RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

MATTERS NOW

SESSION ID: CRYP-T08

WHY JOHNNY THE DEVELOPER CAN'T WORK WITH PUBLIC KEY CERTIFICATES

Martin Ukrop

Usable security researcher
Centre for Research on Cryptography and Security, Masaryk University
Red Hat research cooperation

Vashek Matyas

Head of the security laboratory
Centre for Research on Cryptography and Security, Masaryk University

Simple and Solvable Error

File does not exist.

File */etc/ssl/certs/certificate1.pem*
does not exist.

Clicking OK will solve everything and create
a secure solution with no bugs or vulnerabilities.

Cancel

OK

Annoying Security Control

Something happened and you need to click OK to get on with doing things.

Security error:
Permitted subtree violation

Certificate mismatch security identification administrator communication intercept liliputian snotweasel foxtrot omegaforce.

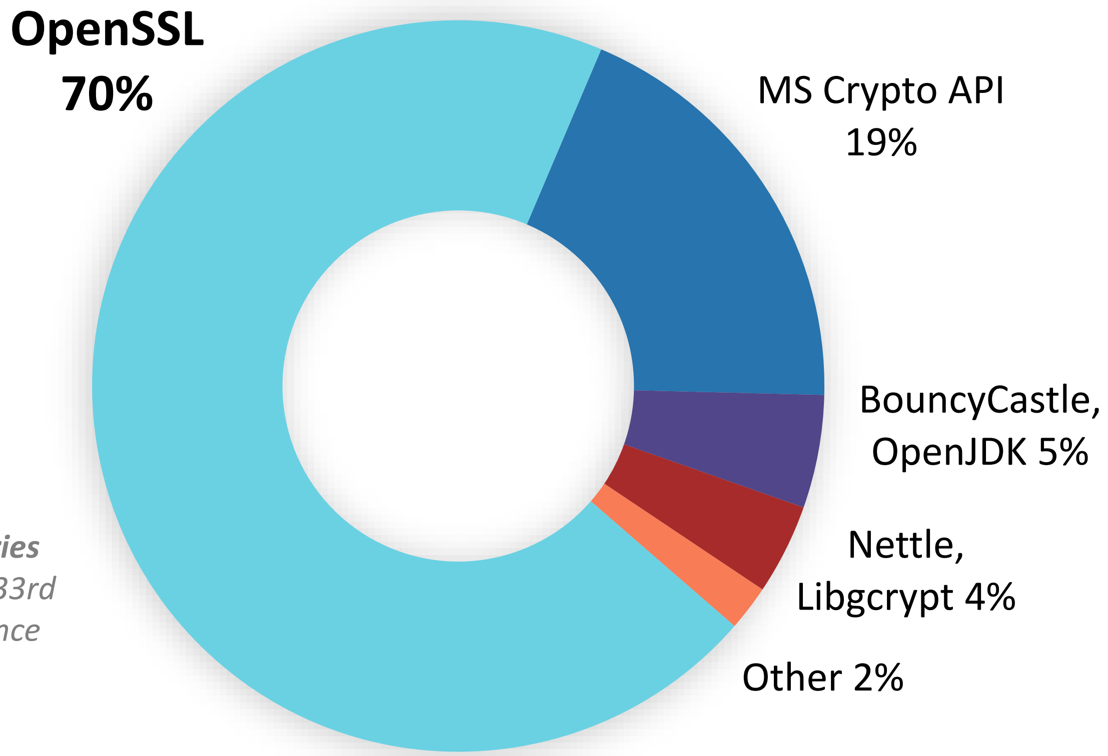
Show more technical crap

Cancel | OK

Joke adapted from
Johnatan Nightingale

Is it really that bad?

- Let's find out!
- What is the most used tool for generating certificates?



Source: Matus Nemec, Dusan Klinec, Petr Svenda, Peter Sekan and Vashek Matyas: ***"Measuring Popularity of Cryptographic Libraries in Internet-Wide Scans"***. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC'2017)*, ACM, 2017.

