Cryptographic readiness levels, and the impact of quantum computers

Daniel J. Bernstein

- How is crypto developed?
- How confident are we that crypto is secure?
- How do we know what a quantum computer will do?

Many stages of research from cryptographic design to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.

Cryptographic readiness levels,
and the impact
of quantum computers

Daniel J. Bernstein

- How is crypto developed?
- How confident are we that crypto is secure?
- How do we know what a quantum computer will do?

Many stages of research from cryptographic design to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.

raphic readiness levels,
impact
tum computers

. Bernstein

---

s crypto developed?

onfident are we
rypto is secure?

lo we know what a
um computer will do?

Many stages of research
from cryptographic design
to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.
6. Study
fault
7. Focus
imple
8. Focus
meeti
requi

diness levels,

uters

n

---

eveloped?

are we
cure?

w what a
ter will do?

Many stages of research
from cryptographic design
to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.

6. Study side-char
fault attacks, e
7. Focus on secur
implementation
8. Focus on imple
meeting perforr
requirements.

els,

?

Many stages of research
from cryptographic design
to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.
6. Study side-channel attack fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementation meeting performance requirements.

Many stages of research from cryptographic design to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.
6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.

Many stages of research from cryptographic design to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.
6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Many stages of research
from cryptographic design
to real-world deployment:

1. Explore space of cryptosystems.
2. Study algorithms for the attackers.
3. Focus on secure cryptosystems.
4. Study algorithms for the users.
5. Study implementations on real hardware: e.g., software for popular CPUs.
6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Getting all this right takes time:
e.g., elliptic-curve cryptography
(ECC) entered stage 1 in 1985.

ages of research
yptographic design
world deployment:

ore space of
osystems.

y algorithms for the
kers.

s on secure cryptosystems.

y algorithms for the users.

y implementations
al hardware: e.g.,
are for popular CPUs.

6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Getting all this right takes time: e.g., elliptic-curve cryptography (ECC) entered stage 1 in 1985.

What's t

Case stu
famous l

2006 Sil
and CVF
studied f
both as
problems
pure and
physics a

Best SV
by 2000:
almost a

search
c design
oyment:

of

ns for the

e cryptosystems.

ns for the users.

ntations
re: e.g.,
pular CPUs.

6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Getting all this right takes time: e.g., elliptic-curve cryptography (ECC) entered stage 1 in 1985.

What's the best a

Case study: SVP,
famous lattice pro

2006 Silverman: "
and CVP, have be
studied for more t
both as intrinsic m
problems and for a
pure and applied m
physics and crypto

Best SVP algorith
by 2000: time $2^{\Theta($
almost all dimensi

stems.

users.

Js.

6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Getting all this right takes time: e.g., elliptic-curve cryptography (ECC) entered stage 1 in 1985.

What's the best attack algo

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, S and CVP, have been intensiv studied for more than 100 ye both as intrinsic mathematic problems and for application pure and applied mathemati physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ fo almost all dimension-N lattic

6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
9. Integrate securely into real-world applications.

Getting all this right takes time: e.g., elliptic-curve cryptography (ECC) entered stage 1 in 1985.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

y side-channel attacks,
attacks, etc.

s on secure, reliable
ementations.

s on implementations
ing performance
rements.

rate securely into
world applications.

all this right takes time:
otic-curve cryptography
ntered stage 1 in 1985.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SV
today: 2

Approx
believed
0.415: 2
0.415: 2

nnel attacks,
tc.

e, reliable
ns.

mentations
mance

ely into
fications.

ght takes time:
cryptography
ge 1 in 1985.

<u>What's the best attack algorithm?</u>

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorith
today: $2^{\Theta(N)}$.

Approx c for some
believed to take ti
0.415: 2008 Nguy
0.415: 2010 Micci

ks,

ns

ime:
phy
985.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithm believed to take time $2^{(c+o($

0.415: 2008 Nguyen–Vidick.

0.415: 2010 Micciancio–Vou

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:
0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.

0.415: 2010 Micciancio–Voulgaris.

0.384: 2011 Wang–Liu–Tian–Bi.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.

0.415: 2010 Micciancio–Voulgaris.

0.384: 2011 Wang–Liu–Tian–Bi.

0.378: 2013 Zhang–Pan–Hu.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

What's the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

the best attack algorithm?

udy: SVP, the most
lattice problem.

verman: "Lattices, SVP
P, have been intensively
for more than 100 years,
intrinsic mathematical
s and for applications in
d applied mathematics,
and cryptography."

