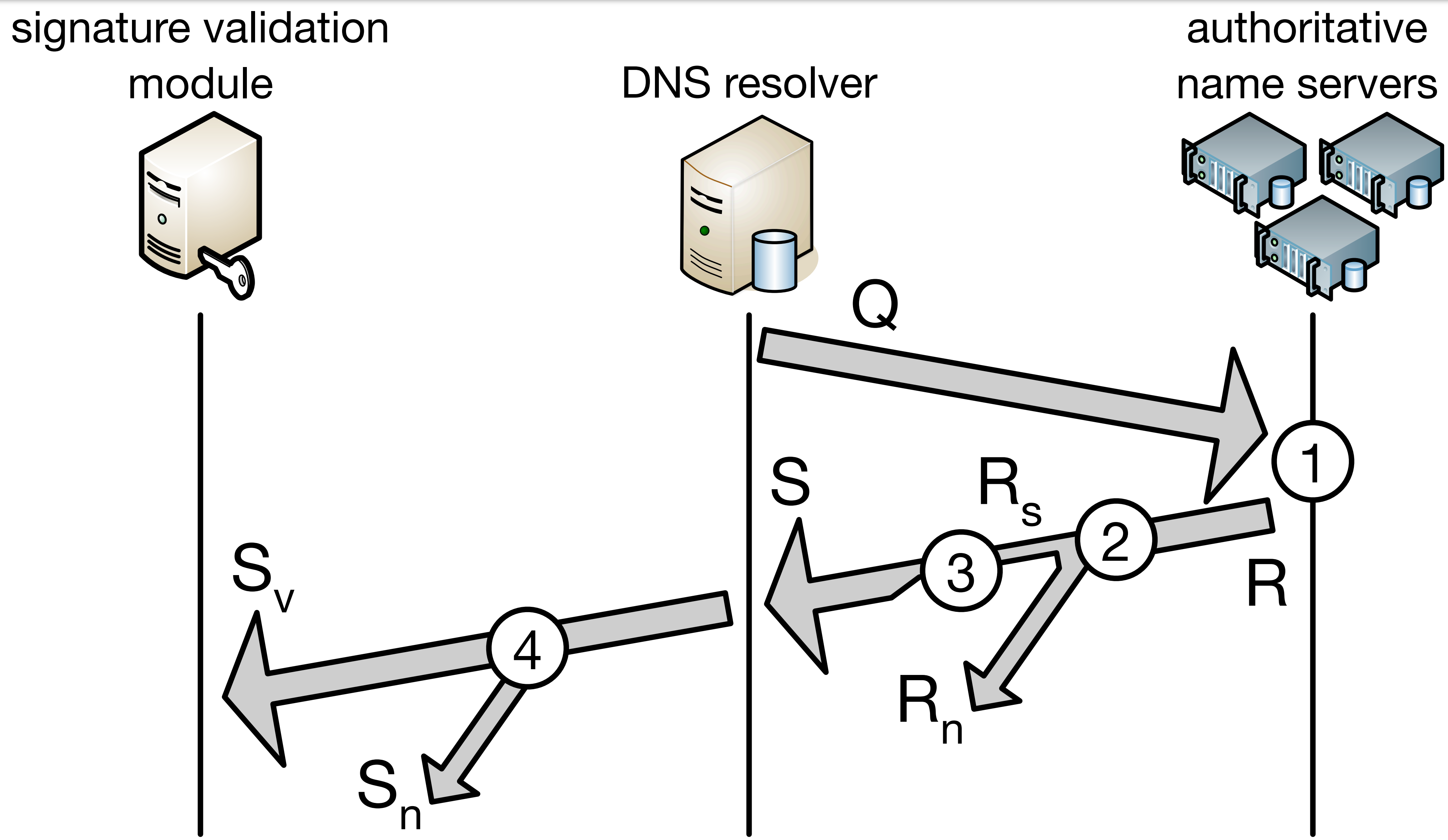
- Not every response contains signatures. Therefore, we record **(2) the number of responses containing signatures (R_s)**

Methodology



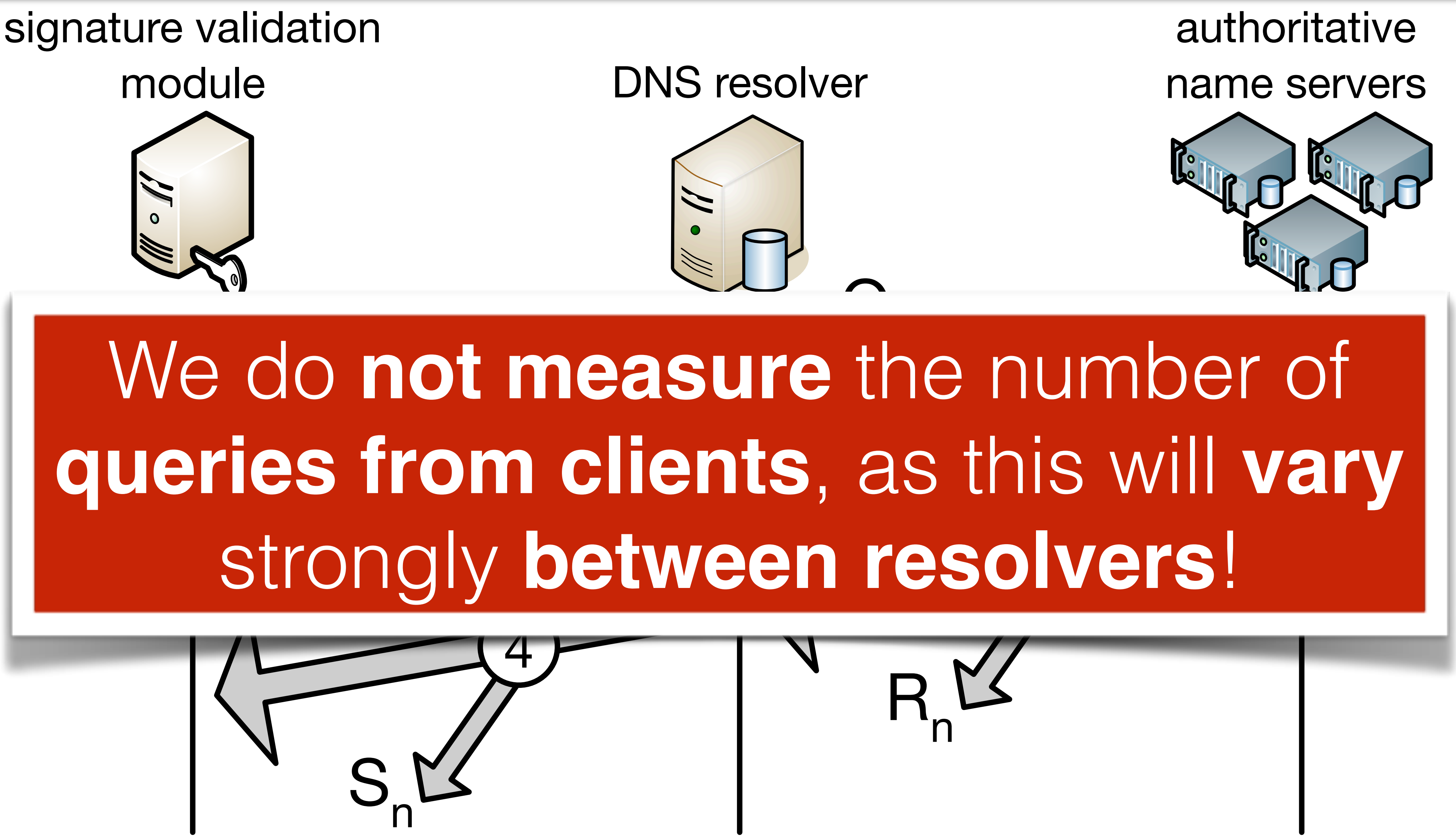
- Not every response contains the same number of signatures, therefore, we record **(3), the number of signatures per response (S)**

