Factoring as a Service

Luke Valenta, Shaanan Cohney, Alex Liao,
Joshua Fried, Satya Bodduluri, Nadia Heninger

University of Pennsylvania

`seclab.upenn.edu/projects/faas`

Textbook RSA

[Rivest Shamir Adleman 1977]

Public Key

N	$= pq$ modulus
e	encryption exponent

Private Key

p, q	primes
d	decryption exponent $(d = e^{-1} \bmod (p-1)(q-1))$

Factoring

Problem: Factor N into p and q

- Lets an attacker compute the private key.
- The RSA assumption is not known to be equivalent to factoring
- Factoring is much harder than multiplication
- Best known algorithm: number field sieve

How long does factoring take with the number field sieve?

Answer 1

$L(1/3, 1.923) = \exp(1.923(\log N)^{1/3}(\log\log N)^{2/3})$

How long does factoring take with the number field sieve?

Answer 2

512-bit RSA:	$<$ 1 core-year
768-bit RSA:	$<$ 1,000 core-years
1024-bit RSA:	$\approx$ 1,000,000 core-years
2048-bit RSA:	Minimum recommended key size today.

How long does factoring take with the number field sieve?

Answer 3

512-bit RSA:	7 months — large academic effort [Cavallar et al., 1999]
768-bit RSA:	2.5 years — large academic effort [Kleinjung et al., 2009]
512-bit RSA:	2.5 months — single machine [Moody, 2009]
512-bit RSA:	72 hours — single Amazon EC2 machine [Harris, 2012]
512-bit RSA:	7 hours — Amazon EC2 cluster [Heninger, 2015]
512-bit RSA:	$<$ 4 hours — Amazon EC2 cluster [this work]

Brief Primer on Amazon EC2

`c4.8xlarge`

- 36 virtualized cores
- two Intel Xeon E5-2666 v3 processor chips
- 60GB RAM

Brief Primer on Amazon EC2

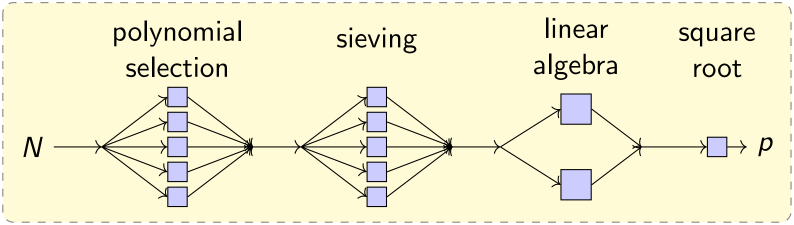
`c4.8xlarge`

- 36 virtualized cores
- two Intel Xeon E5-2666 v3 processor chips
- 60GB RAM

Pricing

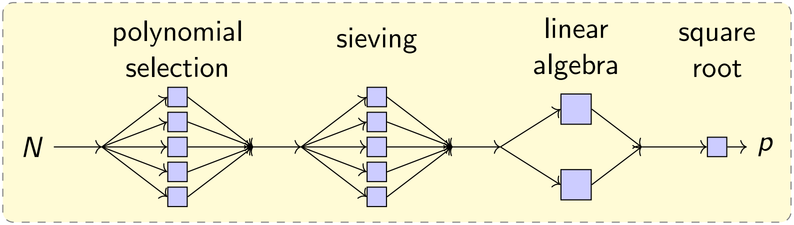
- guaranteed rate of $1.783/hr (on-demand)
- bid on unused capacity at fluctuating rate $0.35+ (spot)