Q: Have you ever used OpenSSL?

```
[xukrop@styx ~]$ openssl version
OpenSSL 1.1.0g-fips  2 Nov 2017
[xukrop@styx ~]$
[xukrop@styx ~]$ openssl x509 -help
Usage: x509 [options]
Valid options are:
 -help                  Display this summary
 -inform format         Input format - default PEM (one of DER, NET or PEM)
 -in infile             Input file - default stdin
 -outform format        Output format - default PEM (one of DER, NET or PEM)
 -out outfile           Output file - default stdout
 -keyform PEM|DER       Private key format - default PEM
 -passin val            Private key password/pass-phrase source
 -serial                Print serial number value
 -subject_hash          Print subject hash value
 -issuer_hash           Print issuer hash value
 -hash                  Synonym for -subject_hash
 -subject               Print subject DN
 -issuer                Print issuer DN
 -email                 Print email address(es)
 -startdate             Set notBefore field
```

Empirical experiment in usability

- 87 participants (👦) of DEVCONF.cz

(developer conference by Red Hat Czech)

Task:

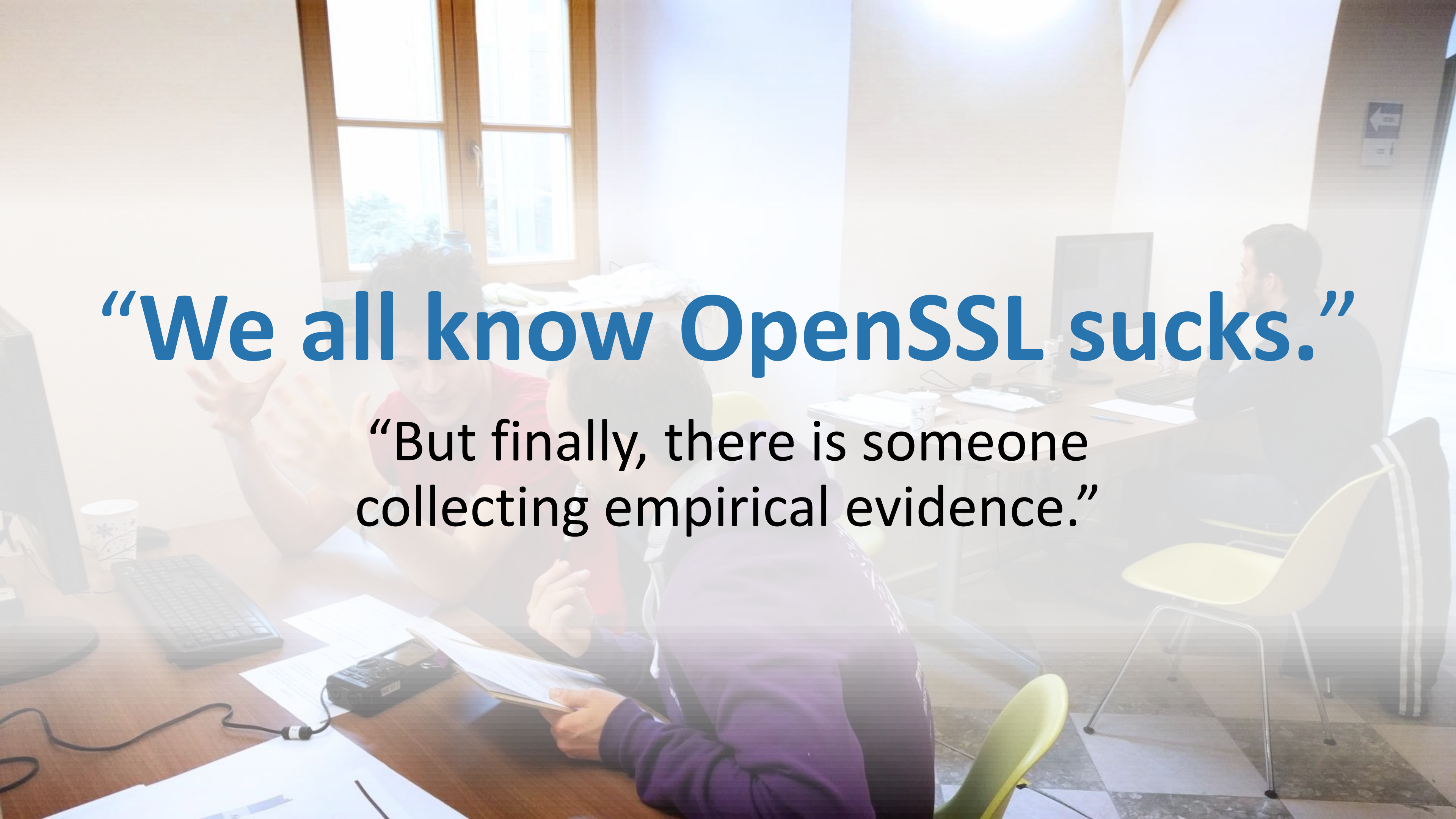
You are a software tester.
Use command line OpenSSL (v1.0.2g)