Methodology



- Not every signature needs to be validated (e.g. because of caching). Therefore, we record **(4) the number of signatures that are validated (S_v)**

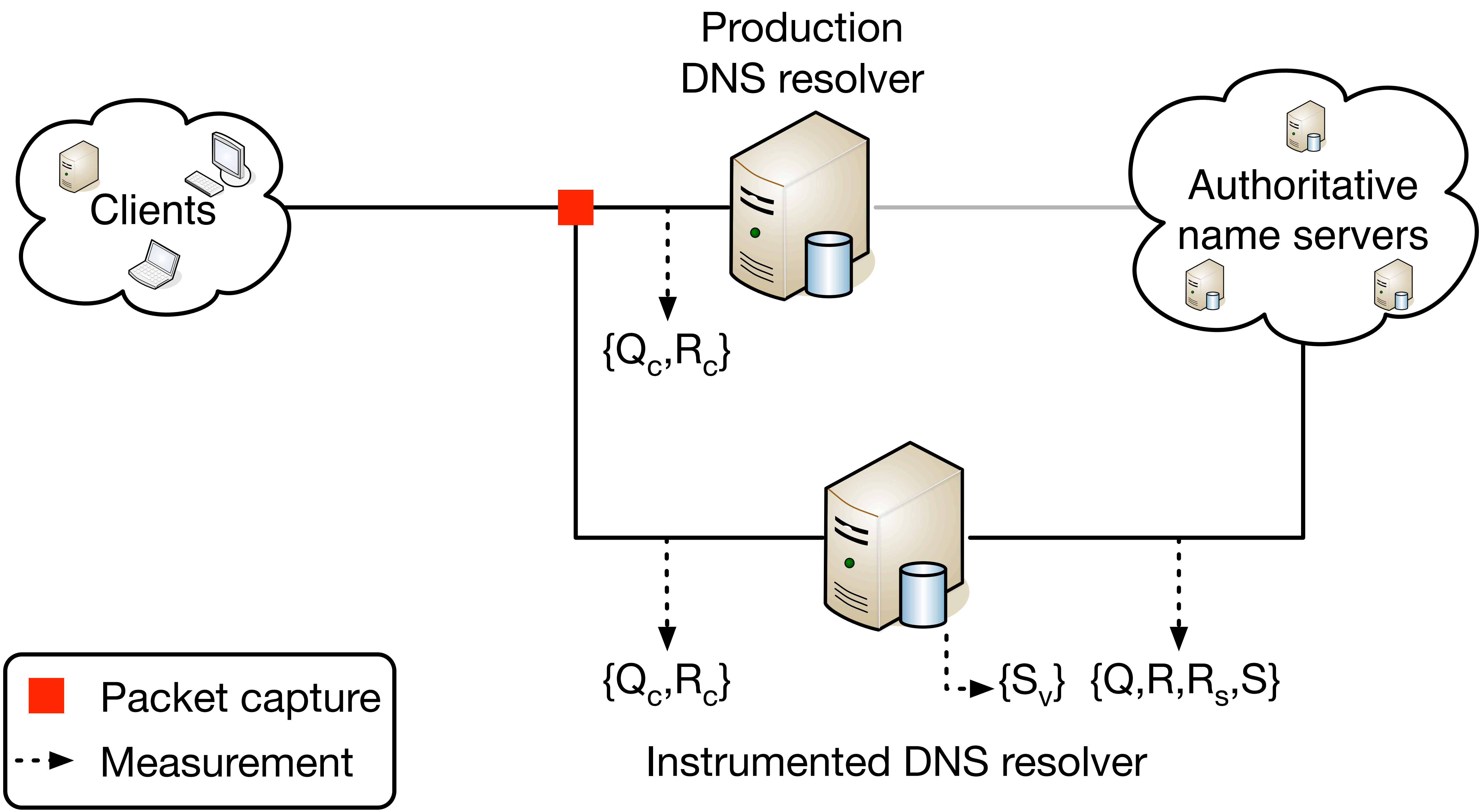
Methodology



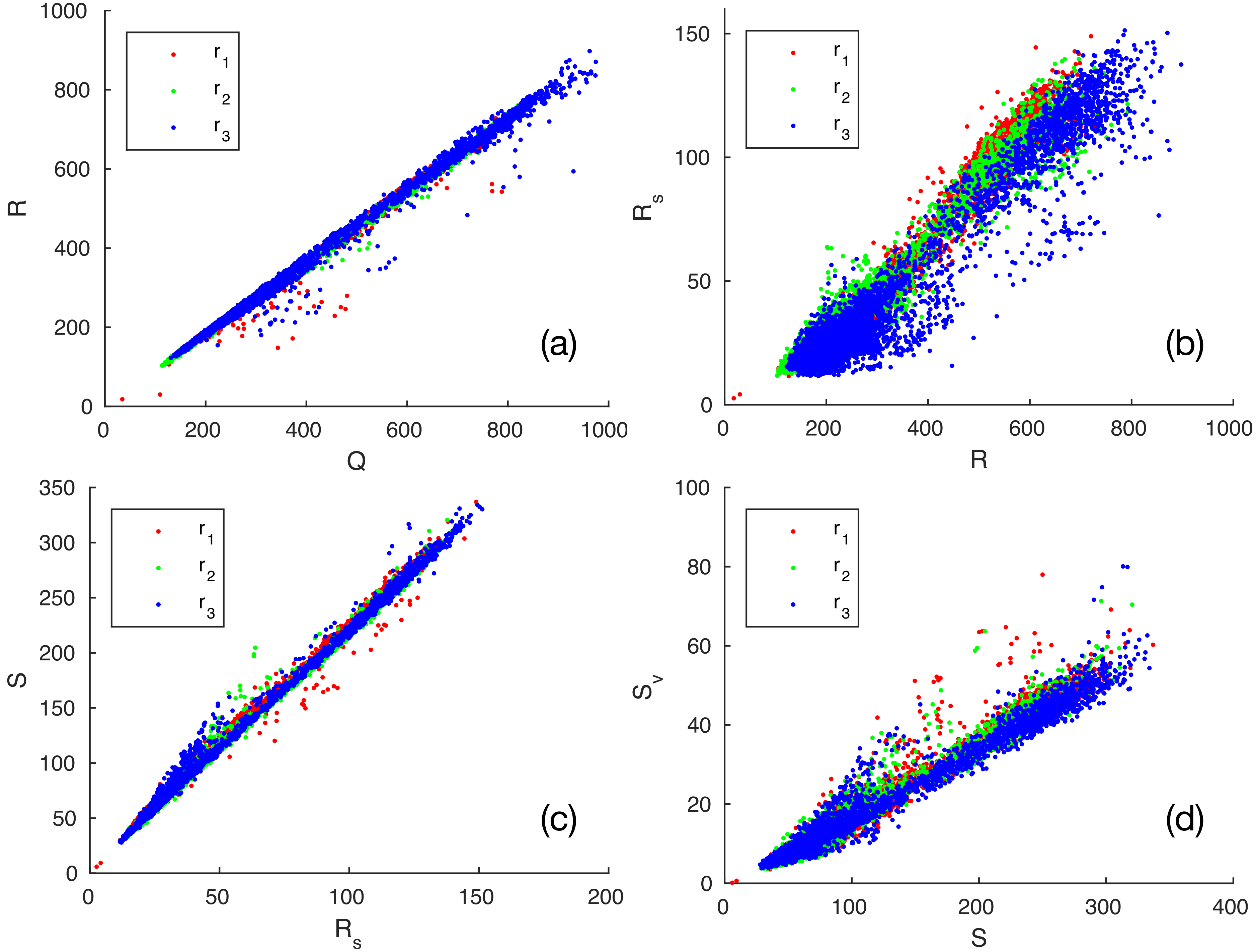
We do **not measure** the number of **queries from clients**, as this will **vary** strongly **between resolvers**!

