Reaping and breaking keys at scale: when crypto meets big data

Nils Amiet
Yolan Romailler

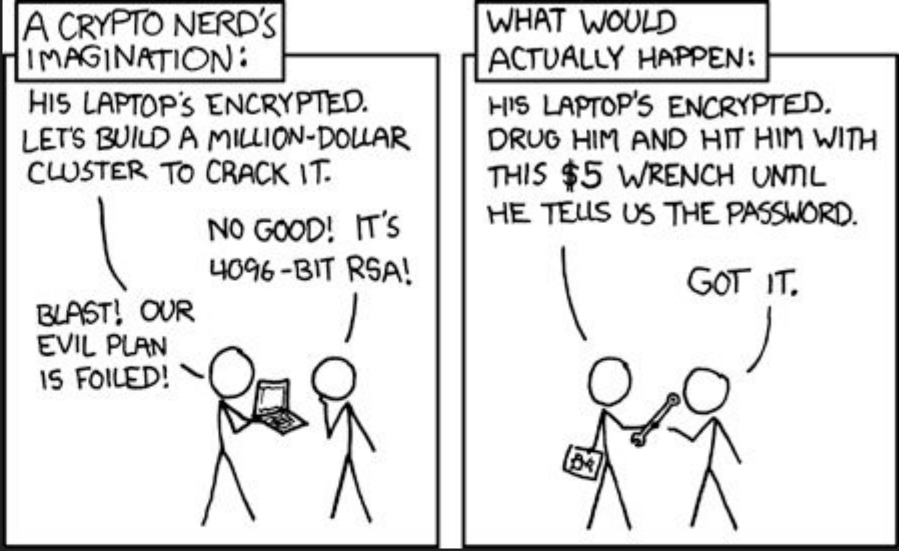
August 2018 — DEF CON 26

KUDELSKI SECURITY

Public keys… what for?

- Break them!
  - Retrieve the private keys
  - Show how easy it is
  - If we can do it…
  - … guess who can too!

Crypto recap: RSA


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

- RSA (Rivest–Shamir–Adleman)
  - Choose two **large prime numbers** p and q, typically 1024-2048 bits.
  - Public key **(n, e)**
    - with n = p * q
    - and some e such that e and λ(n) are coprime
  - Private key **(n, d)** where d ≡ e^−1 (mod λ(n))
  - RSA security relies on the hardness of the **integer factorization problem**

Crypto recap: RSA

p q

Crypto recap: RSA

$$p \quad \cdot \quad q$$

Crypto recap: RSA

$$n = p \cdot q$$

Crypto recap: RSA

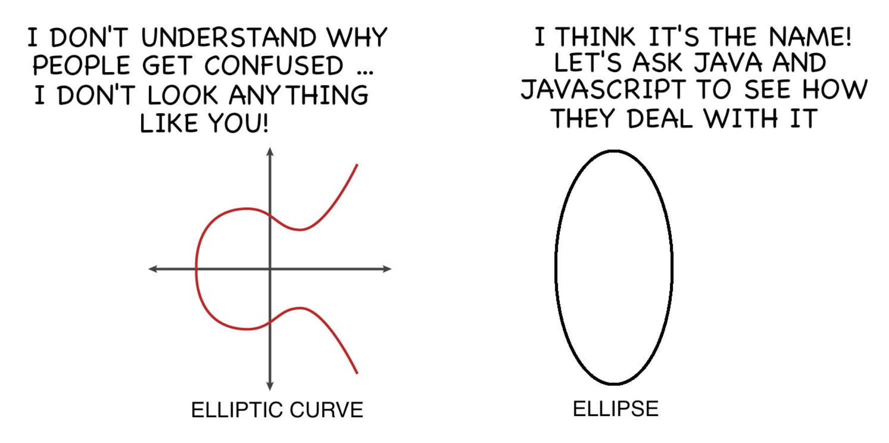
GCD attack: the GCD (greatest common divisor) of **n** and **m** is **q** and we can easily compute **n/q = p** and **m/q = r.**

$$n = p \cdot q$$

$$m = q \cdot r$$

Crypto recap: ECC

- ECC (“Elliptic Curve Cryptography”)
  - Security based on the hardness of the EC **discrete logarithm problem**
  - Working with an elliptic curve C
  - Private key is an integer d
  - Public key is a point Q = (x, y) = dG
    - where (x, y) are the coordinates of the point **on** a given known curve


I DON'T UNDERSTAND WHY PEOPLE GET CONFUSED ... I DON'T LOOK ANYTHING LIKE YOU!
I THINK IT'S THE NAME! LET'S ASK JAVA AND JAVASCRIPT TO SEE HOW THEY DEAL WITH IT
ELLIPTIC CURVE
ELLIPSE

Passive attacks on public keys

- The Return of Coppersmith’s Attack (ROCA)
- Invalid parameters
  - DSA generator
  - Key sizes
  - Invalid curve attacks
- **RSA modulus factorization** (Batch GCD)