P algorithms known
: time $2^{\Theta(N \log N)}$ for
ll dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

Code-ba

Some pa
against 1

1962 Pra
1981 Or
1988 Le
1988 Le
1989 Kr
1989 St
1989 Du
1990 Co
1990 va
1991 Du
1991 Co

ttack algorithm?

the most
blem.

'Lattices, SVP
en intensively
han 100 years,
nathematical
applications in
mathematics,
ography."

ms known
$(N \log N)$ for
on-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

Code-based crypto

Some papers study
against 1978 McE

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Good
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Good

rithm?

SVP
vely
ears,
cal
ns in
cs,

r
ces.

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

Code-based cryptography

Some papers studying attac
against 1978 McEliece syste

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farr

Best SVP algorithms known today: $2^{\Theta(N)}$.

Approx c for some algorithms believed to take time $2^{(c+o(1))N}$:

0.415: 2008 Nguyen–Vidick.
0.415: 2010 Micciancio–Voulgaris.
0.384: 2011 Wang–Liu–Tian–Bi.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
0.292: 2015 Becker–Ducas–Gama–Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

Code-based cryptography

Some papers studying attacks against 1978 McEliece system:

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farrell.

P algorithms known
$2^{\Theta(N)}$.

c for some algorithms
to take time $2^{(c+o(1))N}$:
2008 Nguyen–Vidick.
2010 Micciancio–Voulgaris.
2011 Wang–Liu–Tian–Bi.
2013 Zhang–Pan–Hu.
2014 Laarhoven.
2015 Laarhoven–de Weger.
2015 Becker–Ducas–
Gama–Laarhoven.

crypto: more attack
even less understanding.

<u>Code-based cryptography</u>

Some papers studying attacks against 1978 McEliece system:

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farrell.

1993 Ch
1993 Ch
1994 var
1994 Ca
1998 Ca
1998 Ca
2008 Be
2009 Be
var
2009 Be
2009 Fir
2010 Be
2011 Ma
2011 Be
2012 Be

ms known

e algorithms
me $2^{(c+o(1))N}$:
en–Vidick.
ancio–Voulgaris.
g–Liu–Tian–Bi.
g–Pan–Hu.
hoven.
hoven–de Weger.
er–Ducas–
rhoven.

ore attack
understanding.

Code-based cryptography

Some papers studying attacks against 1978 McEliece system:

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farrell.

1993 Chabanne–C
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Ch
1998 Canteaut–Ch
1998 Canteaut–Se
2008 Bernstein–La
2009 Bernstein–La
van Tilborg.
2009 Bernstein (p
2009 Finiasz–Send
2010 Bernstein–La
2011 May–Meurer
2011 Becker–Coro
2012 Becker–Joux

ns
$1))N$:

ulgaris.

n–Bi.

Veger.

nding.

Code-based cryptography

Some papers studying attacks against 1978 McEliece system:

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farrell.
1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peter
2009 Bernstein–Lange–Peter
van Tilborg.
2009 Bernstein (post-quantu
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peter
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Me

Code-based cryptography

Some papers studying attacks against 1978 McEliece system:

1962 Prange.
1981 Omura.
1988 Lee–Brickell.
1988 Leon.
1989 Krouk.
1989 Stern.
1989 Dumer.
1990 Coffey–Goodman.
1990 van Tilburg.
1991 Dumer.
1991 Coffey–Goodman–Farrell.
1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.

sed cryptography

apers studying attacks
1978 McEliece system:

ange.
nura.
e–Brickell.
on.
ouk.
ern.
umer.
ffey–Goodman.
n Tilburg.
umer.
ffey–Goodman–Farrell.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.

2013 Be
Me
2015 Ma

ography

ying attacks
liece system:

lman.

lman–Farrell.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.

2013 Bernstein–Je
Meurer (post
2015 May–Ozerov

ks

m:

ell.

1993 Chabanne–Courteau.

1993 Chabaud.

1994 van Tilburg.

1994 Canteaut–Chabanne.

1998 Canteaut–Chabaud.

1998 Canteaut–Sendrier.

2008 Bernstein–Lange–Peters.

2009 Bernstein–Lange–Peters–van Tilborg.

2009 Bernstein (post-quantum).

2009 Finiasz–Sendrier.

2010 Bernstein–Lange–Peters.

2011 May–Meurer–Thomae.

2011 Becker–Coron–Joux.

2012 Becker–Joux–May–Meurer.

2013 Bernstein–Jeffery–Lang

Meurer (post-quantum

2015 May–Ozerov.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.
2013 Bernstein–Jeffery–Lange–Meurer (post-quantum).
2015 May–Ozerov.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.
2013 Bernstein–Jeffery–Lange–Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$.
Here $C_0 \approx 0.7418860694$.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.
2013 Bernstein–Jeffery–Lange–Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$. Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today: $(C_0 + o(1))b^2(\lg b)^2$.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
1998 Canteaut–Sendrier.
2008 Bernstein–Lange–Peters.
2009 Bernstein–Lange–Peters–van Tilborg.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2011 Becker–Coron–Joux.
2012 Becker–Joux–May–Meurer.
2013 Bernstein–Jeffery–Lange–Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$. Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today: $(C_0 + o(1))b^2(\lg b)^2$.