- **Intuition**: we can **predict** the **number of signature validations** (S_v) **based on** the **number of outgoing queries** from a resolver (Q)

Measurement setup



Observed behaviour



- **Intuition**: *a **linear** model can predict S_v from Q*

Resolver model

$$
\begin{array}{llll}
f_1 : R = \bar{r}Q + \beta_1 & & f_3 : S = \bar{s}R_s + \beta_3 \\
f_2 : R_s = \alpha_s R + \beta_2 & & f_4 : S_v = \alpha_v S + \beta_4
\end{array}
$$

with:

$\bar{r}$ - the average number of responses per query.

α_s - the fraction of responses with signatures.

$\bar{s}$ - the average number of signatures per response.

α_v - the fraction of signatures that is validated.

$$f : S_v = aQ + b$$

$$a = \alpha_v \bar{s} \alpha_s \bar{r}$$

$$b = \alpha_v(\bar{s}(\alpha_s \beta_1 + \beta_2) + \beta_3) + \beta_4$$

Validation

We **validated** our model **according to** the following **criteria**:

I. The **model works for different** DNS resolver **implementations**

   We tested the model for **two** popular open source DNS resolver **implementations** (Unbound and BIND).

II. The **model** has **stable properties over time**; **only α_s may vary significantly** as time progresses (<— we vary this parameter in our predictions)

   We measured at **different times** over **five months** and compared the model parameters.

Validation

[…] cont'd

III. The **model works for different client populations** (i.e. for different operational DNS resolvers).

*We tested using **traffic** from **four sources** (busy resolvers at SURFnet, and our university resolvers), and performed **worst-case** estimations (see paper).*

IV. The **model is** a good **predictor** of **observed data**.

*We used **statistical goodness-of-fit** tests to **compare predictions** to **empirically observed data**.*

ECC benchmarks

- Benchmarked **5 implementations** on modern CPU:
  - **3x OpenSSL** for **RSA** and **ECDSA**
    - **0.9.8** branch as "**legacy**"
    - **1.0.1** branch as "**LTS**"
    - **1.0.2** branch as "**new and optimised**"
  - **Ed25519** Donna implementation (optimised)
  - **Ed448** Goldilocks implementation (optimised)

- **Method**: perform **100 iterations** of **10 seconds** of continuous signature validation, **count number of validations** in that period.

Benchmarking results

Order of magnitude slower

ECC algorithm	OpenSSL version	Compared to*			
		RSA		ECDSA	
		1024	2048	P-256	P-384
ECDSA P-256	0.9.8zh	27.5	8.4	-	-
	1.0.1f	26.0	7.9	-	-
	1.0.2e	11.5	3.6	-	-
ECDSA P-384	0.9.8zh	57.7	17.6	-	-
	1.0.1f	77.6	23.4	-	-
	1.0.2e	87.3	27.2	-	-
Ed25519	(1.0.2e)†	7.9	2.5	0.7	0.1
Ed448	(1.0.2e)†	23.4	7.3	2.0	0.3

*the number means that the E

†independent implementations compared to this OpenSSL version

Better, but still significantly slower

Key benchmarks

- **Key benchmarks** used later on:

Implementation	Curve	Performance	Why this benchmark?
OpenSSL 1.0.1	ECDSA P-384	1,236 vals/s	Worst-case strongest broadly supported cipher
OpenSSL 1.0.1	ECDSA P-256	3,685 vals/s	LTS for most likely used cipher
OpenSSL 1.0.2	ECDSA P-256	9,787 vals/s	Illustrates what optimisation can do
Donna	Ed25519	14,162 vals/s	High-performance new cipher (RFC 8080)
Goldilocks	Ed448	4,817 vals/s	High-performance new strong cipher (RFC 8080)

Benchmarked on **Intel Xeon E5-2695 v3** at **2.3GHz**

Estimating future performance

- **Scenario 1**:
  Current *DNSSEC* ***deployment*** *switches to ECC overnight*

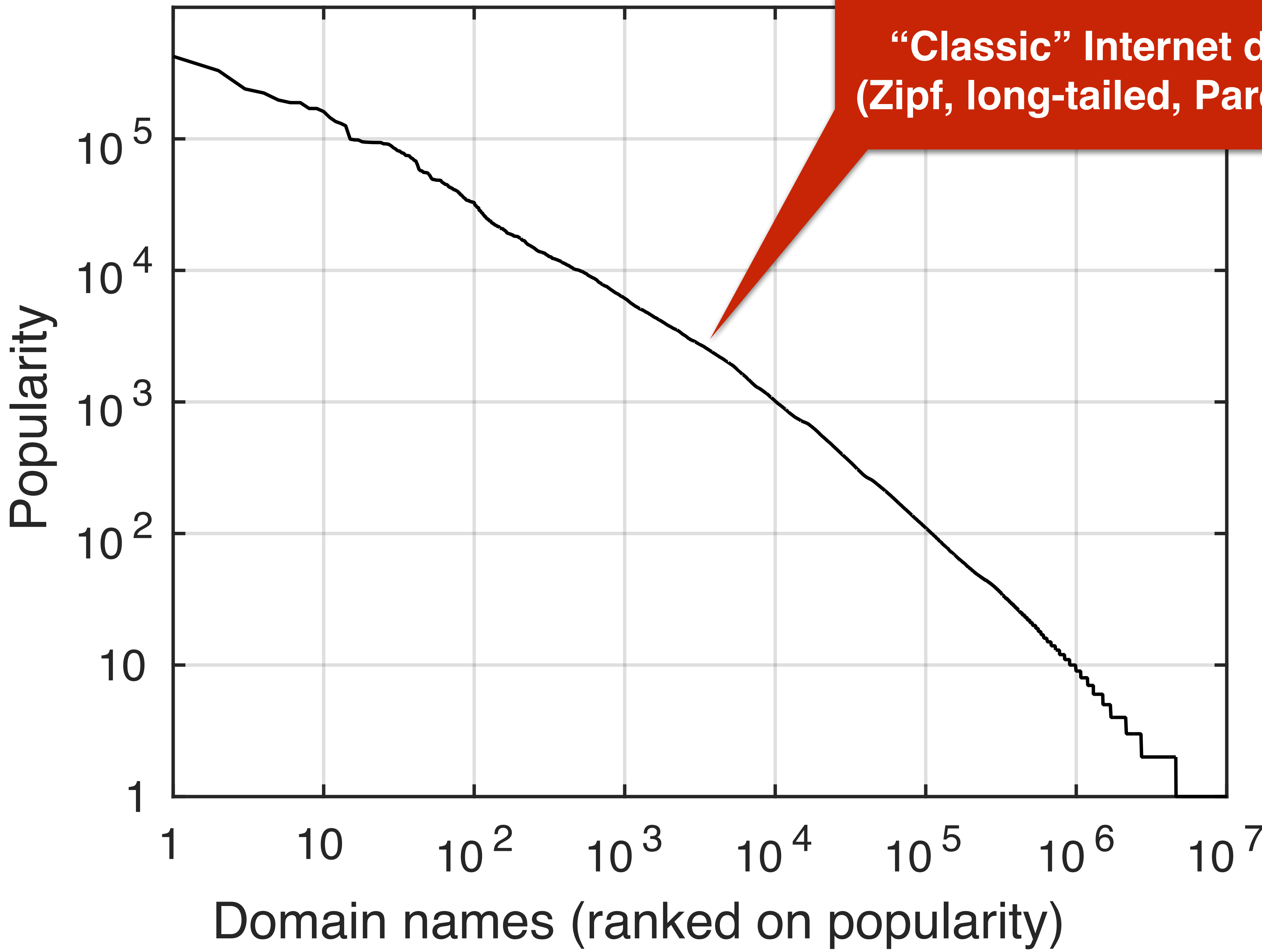
  evaluation: **requires ±150 validations per second** for a busy* resolver, **not a problem**