1. *… to issue a self-signed certificate.*
2. *… to validate 4 given certificates.*

Conference research booth at DEVCONF.cz

“Interacting with OpenSSL *voluntarily*?”

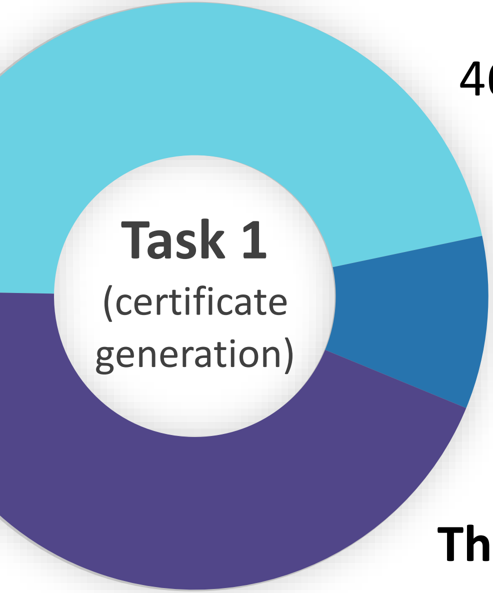
“Sorry, not even for research.”



“We all know OpenSSL sucks.”

“But finally, there is someone collecting empirical evidence.”

Task success



Succeeded
46% (39/87)

Did not succeed
10% (8/87)

Thought that they succeeded but did not
44% (37/87)

Incorrect
10% (7/72)

Success
19% (14/72)

Ignoring OS cert store
71% (51/72)

Created certificates

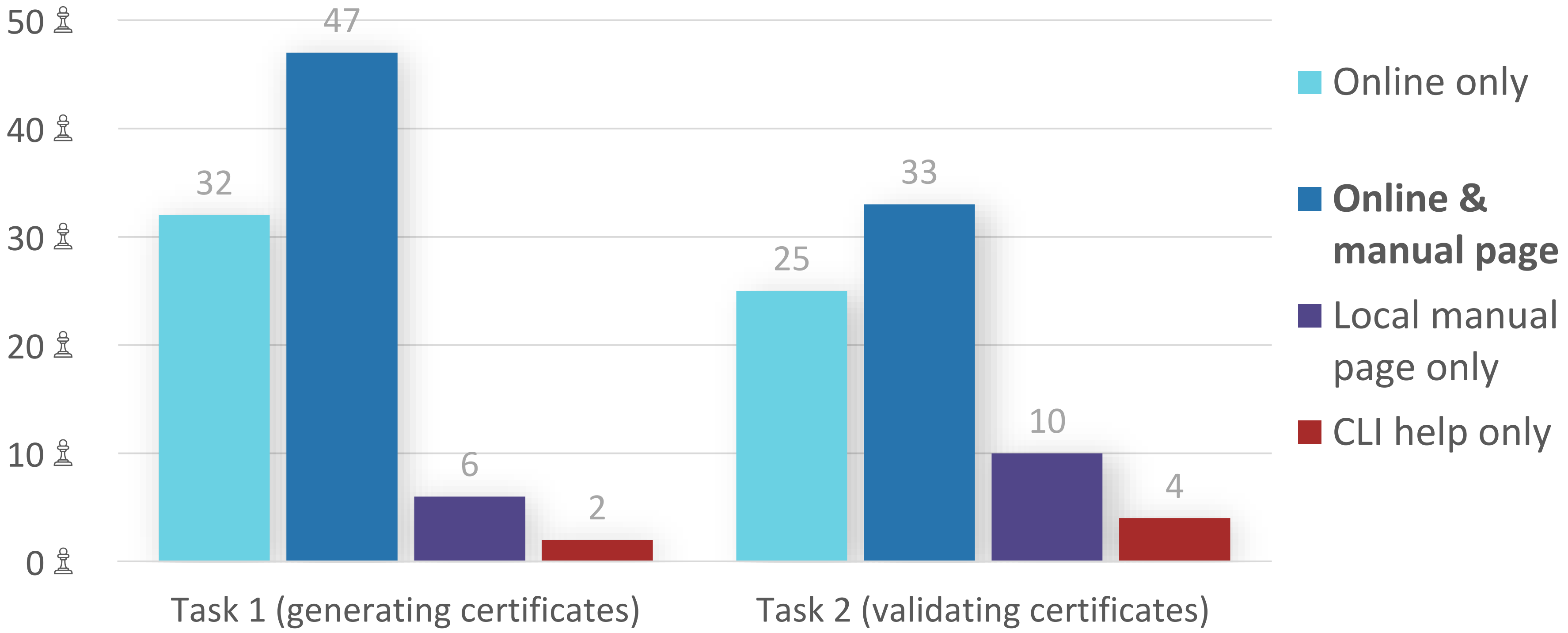
4096-bit key
20% (16/82 👨)