★ Batch GCD already used in 2010, 2012, 2016 to break weak keys

  - On datasets <100M keys

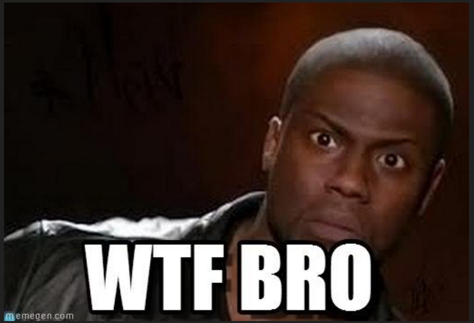
★ These are all **known attacks!**

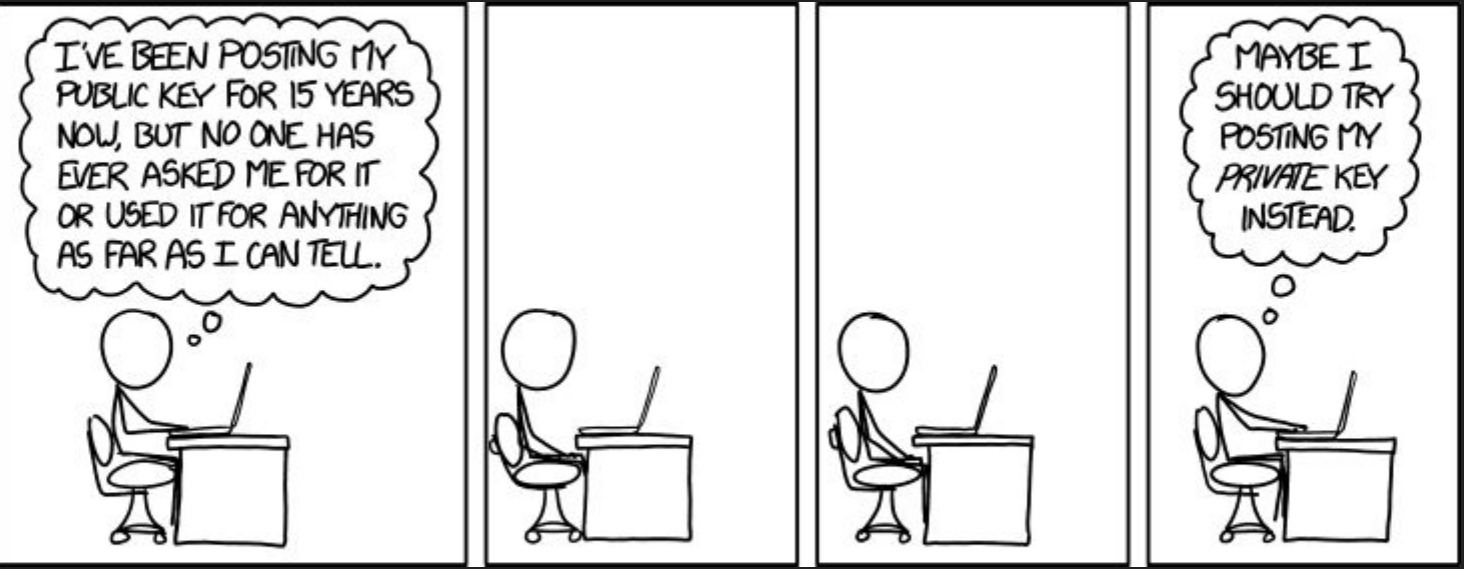
★ And they are completely passive, the target is left unaware

Collecting public keys

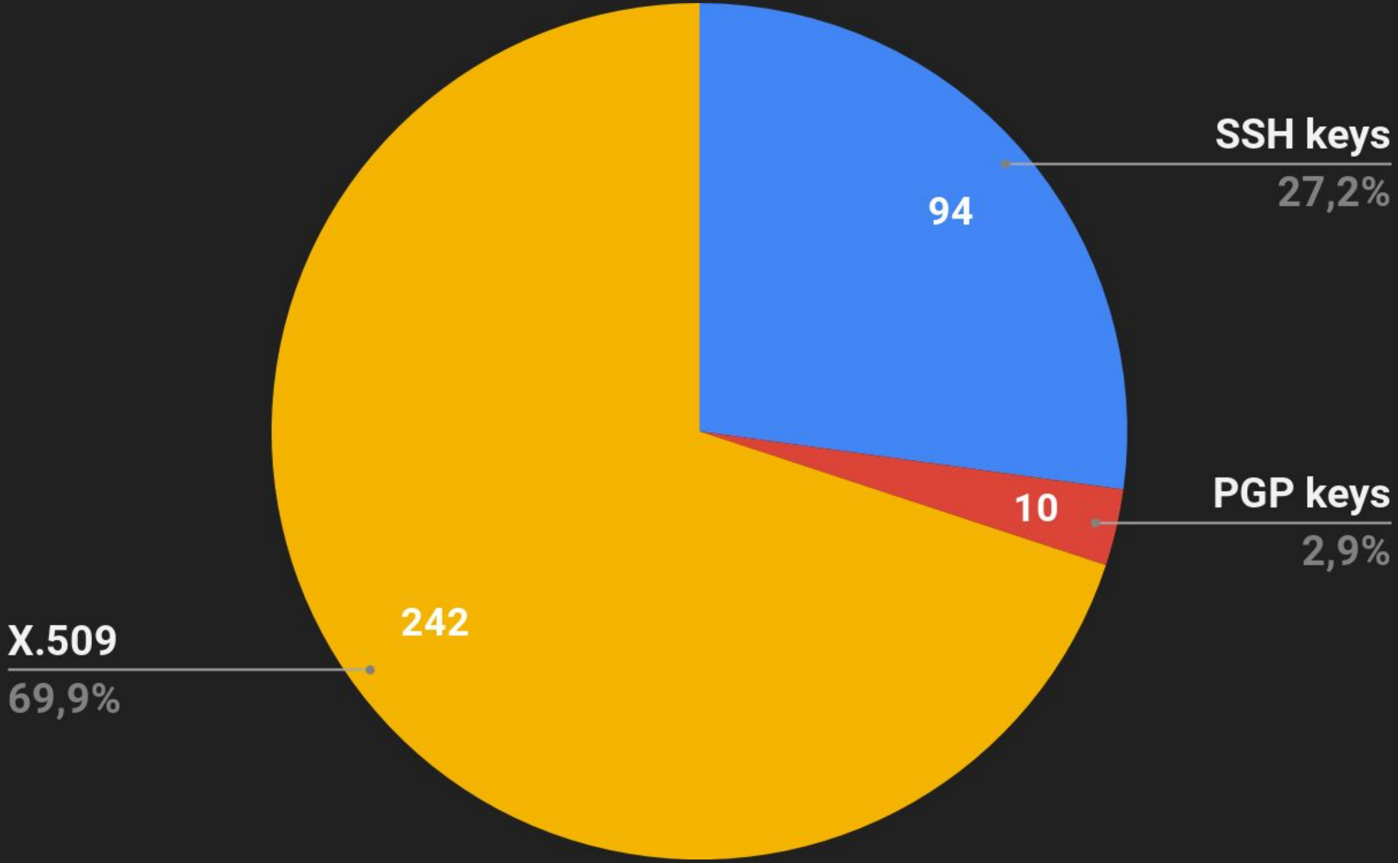
- X.509 certificates
- SSH keys
- PGP keys