- **But what if everybody deployed DNSSEC with ECC?**

- **Scenario 2:**
  Popular-domains-first *growth to* ***100% DNSSEC deployment****, everyone uses* ***ECC***

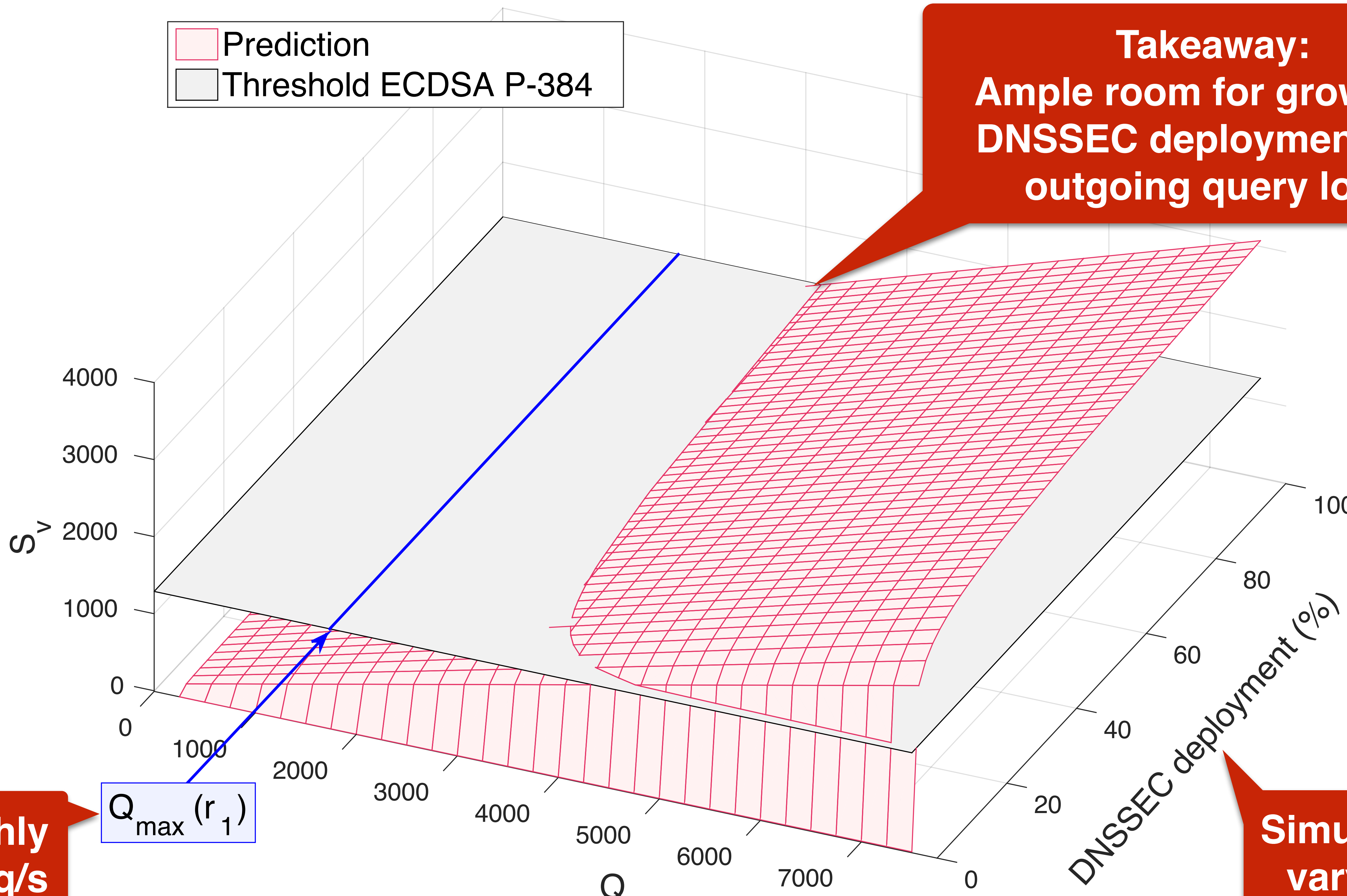
*the busiest resolver in our study processed ~20k qps from clients

What is popular?



"Classic" Internet distribution
(Zipf, long-tailed, Pareto, 80/20…)