2048-bit key
42% (34/82 👨)

1024-bit key
38% (30/82 👨)

- Organization = “Internet Widgits Pty Ltd.”: 42% certificates (27/65 👨)
  - cca 260 000 such certificates online (Censys.io dataset 2018-02-28)

Used resources



Stack Exchange: 73% people (58/79 ♙)

stackoverflow Questions Developer Jobs Tags Users Search…

How to create a self-signed certificate with openssl?

744

504

I'm adding https support to an embedded linux device. I have tried to generate a self-signed certificate with these steps:

```
openssl req -new > cert.csr
openssl rsa -in privkey.pem -out key.pem
openssl x509 -in cert.csr -out cert.pem -req -signkey key.pem -days 1001
cat key.pem>>cert.pem
```

This works, but I get some errors with, for example, google chrome:

> This is probably not the site you are looking for!
> The site's security certificate is not trusted!

Am I missing something? Is this the correct way to build a self-signed certificate?

ssl openssl certificate ssl-certificate x509certificate

WISC knowledge base: 40% people (29/72)

University of Wisconsin KnowledgeBase

DoIT UW MyUW PEOPLE

Middleware KnowledgeBase

Search the KB... All Topics SEARCH Advanced

Quick Links

- DoIT Help Desk
- Main KB
- Middleware @ DoIT
- NetID Login Service Docs

Verifying that a Certificate is issued by a CA

How to use OpenSSL on the command line to verify that a certificate was issued by a specific CA, given that CA's certificate

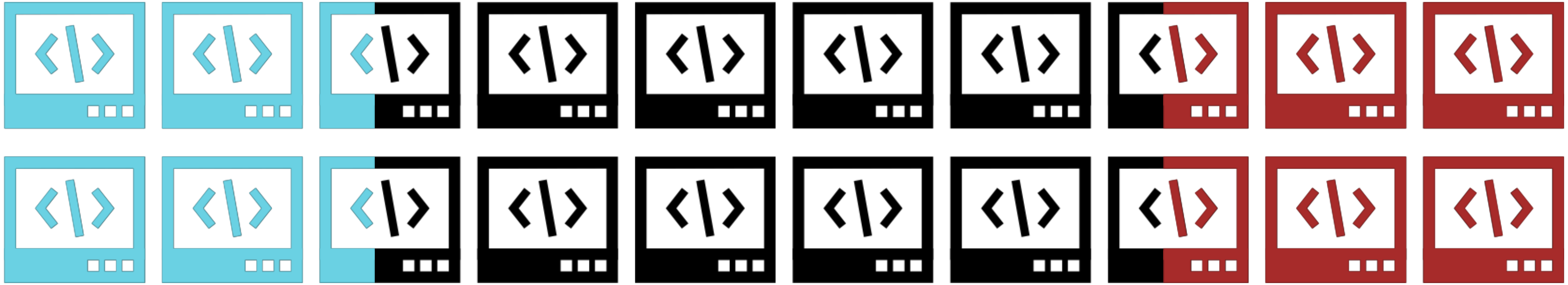
```
$ openssl verify -verbose -CAfile cacert.pem  server.crt
server.crt: OK
```