Key size needed for 2^b security vs. best quantum attack known today: $(4C_0 + o(1))b^2(\lg b)^2$.

abanne–Courteau.
abaud.
n Tilburg.
nteaut–Chabanne.
nteaut–Chabaud.
nteaut–Sendrier.
rnstein–Lange–Peters.
rnstein–Lange–Peters–
n Tilborg.
rnstein (post-quantum).
niasz–Sendrier.
rnstein–Lange–Peters.
ay–Meurer–Thomae.
cker–Coron–Joux.
cker–Joux–May–Meurer.

2013 Bernstein–Jeffery–Lange–
Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$. Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today: $(C_0 + o(1))b^2(\lg b)^2$.

Key size needed for 2^b security vs. best quantum attack known today: $(4C_0 + o(1))b^2(\lg b)^2$.

What is

Quantur
stores m
can effic
"Hadam
"control

Making
is the m
comput

Combine
to comp
. . . "Sim
. . . "Sho
. . . "Gro

ourteau.

habanne.

habaud.

endrier.

ange–Peters.

ange–Peters–

ost-quantum).

drier.

ange–Peters.

–Thomae.

n–Joux.

–May–Meurer.

2013 Bernstein–Jeffery–Lange–
Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$.
Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today:
$(C_0 + o(1))b^2(\lg b)^2$.

Key size needed for 2^b security vs. best quantum attack known today: $(4C_0 + o(1))b^2(\lg b)^2$.

What is a quantu

Quantum compute
stores many "qubi
can efficiently perf
"Hadamard gate",
"controlled NOT g

Making these ins
is the main goal
computer engine

Combine these ins
to compute "Toffo
... "Simon's algo
... "Shor's algorit
... "Grover's algo

rs.

rs–

um).

rs.

urer.

2013 Bernstein–Jeffery–Lange–
Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$.
Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today: $(C_0 + o(1))b^2(\lg b)^2$.

Key size needed for 2^b security vs. best quantum attack known today: $(4C_0 + o(1))b^2(\lg b)^2$.

What is a quantum compute

Quantum computer type 1 (
stores many "qubits";
can efficiently perform
"Hadamard gate", "T gate"
"controlled NOT gate".

Making these instructions
is the main goal of quantu
computer engineering.

Combine these instructions
to compute "Toffoli gate";
. . . "Simon's algorithm";
. . . "Shor's algorithm";
. . . "Grover's algorithm"; et

2013 Bernstein–Jeffery–Lange–Meurer (post-quantum).
2015 May–Ozerov.

Key size needed for 2^b security vs. best attack known in 1978: $(C_0 + o(1))b^2(\lg b)^2$.
Here $C_0 \approx 0.7418860694$.

Key size needed for 2^b security vs. best pre-quantum attack known today: $(C_0 + o(1))b^2(\lg b)^2$.

Key size needed for 2^b security vs. best quantum attack known today: $(4C_0 + o(1))b^2(\lg b)^2$.

What is a quantum computer?

Quantum computer type 1 (QC1): stores many "qubits"; can efficiently perform "Hadamard gate", "T gate", "controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions to compute "Toffoli gate";
... "Simon's algorithm";
... "Shor's algorithm";
... "Grover's algorithm"; etc.

rnstein–Jeffery–Lange–
eurer (post-quantum).
ay–Ozerov.

needed for 2^b security
attack known in 1978:
$(1))b^2(\lg b)^2$.
≈ 0.7418860694.

needed for 2^b security
pre-quantum attack
oday:
$(1))b^2(\lg b)^2$.

needed for 2^b security
quantum attack known
$4C_0 + o(1))b^2(\lg b)^2$.

What is a quantum computer?

Quantum computer type 1 (QC1):
stores many "qubits";
can efficiently perform
"Hadamard gate", "T gate",
"controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions
to compute "Toffoli gate";
... "Simon's algorithm";
... "Shor's algorithm";
... "Grover's algorithm"; etc.

Quantur
stores a
efficientl
laws of q
with as r

This is t
quantum
by 1982
physics v

effery–Lange–
t-quantum).
.

or 2^b security
own in 1978:
$)^2$.
860694.

or 2^b security
um attack

$)^2$.

or 2^b security
attack known
$.))b^2(\lg b)^2$.