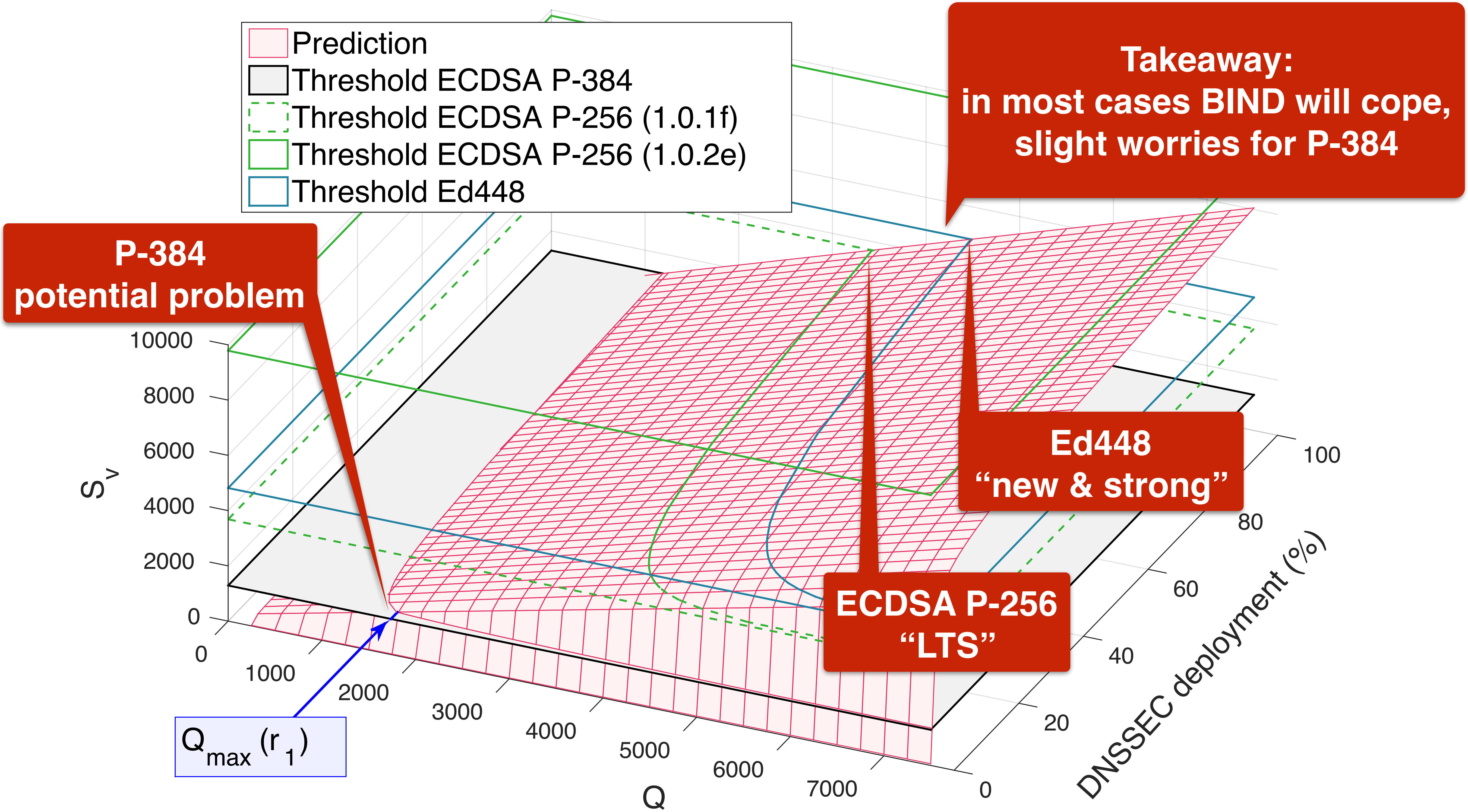
Results for Unbound



Takeaway:
Ample room for growth in DNSSEC deployment and outgoing query load

Roughly 1800 q/s

Simulated by varying α_s


Prediction
Threshold ECDSA P-384
Threshold ECDSA P-256 (1.0.1f)
Threshold ECDSA P-256 (1.0.2e)
Threshold Ed448
Takeaway:
in most cases BIND will cope,
slight worries for P-384
P-384
potential problem
Ed448
“new & strong”
ECDSA P-256
“LTS”
S_v
10000
8000
6000
4000
2000
0
Q
0
1000
2000
3000
4000
5000
6000
7000
DNSSEC deployment (%)
0
20
40
60
80
100
$Q_{max}(r_1)$

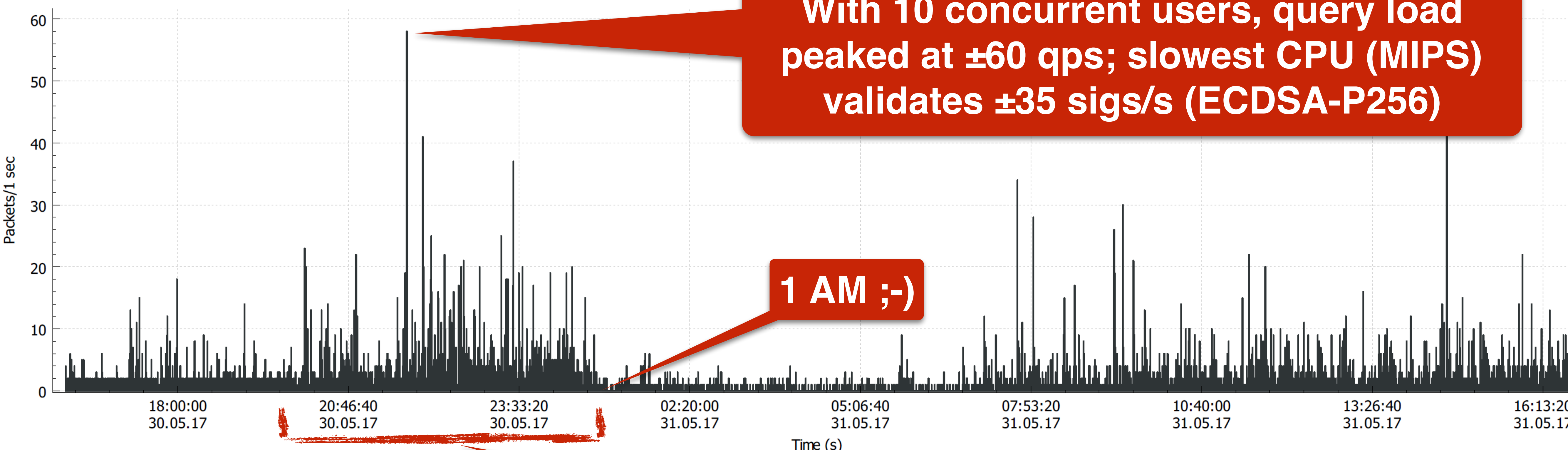
Additional benchmarks

- **Original benchmarks** on **Intel x86 64-bit** CPUs, what about **other architectures?**
- **Student project**: benchmark **ARM** and **MIPS** implementations (common in, e.g., **home routers**)
- Key **takeaways**:
  - **Performance** is **low**, but optimisations are gradually being implemented, **sufficient for "home"** scenarios
  - **ECDSA** sometimes **faster** than **EdDSA** due to availability of optimised implementations

n=1 home router experiment

- One of my students measured on his home router

Takeaway #1:
With 10 concurrent users, query load peaked at ±60 qps; slowest CPU (MIPS) validates ±35 sigs/s (ECDSA-P256)



Takeaway #2:
Student parties are not what they used to be :-)=)

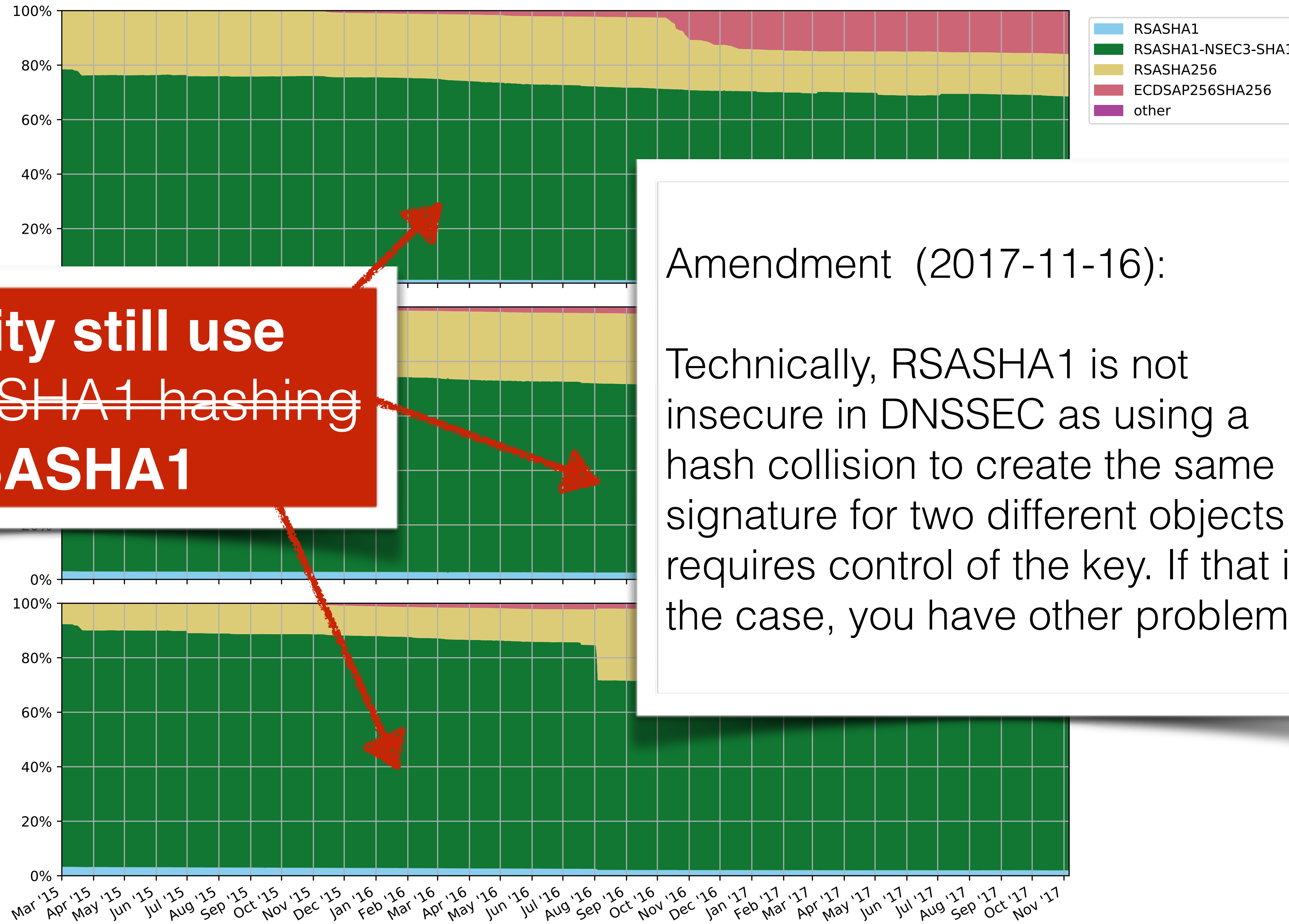
- Note: student got inforn
roommates!