If you get any other message, the certificate was not issued by that CA.

See Also:

- How to turn a X509 Certificate in to a Certificate Signing Request
- Verifying that a Private Key Matches a Certificate

Web pages used



- Describing security implications: 23% web pages
- Explaining individual parameters: 27% web pages

- Changes after copy-paste: 9% people (8/87 👨)

Q: How to display manual page for this?

openssl verify -CAfile ca.pem cert.pem

1.	man openssl	?%	(?/53 👨)
2.	man openssl verify	28%	(15/53 👨)
3.	man openssl.verify	2%	(1/53 👨)
4.	man openssl-verify	8%	(4/53 👨)
5.	man verify	100%	(53/53 👨)

OpenSSL manual page

```
OPENSSL(1)                     OpenSSL                     OPENSSL(1)

NAME
       openssl - OpenSSL command line tool

SYNOPSIS
       openssl command [ command_opts ] [ command_args ]

       openssl list [ standard-commands | digest-commands | cipher-commands |
       cipher-algorithms | digest-algorithms | public-key-algorithms]

       openssl no-XXX [ arbitrary options ]

DESCRIPTION
       OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL
       v2/v3) and Transport Layer Security (TLS v1) network protocols and related
       cryptography standards required by them.

       The openssl program is a command line tool for using the various
       cryptography functions of OpenSSL's crypto library from the shell.  It can
       be used for
```

Has the world moved on?

Our work

- *man openssl verify* now works
- Fixed URLs for online documentation
- Research into better error messages

OpenSSL team

- High-level *help* command
- *-help* argument for each command
- Improved defaults (key lengths, …)

Takeaways I.

OpenSSL usability is poor.

(But better than other tools.)

Improvement is possible!

Takeaways II.

People may not know they failed if the tool *does not tell them*.

Takeaways III.

Documentation
(manuals, tutorials, Q&A forums, …)
matters a lot.

Stack Overflow
is a seriously used resource.

What should you do next?

USING security products?

- Ask your developers what they find unusable.
- Investigate past vulnerabilities: Caused by tool unusability?
- Report usability issues back to developers.

DEVELOPING security products?

- In your project, strive for good “developer experience” (DX).
- Ask users on usability feedback.
- Organize a usability lab study to improve your product.

RSAConference2018

NOW MATTERS

#RSAC

CARE FOR YOUR DEVELOPERS. DEVELOPER EXPERIENCE (DX) MATTERS.

Martin Ukrop
mukrop@mail.muni.cz

Centre for Research on Cryptography and Security
Masaryk University, Brno, Czech Republic

CROCS
Centre for Research on Cryptography and Security

#RSAC

RSAConference2018

San Francisco | April 16 – 20 | Moscone Center

Cryptographers' Track (CT-RSA)

MATTERS NOW

Improved Factorization of $N = p^r q^s$

Jean-Sébastien Coron,

Rina Zeitoun

Speaker: Mehdi Tibouchi

UNIVERSITÉ DU LUXEMBOURG

IDEMIA

Agenda

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$

Outline

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$

RSA Cryptosystem / Signature

RSA Key Generation

- Generate two large primes p and q
- Compute $N = p \times q$
- Select (e, d) such that $ed \equiv 1 \mod \phi(N)$

Encryption/Decryption Process

$$C \equiv m^e \mod N \quad \longrightarrow \quad m \equiv C^d \mod N$$

$\boxed{\text{Factorize } N = p \times q \quad \Rightarrow \quad \text{Break RSA}}$

State-of-the-art: Modulus $N = p^r q$

Decryption with Modulus $N = p^r q$ [Takagi98]

- For equivalent security, use a smaller prime p
  - ☞ **Decryption** becomes **faster**

[Takagi98] *Fast RSA-type cryptosystem modulo $p^k q$.* Takagi, 1998.