What is a quantum computer?

Quantum computer type 1 (QC1):
stores many "qubits";
can efficiently perform
"Hadamard gate", "T gate",
"controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions
to compute "Toffoli gate";
. . . "Simon's algorithm";
. . . "Shor's algorithm";
. . . "Grover's algorithm"; etc.

Quantum compute
stores a simulated
efficiently simulate
laws of quantum p
with as much accu

This is the origina
quantum compute
by 1982 Feynman
physics with comp

ge–
).

rity
78:

rity
(

rity
own
2.

What is a quantum computer?

Quantum computer type 1 (QC1):
stores many "qubits";
can efficiently perform
"Hadamard gate", "*T* gate",
"controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions
to compute "Toffoli gate";
... "Simon's algorithm";
... "Shor's algorithm";
... "Grover's algorithm"; etc.

Quantum computer type 2 (
stores a simulated universe;
efficiently simulates the
laws of quantum physics
with as much accuracy as de

This is the original concept
quantum computers introdu
by 1982 Feynman "Simulati
physics with computers".

What is a quantum computer?

Quantum computer type 1 (QC1): stores many "qubits"; can efficiently perform "Hadamard gate", "*T* gate", "controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions to compute "Toffoli gate";
... "Simon's algorithm";
... "Shor's algorithm";
... "Grover's algorithm"; etc.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

What is a quantum computer?

Quantum computer type 1 (QC1):
stores many "qubits";
can efficiently perform
"Hadamard gate", "*T* gate",
"controlled NOT gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions
to compute "Toffoli gate";
. . . "Simon's algorithm";
. . . "Shor's algorithm";
. . . "Grover's algorithm"; etc.

Quantum computer type 2 (QC2):
stores a simulated universe;
efficiently simulates the
laws of quantum physics
with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2.
Partial proof: see, e.g.,
2011 Jordan–Lee–Preskill
"Quantum algorithms for
quantum field theories".

a quantum computer?

m computer type 1 (QC1):
any "qubits";
iently perform
ard gate", "T gate",
led NOT gate".

these instructions work
nain goal of quantum-
er engineering.

e these instructions
ute "Toffoli gate";
non's algorithm";
or's algorithm";
over's algorithm"; etc.

Quantum computer type 2 (QC2):
stores a simulated universe;
efficiently simulates the
laws of quantum physics
with as much accuracy as desired.

This is the original concept of
quantum computers introduced
by 1982 Feynman "Simulating
physics with computers".

General belief: any QC1 is a QC2.
Partial proof: see, e.g.,
2011 Jordan–Lee–Preskill
"Quantum algorithms for
quantum field theories".

Quantur
efficientl
that any
can com

m computer?

er type 1 (QC1):
ts";
form
. "*T* gate",
gate".

structions work
of quantum-
ering.

tructions
oli gate";
rithm";
thm";
orithm"; etc.

Quantum computer type 2 (QC2):
stores a simulated universe;
efficiently simulates the
laws of quantum physics
with as much accuracy as desired.

This is the original concept of
quantum computers introduced
by 1982 Feynman "Simulating
physics with computers".

General belief: any QC1 is a QC2.
Partial proof: see, e.g.,
2011 Jordan–Lee–Preskill
"Quantum algorithms for
quantum field theories".

Quantum compute
efficiently compute
that any physical c
can compute effici

er?

(QC1):

,

work

um-

c.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2. Partial proof: see, e.g., 2011 Jordan–Lee–Preskill "Quantum algorithms for quantum field theories".

Quantum computer type 3 (

efficiently computes anythin

that any physical computer

can compute efficiently.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2. Partial proof: see, e.g., 2011 Jordan–Lee–Preskill "Quantum algorithms for quantum field theories".

Quantum computer type 3 (QC3): efficiently computes anything that any physical computer can compute efficiently.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2. Partial proof: see, e.g., 2011 Jordan–Lee–Preskill "Quantum algorithms for quantum field theories".

Quantum computer type 3 (QC3): efficiently computes anything that any physical computer can compute efficiently.

General belief: any QC2 is a QC3. Argument for belief: any physical computer must follow the laws of quantum physics, so a QC2 can efficiently simulate any physical computer.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2. Partial proof: see, e.g., 2011 Jordan–Lee–Preskill "Quantum algorithms for quantum field theories".

Quantum computer type 3 (QC3): efficiently computes anything that any physical computer can compute efficiently.

General belief: any QC2 is a QC3. Argument for belief: any physical computer must follow the laws of quantum physics, so a QC2 can efficiently simulate any physical computer.

General belief: any QC3 is a QC1. Argument for belief: look, we're building a QC1.