Insight into adoption

- **Until 2015** there was **virtually no adoption** of ECDSA signing schemes standardised in RFC 6605
- **Late 2015**, **CloudFlare** was the **first DNS operator to adopt ECDSA** signing at scale
- **How has adoption developed since then?**

Adoption of ECC

.com



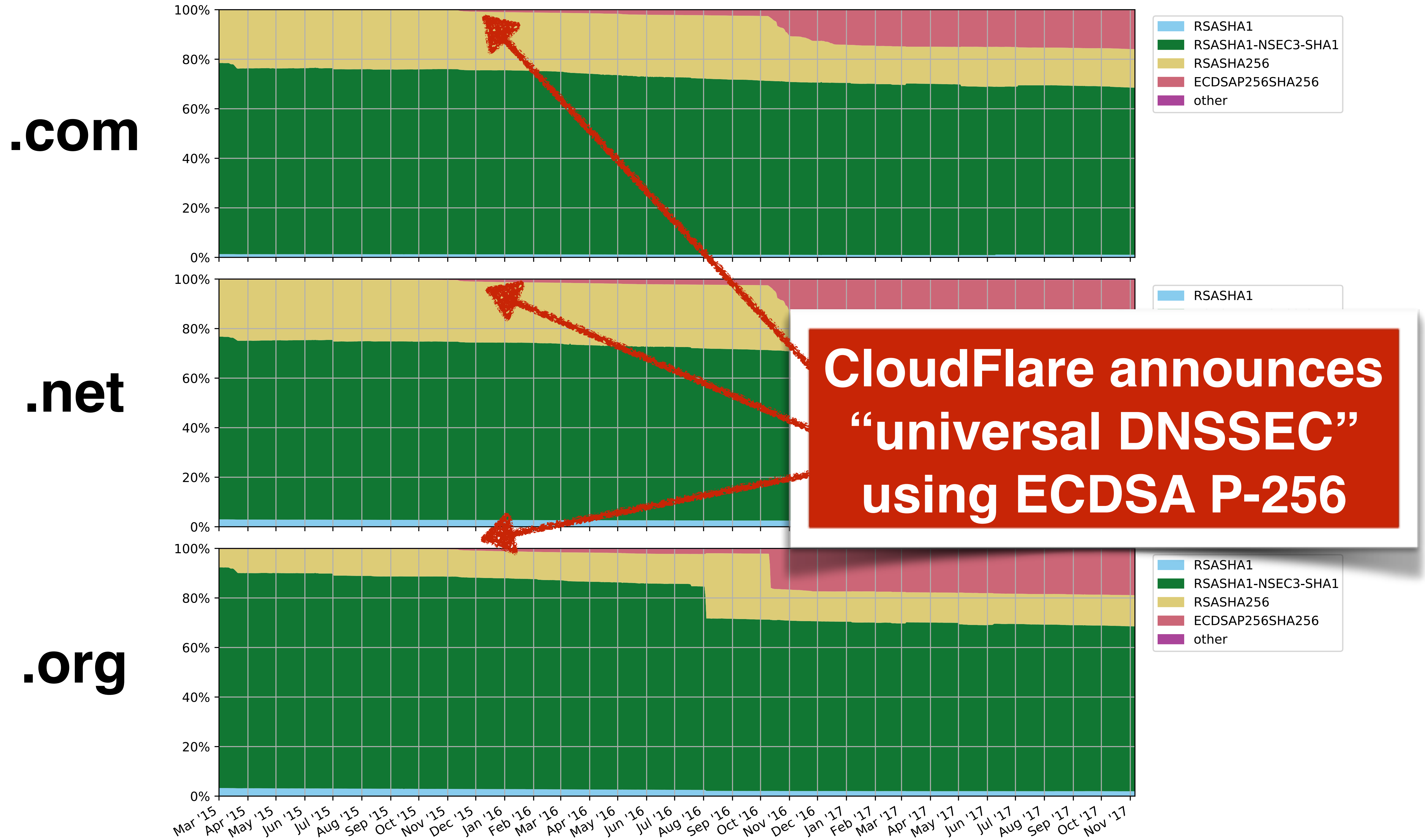
.org

Majority still use
~~*insecure* SHA1 hashing~~
RSASHA1

Amendment (2017-11-16):