Fun fact:
Some certificates have a negative validity period!




I'VE BEEN POSTING MY PUBLIC KEY FOR 15 YEARS NOW, BUT NO ONE HAS EVER ASKED ME FOR IT OR USED IT FOR ANYTHING AS FAR AS I CAN TELL.
MAYBE I SHOULD TRY POSTING MY *PRIVATE* KEY INSTEAD.

Keys (millions) per key container type



Keys collected per data source

- X.509 certificates
  - > 200M from HTTPS scans
  - 1-2M each from SMTP(S), POP3(S) and IMAP(S) scans
- SSH keys
  - 71M from CRoCS* dataset
  - 17M from SSH scans
  - 4.7M on Github.com
  - 1.2M on Gitlab.com
- PGP keys
  - 9.5M on SKS key servers
  - 220k on Keybase.io
  - 8k on Github.com

Fun fact:
We validated CRoCS results.
One smart card model had a bad RNG and generated keys with common factors

*CRoCS: Center for Research on Cryptography and Security

Our public keys stash: Big Brother style

- Attacks like RSA Batch GCD work best with larger datasets
  - More keys = more chances of finding common factors
- We collected as many public keys as we could
  - > 346M unique keys and growing
  - Collection made over 1 year
- 273M unique domain names on Certificate Transparency… profit!
  - Still in the process of ingesting all the certificates!

Key types

RSA	327M
ECC	14M
DSA	2.6M
ElGamal	2.5M
GOST R 34.10-2001	1k
Other	<1k

Tools

Data collection:

- Fingerprinting with cert/key grabbing: **Scannerl** with custom modules
- Key parsers: Python
- Data ingestion: NiFi and HDFS
- Data exploration: Presto

Breaking keys:

- Batch GCD on RSA keys, using a custom **distributed** implementation
- ROCA attack on RSA keys
- Sanity checks on EC keys