The Number Field Sieve Algorithm



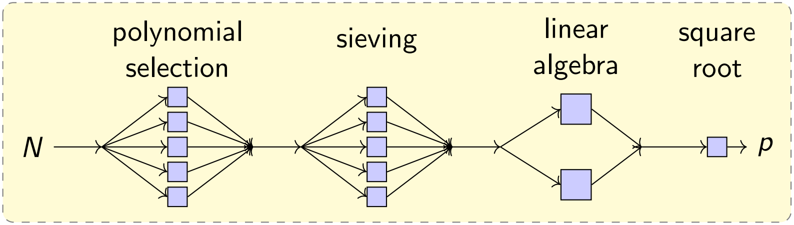
The Number Field Sieve Algorithm

- **Polynomial selection** Choose a good number field
  embarassingly parallel, 120 CPU-hours



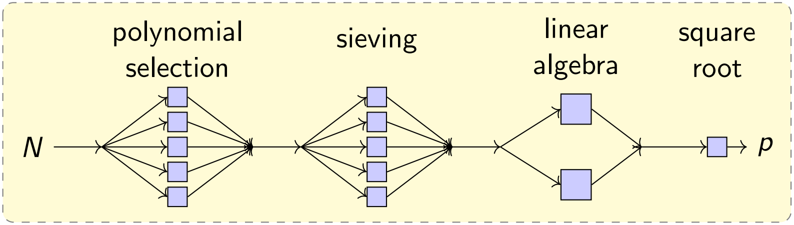
The Number Field Sieve Algorithm

- **Polynomial selection** Choose a good number field
  embarassingly parallel, 120 CPU-hours
- **Sieving** Factor small-ish integers to find algebraic relations
  embarassingly parallel, 2,800 CPU-hours



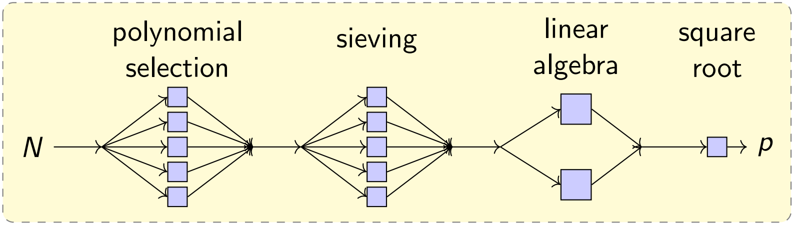
The Number Field Sieve Algorithm

- **Polynomial selection** Choose a good number field
  embarassingly parallel, 120 CPU-hours
- **Sieving** Factor small-ish integers to find algebraic relations
  embarassingly parallel, 2,800 CPU-hours
- **Linear algebra** Build matrix from relations, reduce to find squares
  semi-parallel, 250 CPU-hours



The Number Field Sieve Algorithm

- **Polynomial selection** Choose a good number field
  embarassingly parallel, 120 CPU-hours
- **Sieving** Factor small-ish integers to find algebraic relations
  embarassingly parallel, 2,800 CPU-hours
- **Linear algebra** Build matrix from relations, reduce to find squares
  semi-parallel, 250 CPU-hours
- **Square root** Take square roots and check if factor N
  mostly non-parallel, 10 CPU-minutes



Making Sieving Fast

- **Goal:** Distribute many small tasks to a compute cluster

Making Sieving Fast

- **Goal:** Distribute many small tasks to a compute cluster
- **Problems:** CADO-NFS job distribution has scaling issues

Making Sieving Fast

- **Goal:** Distribute many small tasks to a compute cluster
- **Problems:** CADO-NFS job distribution has scaling issues
- **Solution:** Replace job distribution with Slurm

Making Sieving Fast

- **Goal:** Distribute many small tasks to a compute cluster
- **Problems:** CADO-NFS job distribution has scaling issues
- **Solution:** Replace job distribution with Slurm
- **More Problems:** Cannot submit many small tasks to Slurm at once

Making Sieving Fast

- **Goal:** Distribute many small tasks to a compute cluster
- **Problems:** CADO-NFS job distribution has scaling issues
- **Solution:** Replace job distribution with Slurm
- **More Problems:** Cannot submit many small tasks to Slurm at once
- **More Solutions:** Fix with batching logic