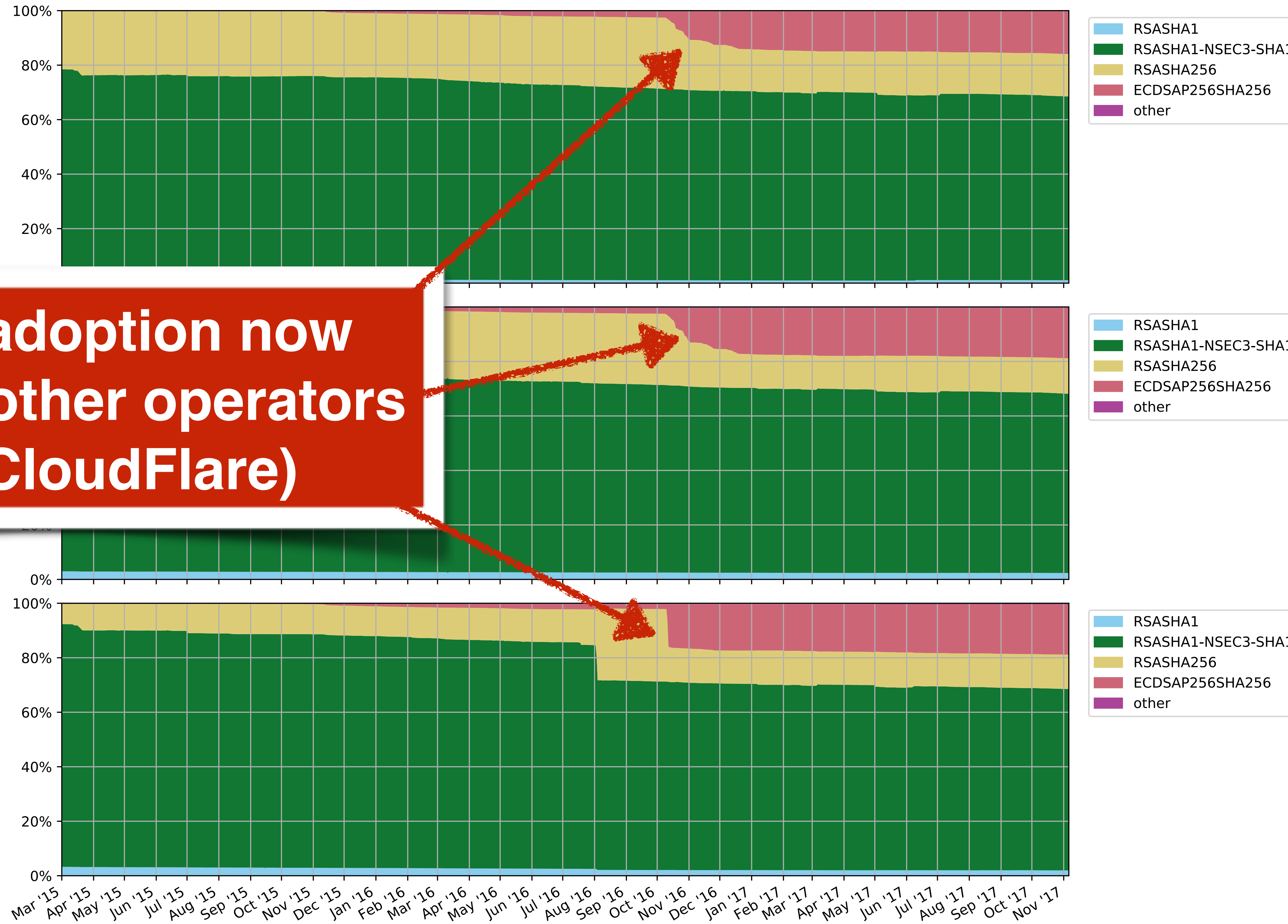
Technically, RSASHA1 is not insecure in DNSSEC as using a hash collision to create the same signature for two different objects requires control of the key. If that is the case, you have other problems.

Adoption of ECC



Adoption of ECC

.com



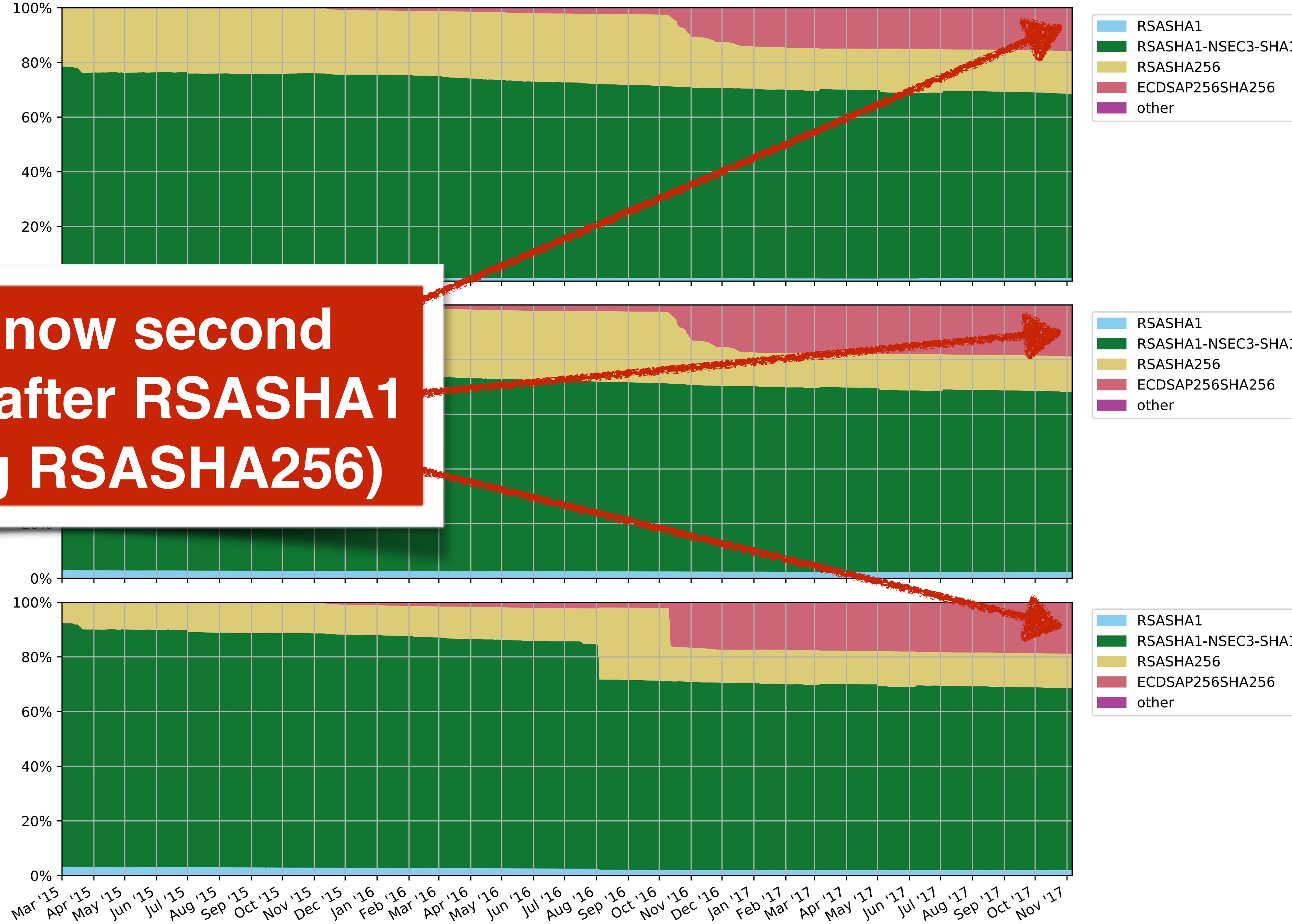
ECDSA adoption now driven by other operators (than CloudFlare)

.org

Adoption of ECC

.com

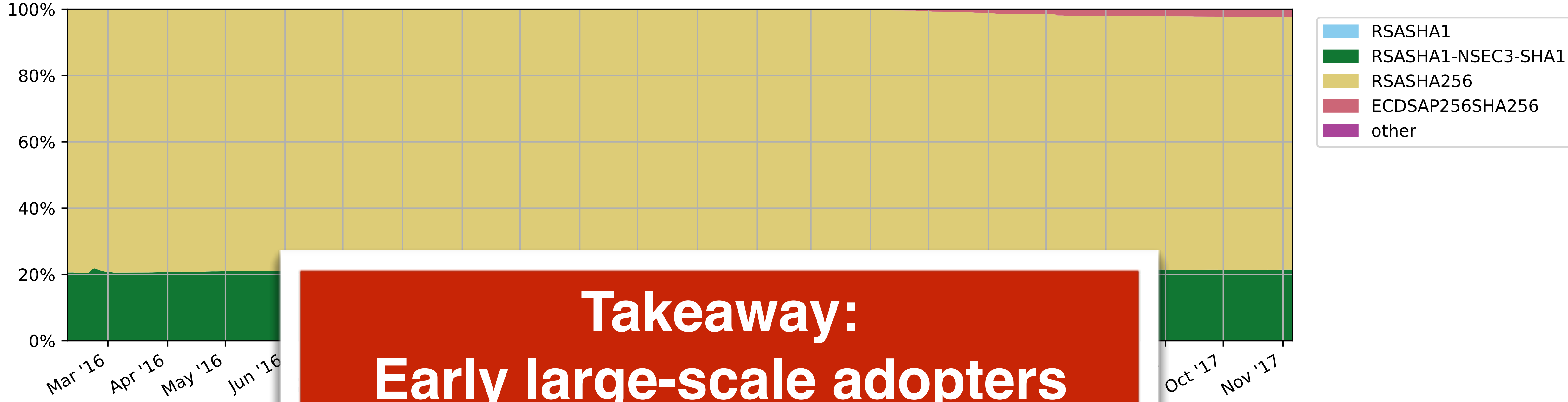
.org



ECDSA now second algorithm after RSASHA1 (replacing RSASHA256)