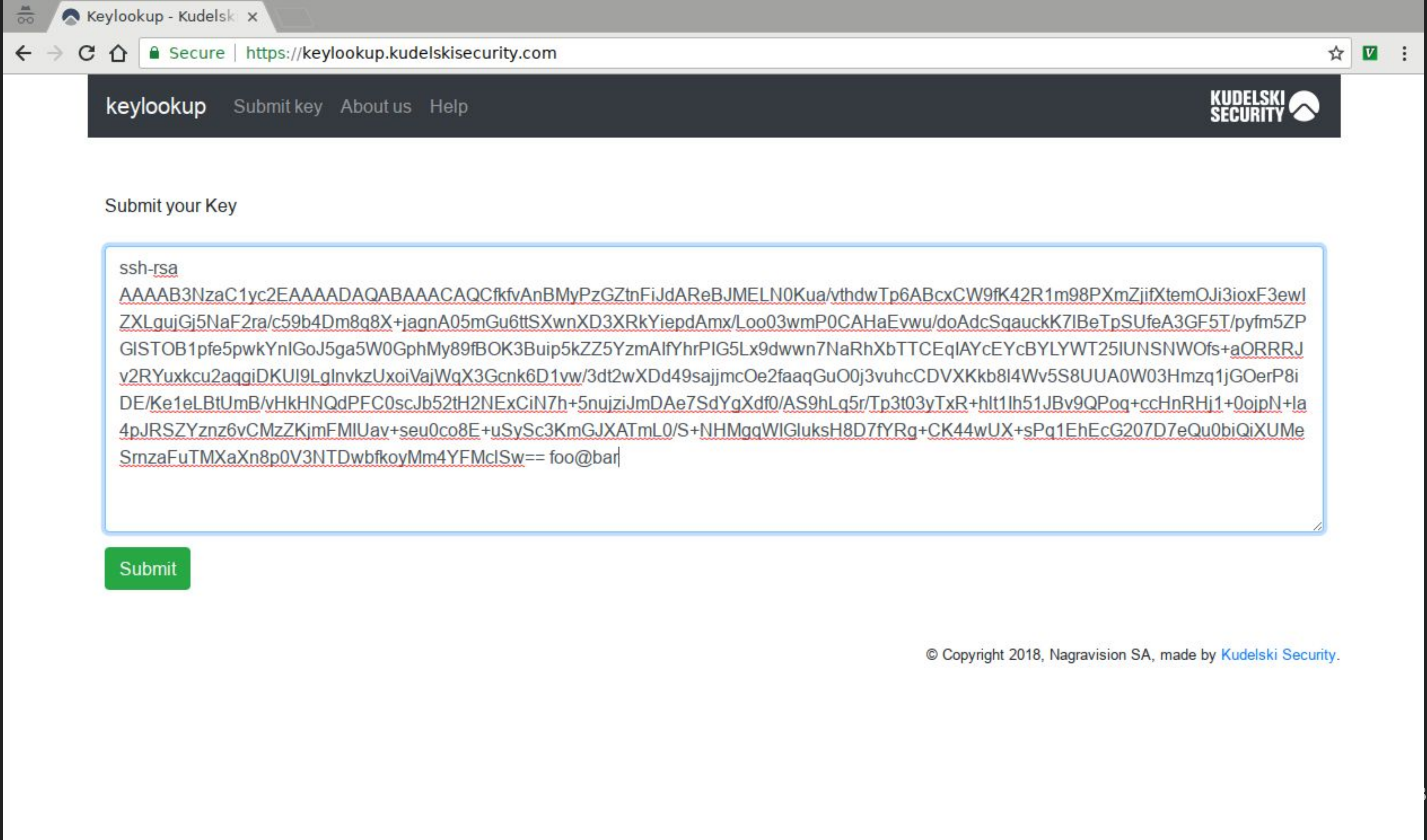
Demo

Test your keys today!

You can go to our website:

keylookup.kudelskisecurity.com

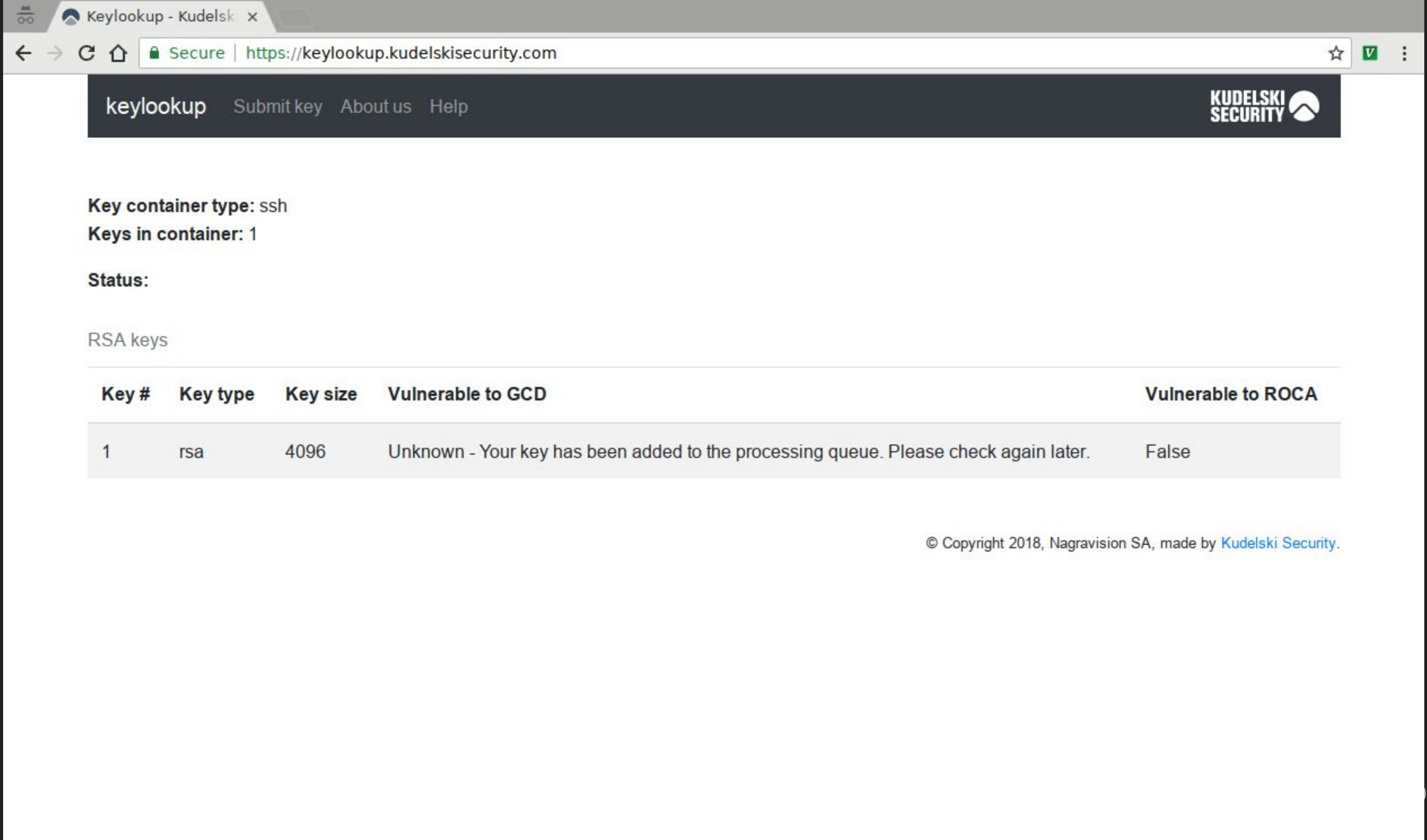
and submit your key to test it against our dataset!