Making Sieving Fast

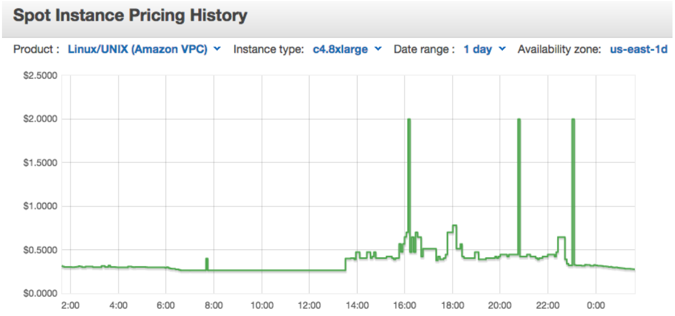
- **Goal:** Distribute many small tasks to a compute cluster
- **Problems:** CADO-NFS job distribution has scaling issues
- **Solution:** Replace job distribution with Slurm
- **More Problems:** Cannot submit many small tasks to Slurm at once
- **More Solutions:** Fix with batching logic

Now we can parallelize sieving away, right?!

Reality Check

- You can't actually launch that many spot instances at once
- Amazon runs pretty close to capacity
- On-demand instances are much more expensive

Price spikes: launching a 50-node cluster


Spot Instance Pricing History

Product : Linux/UNIX (Amazon VPC) Instance type: c4.8xlarge Date range : 1 day Availability zone: us-east-1d

Making Linear Algebra Fast

Goal: divide up large matrix into smaller grids, which must communicate periodically.

Problems:

Solutions:

Making Linear Algebra Fast

Goal: divide up large matrix into smaller grids, which must communicate periodically.

Problems:

CADO-NFS linear algebra runtime *increased* with more nodes

Solutions:

Use Msieve's implementation instead; performs better for 512-bit keys

Making Linear Algebra Fast

Goal: divide up large matrix into smaller grids, which must communicate periodically.

Problems:	Solutions:
CADO-NFS linear algebra runtime *increased* with more nodes	Use Msieve's implementation instead; performs better for 512-bit keys
High communication requirements make networking a bottleneck	Use Amazon's Enhanced Networking for 10Gbit bandwidth

Making Linear Algebra Fast

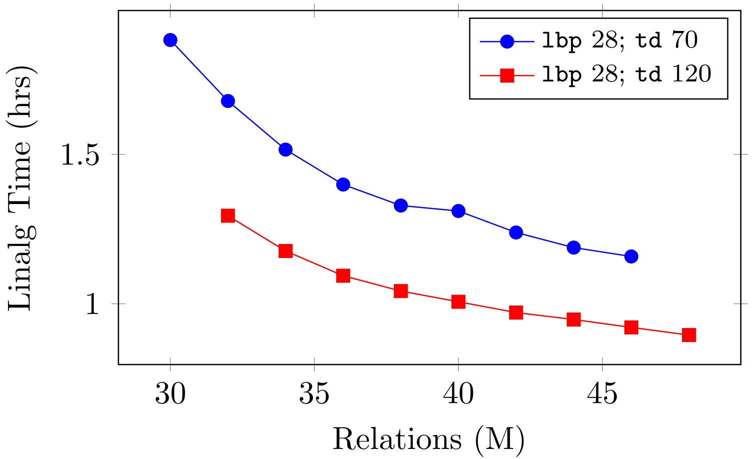
Goal: divide up large matrix into smaller grids, which must communicate periodically.