State-of-the-art: Modulus $N = p^r q$

Decryption with Modulus $N = p^r q$ [Takagi98]

- For equivalent security, use a smaller prime p
  - ☞ **Decryption** becomes **faster**

Vulnerability of $N = p^r q$ [BDH99]

- **BDH method:** Factoring $N = p^r q$ for Large r
  - ☞ **Condition:** $r \simeq \log q$

[Takagi98] *Fast RSA-type cryptosystem modulo $p^k q$.* Takagi, 1998.

[BDH99] *Factoring $N = p^r q$ for Large r.* Boneh, Durfee, Howgrave-Graham, 1999.

What About Modulus $N = p^r q^s$?

Decryption with Modulus $N = p^r q^s$ [LKYL00]

- Decryption is even faster
  - ☞ For an 8192-bit $N = p^2 q^3$: decryption is 15 times faster

[LKYL00] *A Generalized Takagi-Cryptosystem with a Modulus of the Form* $p^r q^s$. Lim, Kim, Yie, Lee, 2000.

What About Modulus $N = p^r q^s$?

Decryption with Modulus $N = p^r q^s$ [LKYL00]

- Decryption is even faster
  - ☞ For an 8192-bit $N = p^2 q^3$: decryption is 15 times faster

Problem: Factorization of $N = p^r q^s$

- Factorization of $N = p^r q^s$ in polynomial time
  - ☞ Left as an open problem in [BDH99]

[LKYL00] *A Generalized Takagi-Cryptosystem with a Modulus of the Form $p^r q^s$.* Lim, Kim, Yie, Lee, 2000.

[BDH99] *Factoring $N = p^r q$ for Large r.* Boneh, Durfee, Howgrave-Graham, 1999.

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks

CFRZ method

BDH method

Coppersmith method

[CFRZ16] *Factoring* $N = p^r q^s$ *for Large* r *and* s. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks

$$\begin{tikzcd}
 & \text{CFRZ method} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
\text{BDH method} \arrow[rr, Rightarrow] & & \text{Coppersmith method}
\end{tikzcd}$$

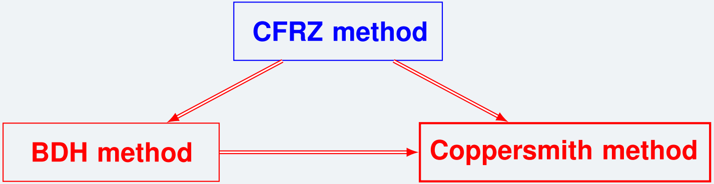
[CFRZ16] *Factoring* $N = p^r q^s$ *for Large* r *and* s. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks



[CFRZ16] *Factoring $N = p^r q^s$ for Large r and s.* Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

Outline

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$

Coppersmith's Theorem

The Problem (Univariate Modular Case)

- **Input:**
  - A polynomial $f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1 x + a_0$
  - N an integer of unknown factorization
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$

Coppersmith's Theorem

The Problem (Univariate Modular Case)

- **Input:**
  - A polynomial $f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1 x + a_0$
  - N an integer of unknown factorization
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$

Coppersmith's Theorem for the Univariate Modular case

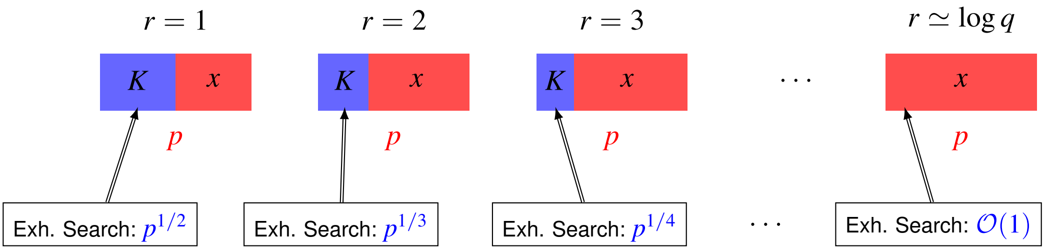
- The solutions x_0 can be found in polynomial time in $\log(N)$ if