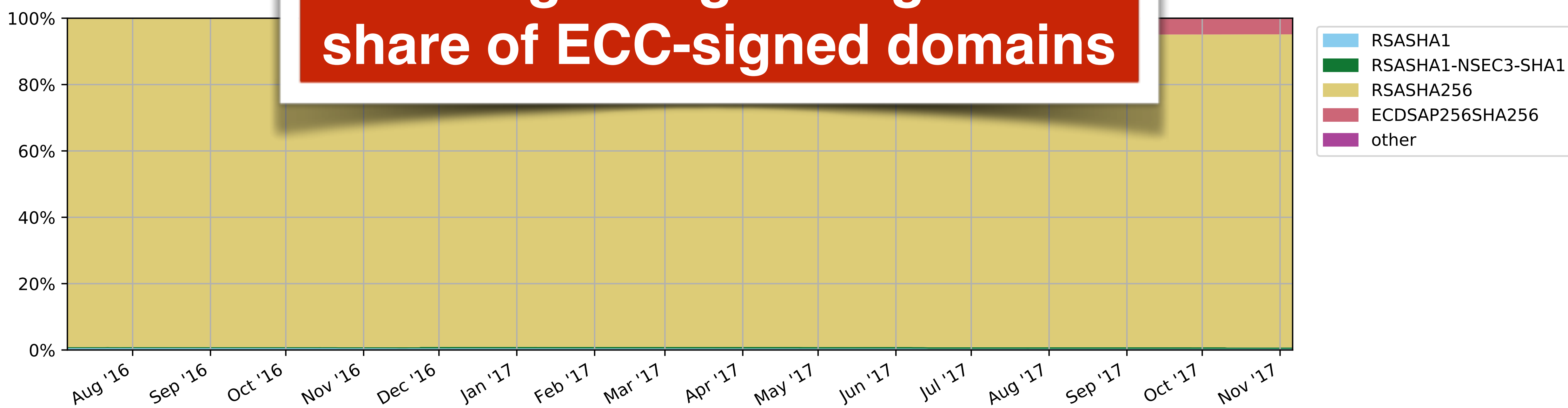
Adoption of ECC

.nl



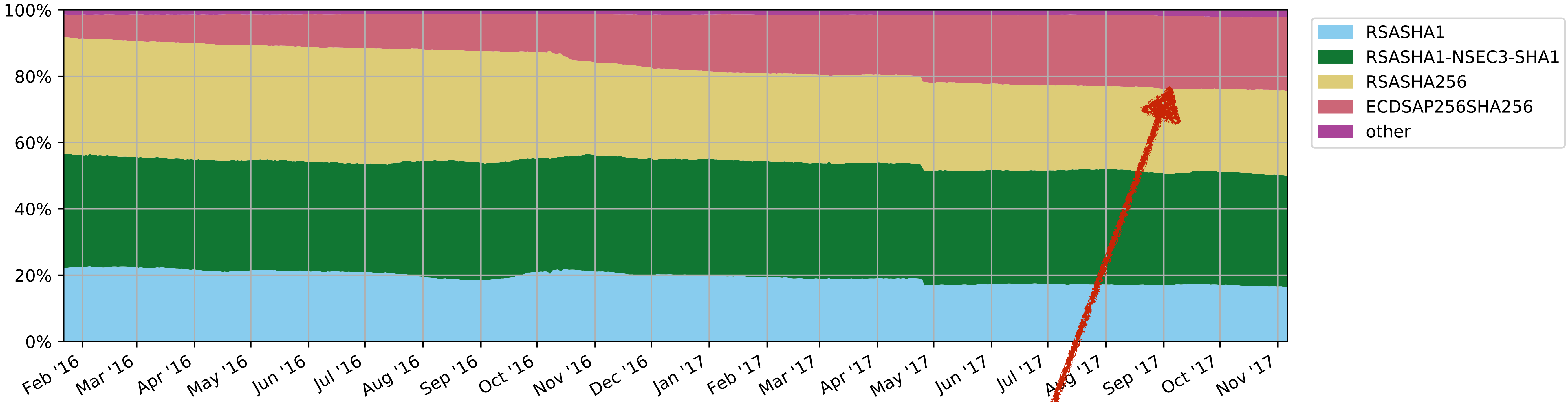
Takeaway:
Early large-scale adopters take longer to get a significant share of ECC-signed domains

.se



Adoption of ECC

Alexa top 1M (1.7% DNSSEC signed)



22% use ECDSA
61% of those use CloudFlare

Conclusions

- **ECC** algorithms are **sufficiently performant for widespread adoption in DNSSEC**
- **Recommendation**: operators should **switch to** using **ECDSA** for signing, and consider **EdDSA** for the **longer term**
    - Recap of **benefits**: (much) **smaller** DNS **packets** —> **no fragmentation**, much **less amplification**
    - **Resolver operators**: prefer **newer, optimised crypto libraries** for increased performance
- **Adoption** is slowly **taking off**

References

[ComMag14] - van den Broek, G., van Rijswijk, R. M., Sperotto, A., & Pras, A. (2014). DNSSEC Meets Real World: Dealing with Unreachability Caused by Fragmentation. IEEE Communications Magazine, 52(April), 154–160.

[IMC14] - van Rijswijk-Deij, R., Sperotto, A., & Pras, A. (2014). DNSSEC and its potential for DDoS attacks. In Proceedings of ACM IMC 2014. Vancouver, BC, Canada: ACM Press.

[CCR15] - van Rijswijk-Deij, R., Sperotto, A., & Pras, A. (2015). Making the Case for Elliptic Curves in DNSSEC. ACM Computer Communication Review (CCR), 45(5).

[CNSM16] - van Rijswijk-Deij, R., Jonker, M., & Sperotto, A. (2016). On the Adoption of the Elliptic Curve Digital Signature Algorithm (ECDSA) in DNSSEC. In Proceedings of the 12th International Conference on Network and Service Management (CNSM 2016). Montréal, Canada: IFIP.

[ToN17] - van Rijswijk-Deij, R., Hageman, K., Sperotto, A., & Pras, A. (2017). The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation. IEEE/ACM Transactions on Networking, 25(2).

Thank you for your attention!
Questions?

acknowledgments: with thanks to my students
Kaspar Hageman, Breus Blaauwendraad and J.J. Yu

Data for adoption supplied by the OpenINTEL project (https://openintel.nl/)

(references to papers included in PDF of full slide deck)

nl.linkedin.com/in/rolandvanrijswijk

@reseauxsansfil

roland.vanrijswijk@surfnet.nl
r.m.vanrijswijk@utwente.nl

UNIVERSITY OF TWENTE.

SURF NET