Problems:	Solutions:
CADO-NFS linear algebra runtime *increased* with more nodes	Use Msieve's implementation instead; performs better for 512-bit keys
High communication requirements make networking a bottleneck	Use Amazon's Enhanced Networking for 10Gbit bandwidth
Inter-node latency is higher than expected (150μs)	Tune implementation parameters instead

Make Linear Algebra Easier

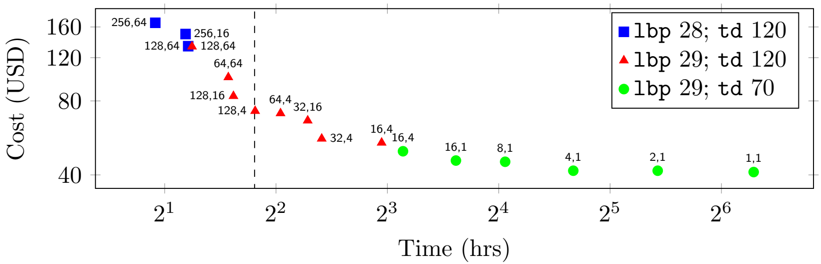
by Making Sieving Harder

Oversieving “generating excess relations”



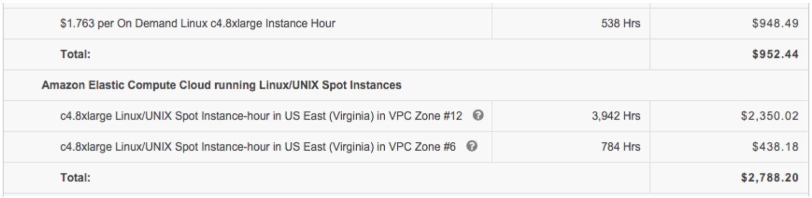
Putting it All Together

- Spend more money to make factoring faster, but with diminishing returns
- Large clusters are prone to random node failures and instability



The Cost of Research

August 2015 EC2 bill



$1.763 per On Demand Linux c4.8xlarge Instance Hour	538 Hrs	$948.49
Total:		**$952.44**
Amazon Elastic Compute Cloud running Linux/UNIX Spot Instances		
c4.8xlarge Linux/UNIX Spot Instance-hour in US East (Virginia) in VPC Zone #12	3,942 Hrs	$2,350.02
c4.8xlarge Linux/UNIX Spot Instance-hour in US East (Virginia) in VPC Zone #6	784 Hrs	$438.18
Total:		**$2,788.20**

Shoutout to our sponser: Thanks Amazon!

Is anyone still using 512-bit RSA?

Is anyone still using 512-bit RSA?

[RSA export + FREAK attack]

International Traffic in Arms Regulations [April 1, 1992 version]

```
Category XIII--Auxiliary Military Equipment ...

(1) Cryptographic (including key management) systems, equipment, assemblies,
modules, integrated circuits, components or software with the capability of
maintaining secrecy or confidentiality of information or information
systems...
```

Commerce Control List [current]

```
a.1.b.1. Factorization of integers in excess of 512 bits (e.g., RSA);
```

April 2015: FREAK attack [BDFKPSZZ 2015]: Implementation flaw; use fast 512-bit factorization to downgrade modern browsers to broken export-grade RSA.

"...we observe that 512-bit factorization is currently solvable at most in weeks..."

Who is using 512-bit RSA?

TLS measurements [scans.io]

HTTPS

March 2015: 8.9M (26.3%) HTTPS servers support `RSA_EXPORT`

September 2015: 2.6M (7.7%) HTTPS servers support `RSA_EXPORT`

Who is using 512-bit RSA?

TLS measurements [scans.io]