$$|x_0| < N^{1/\delta}$$

BDH Method to factorize $N = p^r q$

☞ **Extension of Coppersmith method for unknown modulus**

- Write $p = K \cdot t + x$
- Solve $f(x) = (K \cdot t + x)^r \equiv 0 \mod p^r$



Outline

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log^3 \max(p, q))$$

Decompose r and s, Apply BDH or Coppersmith

$$N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

[CFRZ16] *Factoring $N = p^r q^s$ for Large r and s.* Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log^3 \max(p, q))$$

Decompose r and s, Apply BDH or Coppersmith

$$N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

$\Rightarrow$ Apply BDH or Coppersmith method depending on a and b

[CFRZ16] *Factoring $N = p^r q^s$ for Large r and s.* Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

[CFRZ16] Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log^3 \max(p, q))$$

Decompose r and s, Apply BDH or Coppersmith

$$N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

$\Rightarrow$ Apply BDH or Coppersmith method depending on a and b

☞ $u = \Omega(\log Q) \quad \Rightarrow \quad r = \Omega(\log^3 \max(p, q))$

[CFRZ16] *Factoring $N = p^r q^s$ for Large r and s.* Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.

[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose r and s

$$\begin{cases} r = \boxed{u} \cdot \boxed{\alpha} + \boxed{a} \\ s = \boxed{u} \cdot \boxed{\beta} + \boxed{b} \end{cases}$$

$$\underbrace{u}_{\text{large}} \quad \underbrace{\alpha, \beta}_{\text{small}} \quad \underbrace{a, b}_{\text{small}}$$

[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose r and s

$$\begin{cases} r = \boxed{u} \cdot \boxed{\alpha} + \boxed{a} \\ s = \boxed{u} \cdot \boxed{\beta} + \boxed{b} \end{cases}$$

large | small | small

☞ Use LLL

[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose r and s

$$\begin{cases} r = \boxed{u} \cdot \boxed{\alpha} + \boxed{a} \\ s = \boxed{u} \cdot \boxed{\beta} + \boxed{b} \end{cases}$$

large | small | small

☞ Use LLL

Rewrite N from new decomposition

$$N = p^r q^s = p^{u \cdot \alpha + a} q^{u \cdot \beta + b} = p^{u \cdot \alpha} q^{u \cdot \beta} p^a q^b = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose r and s

$$\begin{cases} r = \boxed{u} \cdot \boxed{\alpha} + \boxed{a} \\ s = \boxed{u} \cdot \boxed{\beta} + \boxed{b} \end{cases}$$

large | small | small

☞ Use LLL

Rewrite N from new decomposition

$$N = p^r q^s = p^{u\cdot\alpha+a} q^{u\cdot\beta+b} = p^{u\cdot\alpha} q^{u\cdot\beta} p^a q^b = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

Last Step: recover p and q

Recover P → Compute $Q = N/P^u$ → Retrieve p and q

Outline

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$

Our Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log q)$$

Our Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log q)$$

The idea: Decompose N^α instead of N

[CFRZ16]:

- Write $N = P^u Q$
- Apply BDH/Copp. on N
- Condition: $u = \Omega(\log Q)$

This paper:

- Write $N^\alpha = P^r q$
- Apply BDH on N^α
- Condition: $r = \Omega(\log q)$

Our Improvement to Factorize $N = p^r q^s$

First Step: Find α and β

Since $\gcd(r, s) = 1$, there exist $\alpha, \beta \in \mathbb{N}$ such that:

$$\alpha \cdot s - \beta \cdot r = 1$$

Our Improvement to Factorize $N = p^r q^s$

First Step: Find α and β

Since $\gcd(r, s) = 1$, there exist $\alpha, \beta \in \mathbb{N}$ such that:

$$\alpha \cdot s - \beta \cdot r = 1$$

Rewrite N^α from new decomposition

$$N^\alpha = (p^r q^s)^\alpha = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r + 1} = (p^\alpha q^\beta)^r q = P^r q$$