Submit your Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCfkfvAnBMyPzGZtnFiJdAReBJMELN0Kua/vthdwTp6ABcxCW9fK42R1m98PXmZjifXtemOJi3ioxF3ewl
ZXLgujGj5NaF2ra/c59b4Dm8q8X+jagnA05mGu6ttSXwnXD3XRkYiepdAmx/Loo03wmP0CAHaEvwu/doAdcSqauckK7lBeTpSUfeA3GF5T/pyfm5ZP
GlSTOB1pfe5pwkYnlGoJ5ga5W0GphMy89fBOK3Buip5kZZ5YzmAlfYhrPlG5Lx9dwwn7NaRhXbTTCEqlAYcEYcBYLYWT25lUNSNWOfs+aORRRJ
v2RYuxkcu2aqgiDKUI9LglnvkzUxoiVajWqX3Gcnk6D1vw/3dt2wXDd49sajjmcOe2faaqGuO0j3vuhcCDVXKkb8l4Wv5S8UUA0W03Hmzq1jGOerP8i
DE/Ke1eLBtUmB/vHkHNQdPFC0scJb52tH2NExCiN7h+5nujziJmDAe7SdYgXdf0/AS9hLq5r/Tp3t03yTxR+hlt1Ih51JBv9QPoq+ccHnRHj1+0ojpN+la
4pJRSZYznz6vCMzZKjmFMlUav+seu0co8E+uSySc3KmGJXATmL0/S+NHMgqWlGluksH8D7fYRg+CK44wUX+sPq1EhEcG207D7eQu0biQiXUMe
SrnzaFuTMXaXn8p0V3NTDwbfkoyMm4YFMclSw== foo@bar
```

© Copyright 2018, Nagravision SA, made by Kudelski Security.



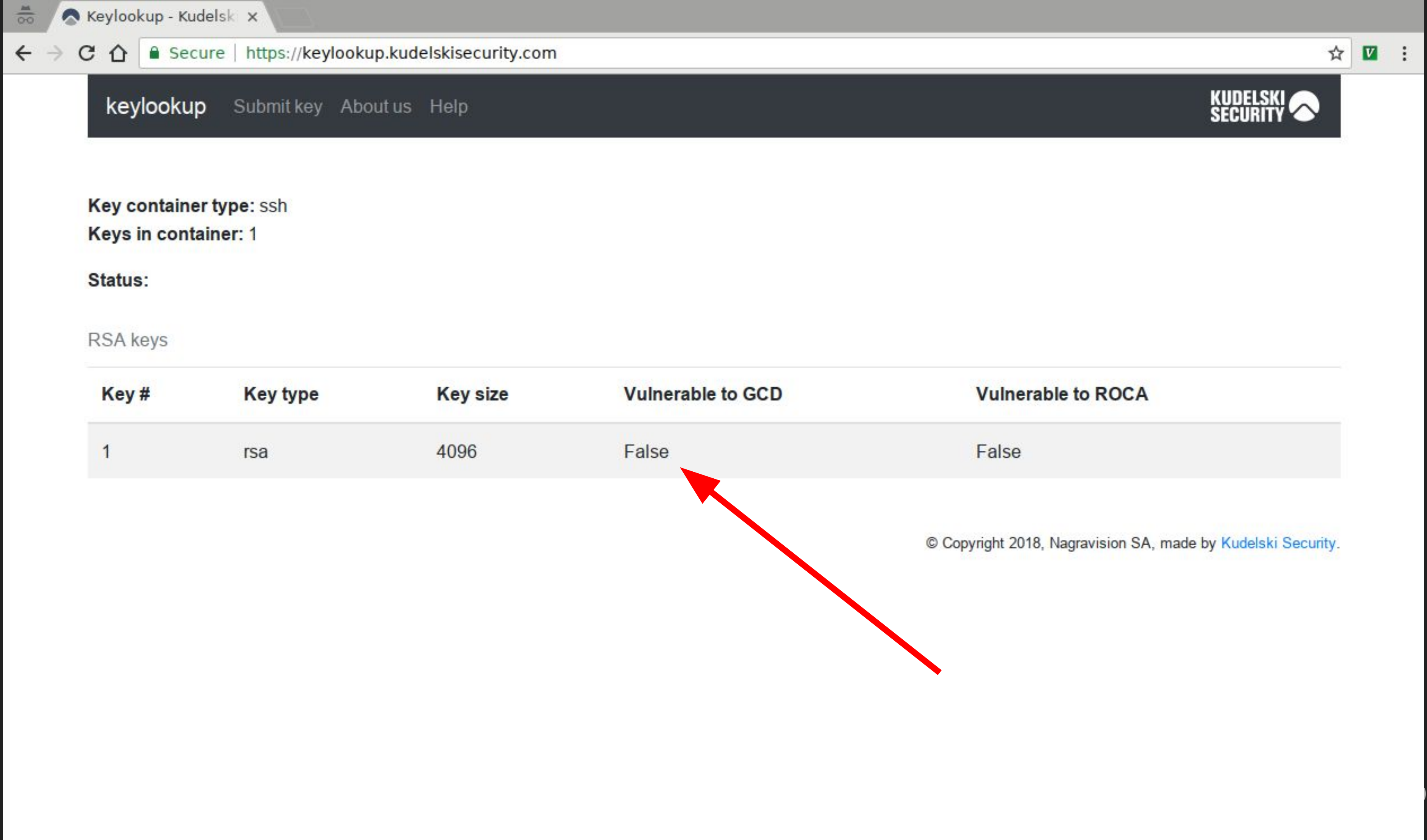
Key container type: ssh
Keys in container: 1

Status:

RSA keys

Key #	Key type	Key size	Vulnerable to GCD	Vulnerable to ROCA