HTTPS

March 2015: 8.9M (26.3%) HTTPS servers support `RSA_EXPORT`

September 2015: 2.6M (7.7%) HTTPS servers support `RSA_EXPORT`

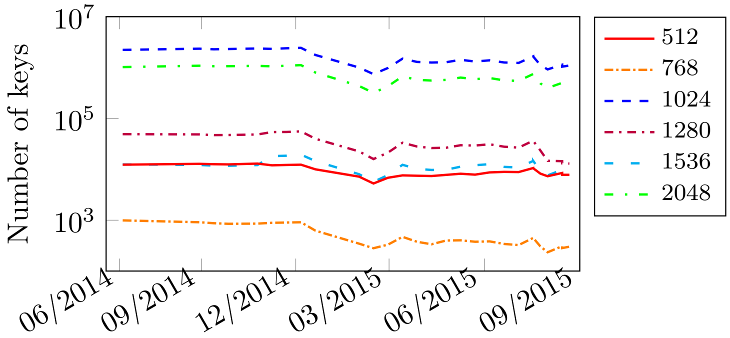
SMTP missed the memo

September 2015: 1.5M (30.8%) SMTP/StartTLS servers support `RSA_EXPORT`

DNSSEC: Domain Name System Security Extensions

[Rapid7 + SURFnet datasets + our own scans]

Key sizes are way too small



DNSSEC: Domain Name System Security Extensions

[Rapid7 + SURFnet datasets + our own scans]

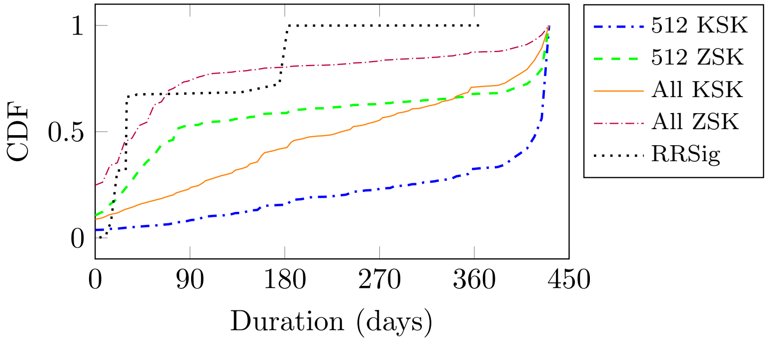
RFC 6781 [2012]

“it is estimated that most zones can safely use 1024-bit keys for at least the next ten years.”

DNSSEC: Domain Name System Security Extensions

[Rapid7 + SURFnet datasets + our own scans]

Keys are rotated infrequently



DKIM: Domain-Keys Identified Mail

[Rapid7 + SURFNET + our own scans]

Public Keys

512 bits	103 (0.9%)
384 bits	20 (0.2%)
128 bits	1 (0.0%)
Parse error	591 (5.1%)
Total	11,637

DKIM: Domain-Keys Identified Mail

[Rapid7 + SURFNET + our own scans]

Public Keys

512 bits	103 (0.9%)
384 bits	20 (0.2%)
128 bits	1 (0.0%)
Parse error	591 (5.1%)
Total	11,637

128-bit key

```
[REDACTED] bdb6389e41d8df6141acdda91a7c23c1
```

DKIM: Domain-Keys Identified Mail

[Rapid7 + SURFNET + our own scans]

Public Keys

512 bits	103 (0.9%)
384 bits	20 (0.2%)
128 bits	1 (0.0%)
Parse error	591 (5.1%)
Total	11,637

128-bit key

```
[REDACTED] bdb6389e41d8df6141acdda91a7c23c1
```

```
sage: time factor(Integer("bdb6389e41d8df6141acdda91a7c23c1",16))
CPU times: user 68.3 ms, sys: 17.3 ms, total: 85.6 ms
Wall time: 132 ms
14060786408729026139 * 17934291173672884499
```

Takeaways

- Amazon EC2 is not a traditional supercomputing platform
- Anyone can factor 512-bit RSA in $<$4 hours for \$75 on the cloud
- Use RSA responsibly: keys $\geq$ 2048 bits
- Backdoors and legal restrictions on crypto are bad

Factoring as a Service

Luke Valenta, Shaanan Cohney, Alex Liao,
Joshua Fried, Satya Bodduluri, Nadia Heninger

University of Pennsylvania

`seclab.upenn.edu/projects/faas`