Our Improvement to Factorize $N = p^r q^s$

First Step: Find α and β

Since $\gcd(r, s) = 1$, there exist α, $\beta \in \mathbb{N}$ such that:

$$\alpha \cdot s - \beta \cdot r = 1$$

Rewrite N^α from new decomposition

$$N^\alpha = (p^r q^s)^\alpha = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r + 1} = (p^\alpha q^\beta)^r q = P^r q$$

Last Step: Apply BDH and recover p and q

$$\boxed{\text{Recover } P} \Longrightarrow \boxed{\text{Compute } q = N^\alpha / P^r} \Longrightarrow \boxed{\text{Compute } p = (N/q^s)^{1/r}}$$

Complexity comparison:

Time complexities for factoring $N = p^r q^s$, where $\log p \approx \log q$

Condition on $N = p^r q^s$	[CFRZ16]	New Method
$r = \Omega(\log q)$		$\mathcal{O}(\log^{12.5} N)$
$r = \Omega(\log^3 q)$	$\mathcal{O}(\log^8 N)$	$\mathcal{O}(\log^{14.25} N)$

Generalization for k prime factors

Factoring $N = \prod_{i=1}^{k} p_i^{r_i}$ for large r_i

- **Condition** to find a non-trivial factor of N in time polynomial (with $r = \max\{r_i\}$ and $p = \max\{p_i\}$):

$$r = \Omega(\log^{\theta_k} p)$$

where $$\theta_k = 2(k-1)\left(1 + \sum_{i=1}^{k-2}\prod_{j=i}^{k-2} j\right) + 1$$

k	2	3	4	5	6
θ_k in [CFRZ16]	3	17	61	257	1301
New θ_k	1	9	31	129	651

Experiments for $N = p^r q^s$ with 128-bit primes p and q

	Method	Decomposition	Bits given	Dim.	LLL_f	LLL_c	Est. time
$N = p^5q^3$	[CFRZ16]	$N = (p^2q)^3p^{-1}$	57	52	17s	3.5s	$1.6 \cdot 10^{10}$ years
	New	$N^2 = (p^2q)^5q$	**46**	**78**	**0.3h**	**29s**	**$6.5 \cdot 10^7$ years**
$N = p^7q^4$	[CFRZ16]	$N = (p^2q)^4p^{-1}$	51	57	45s	2.4s	$1.7 \cdot 10^8$ years
	New	$N^2 = (p^2q)^7q$	**43**	**92**	**1.9h**	**291s**	**$8.1 \cdot 10^7$ years**
$N = p^8q^3$	[CFRZ16]	$N = (p^2q)^4q^{-1}$	51	61	86s	4.2s	$3 \cdot 10^8$ years
	New	$N^3 = (p^3q)^8q$	**57**	**95**	**6h**	**320s**	**$1.4 \cdot 10^{12}$ years**
$N = p^9q^5$	[CFRZ16]	$N = (p^2q)^5p^{-1}$	48	61	113s	4.2s	$3.7 \cdot 10^7$ years
	New	$N^2 = (p^2q)^9q$	**43**	**108**	**3.9h**	**801s**	**$2.2 \cdot 10^8$ years**

☞ Still unpractical but asymptotically polynomial-time for large r

Conclusion and Remarks

Conclusion

- Improvement over [CFRZ16] method to factorize $N = p^r q^s$ for large r
- Generalization for moduli $N = \prod p_i^{r_i}$

Conclusion and Remarks

Conclusion

- Improvement over [CFRZ16] method to factorize $N = p^r q^s$ for large r
- Generalization for moduli $N = \prod p_i^{r_i}$

Remarks

- Much simpler than [CFRZ16]
  - Coppersmith's method is not used anymore
  - Bézout instead of LLL used to find good decomposition of r and s
- For 128-bit primes, unpractical compared to ECM factorisation
- But ECM scales exponentially while our method is polynomial
  - For large p and q, our algorithm must outpace ECM