1	rsa	4096	Unknown - Your key has been added to the processing queue. Please check again later.	False

© Copyright 2018, Nagravision SA, made by Kudelski Security.



Key container type: ssh
Keys in container: 1

Status:

RSA keys

Key #	Key type	Key size	Vulnerable to GCD	Vulnerable to ROCA
1	rsa	4096	False	False

© Copyright 2018, Nagravision SA, made by Kudelski Security.

Behind the scenes

- Batch-GCD:
  - 280 vCPUs cluster
  - 2 TB storage for storing product trees
  - Test new keys incrementally
    - Takes less than 1 hour for a bunch of keys
- HDFS cluster with 10+ data nodes
- Quick DB lookups thanks to partitioned tables
- Distributed fingerprinting using 50 Scannerl slaves

Results: RSA keys

Over 210k RSA keys factored through batch GCD

- Actually broken keys!
- 207k X.509 certificates
  - 260+ certs currently in use, 1400+ certs used over last year
- 3100+ SSH keys
- 295 PGP keys with common factors
  - 287 keys with more than 2 factors

Fun fact:
There are more PGP keys with 3+ factors than both SSH and X.509 ones together.

Results: RSA keys

Over 4k RSA keys vulnerable to ROCA

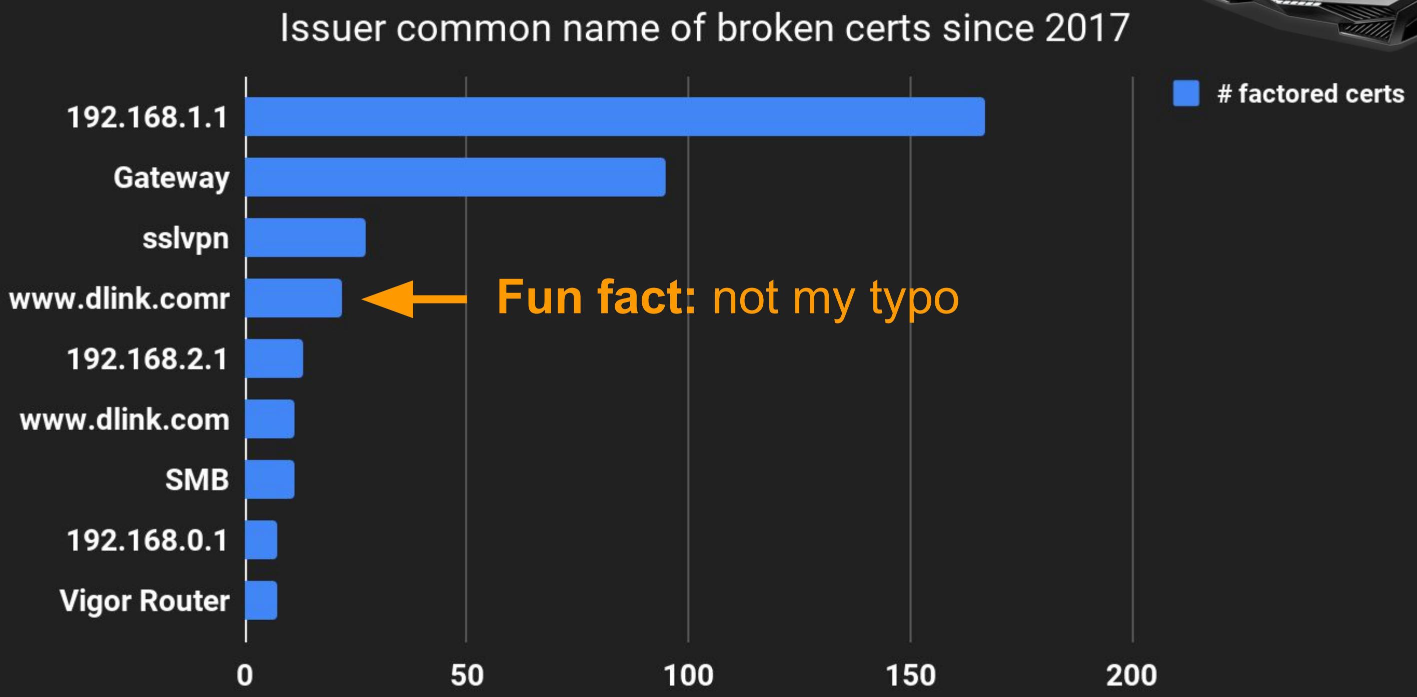
- 33% of size 2048 (weak), 64% of size 4096 (should be fine)
- Mostly PGP keys (97%)
- Found vulnerable keys on Keybase.io, Github.com and Gitlab.com!

Double check your keys!

Results: RSA keys

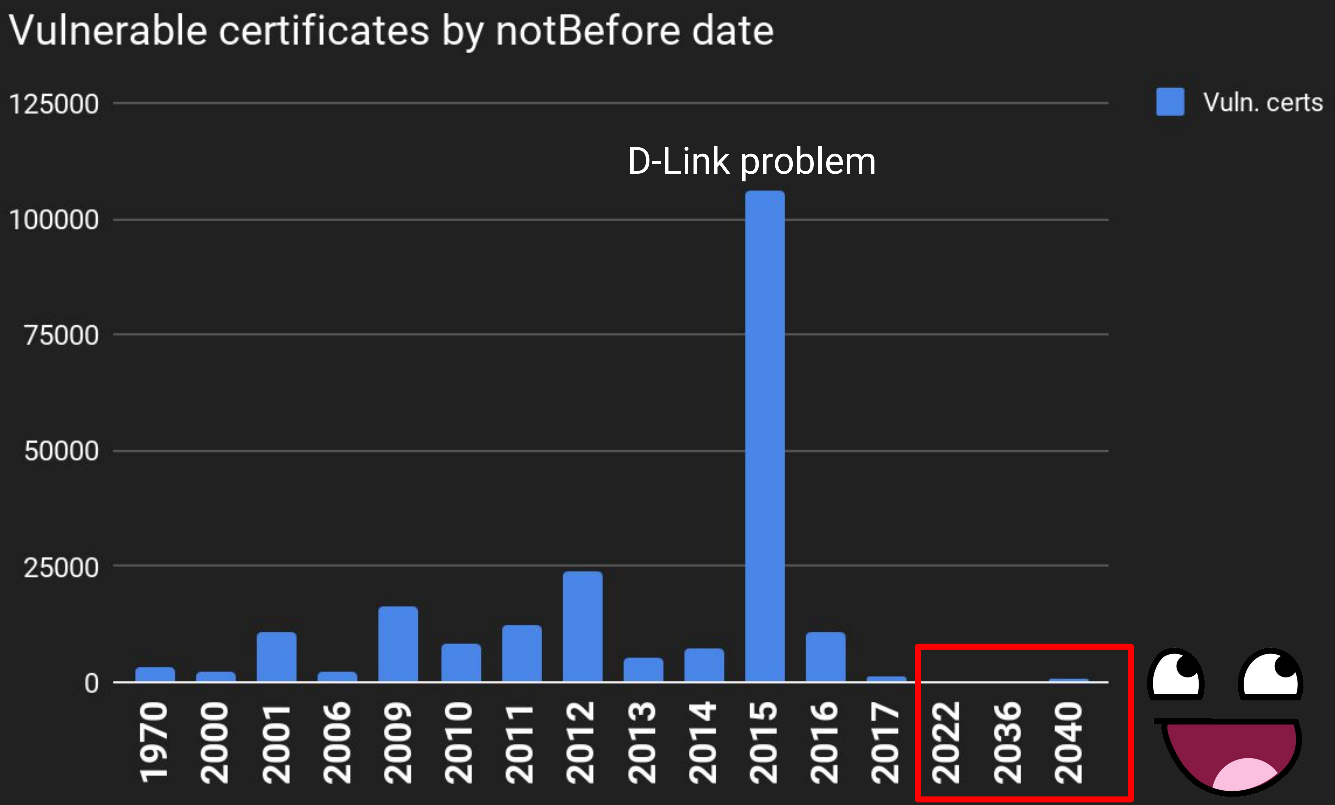
car salesman: *slaps roof of router* this bad boy can fit so many vulnerabilities in it.

Many routers seem concerned:



Fun fact: not my typo

Results: RSA keys

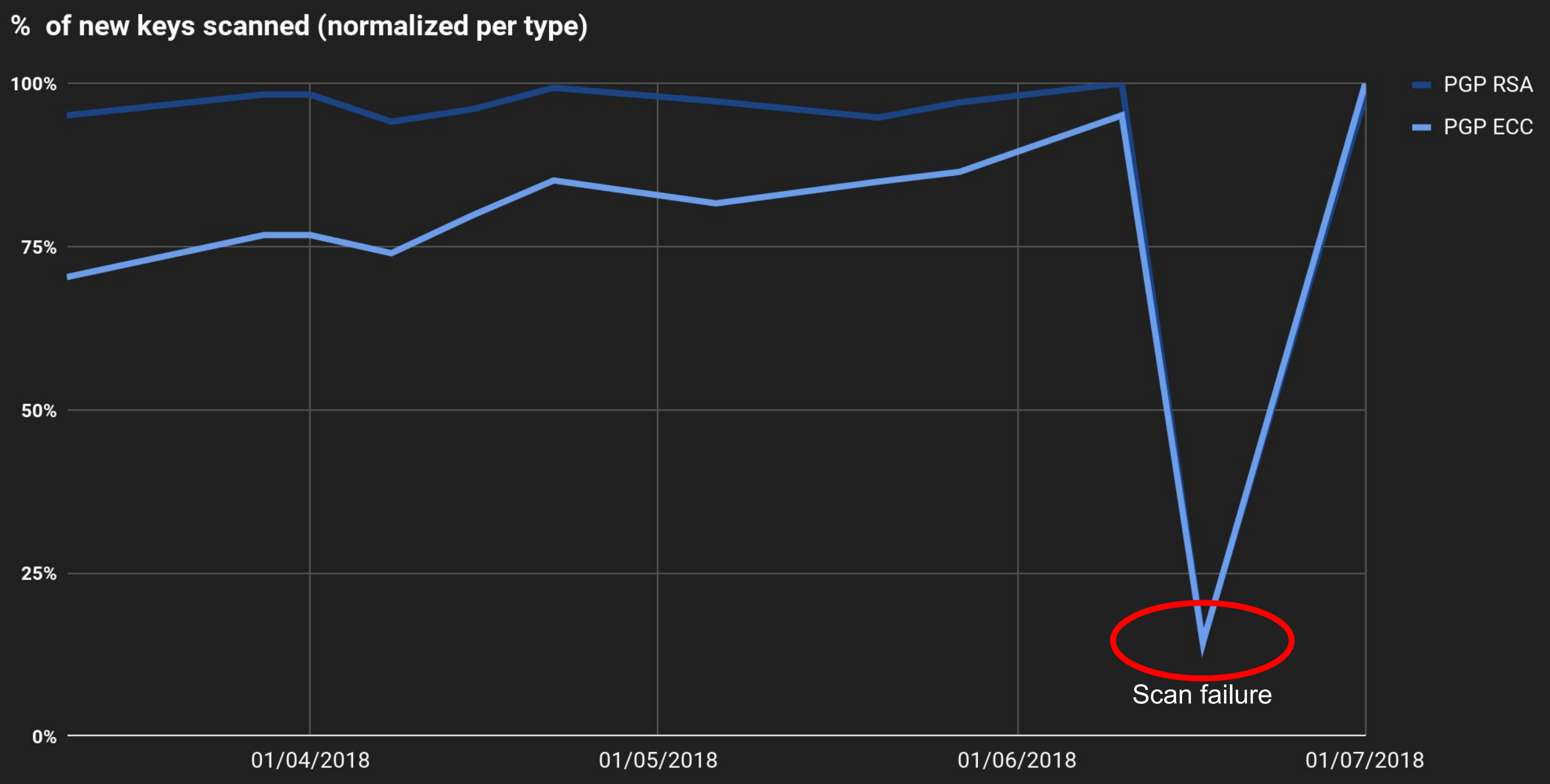


Results: ECC keys

- The adoption rate of ECC differs greatly depending on the source:
    - X509 and PGP are steadily adopting ECC
- Most common curves for SSH:

secp256r1	97,68%
secp521r1	1,87%
Curve25519	0,37%
secp384r1	0,07%

Growth of ECC keys



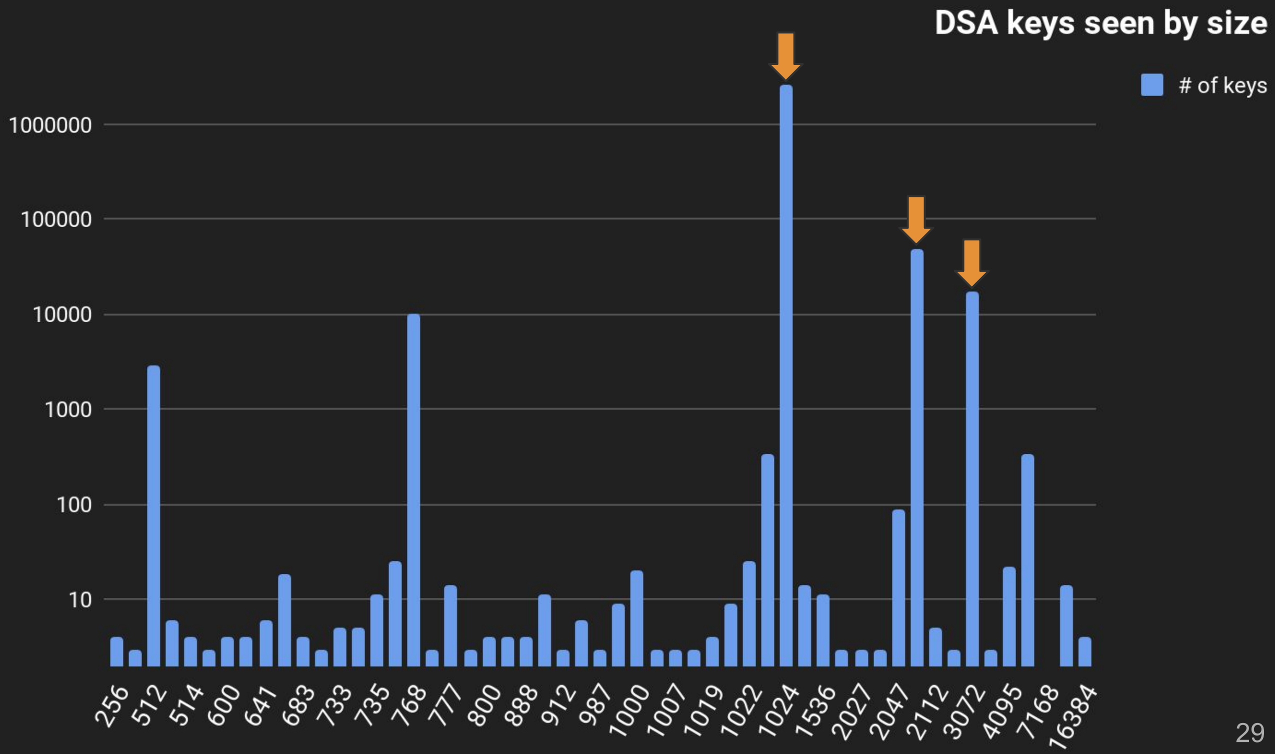
Fun facts

- At least 3442 keys are **re-used** as PGP keys, SSH keys and/or X509 certs!
- PGP subkey/master key ratio
  - Most people have only one subkey?!
- At least 486 of the keys we could factor had **more than 2 factors**!
- **DSA is dead** (OpenSSL deprecated it in 2015):
  - Only 3106 X.509 certs seen over last year
  - Less than 0.55% of SSH keys are DSA based

Fun facts

- Speaking of DSA:

FIPS 186-3 specifies L and N length pairs of:
(**1024**, 160),
(**2048**, 224),
(**2048**, 256),
(**3072**, 256).



Conclusion

- Mind your keys!
- Anybody can do the same kind of silent attack! *And maybe they already do…*
- Thank you!

Follow us: Twitter/Github

- Nils: github.com/amietn
- Yolan: @anomalroil
- Kudelski Security

Links

- Check your keys
  - https://keylookup.kudelskisecurity.com
- Find our open source code on Github
  - https://github.com/kudelskisecurity/k-reaper
  - https://github.com/kudelskisecurity/scannerl
- Find more results and analysis on our blog
  - https://research.kudelskisecurity.com