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.
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.
6. Study side-channel attacks, fault attacks, etc.
7. Focus on secure, reliable implementations.
8. Focus on implementations meeting performance requirements.
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.
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.
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.
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.

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 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.
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.

What’s the best attack algorithm?

Case study: SVP, the most famous lattice problem.

2006 Silverman: “Lattices, SVP and CVP, have been studied for more than a century 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.
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.

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.
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.
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.

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.
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.

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

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.

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

Best SVP algorithms known by 2000: time $2^{\Theta(N \log N)}$ for almost all dimension-$N$ lattices.
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.
- 0.378: 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.
- 0.378: 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.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.
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.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.
- 0.298: 2015 Laarhoven–de Weger.

Lattice crypto: more attack avenues; even less understanding.
What's the best attack algorithm?

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.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.

Lattice crypto: more attack avenues; even less understanding.

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 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.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.

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.
What's the best attack algorithm?

The most famous lattice problem.

"Lattices, SVP 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 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.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.
- 0.298: 2015 Laarhoven–de Weger.

Lattice crypto: more attack avenues; even less understanding.

Code-based cryptography
Some papers studying attacks against 1978 McEliece system:
- 1962 Prange.
- 1981 Omura.
- 1988 Leon.
- 1989 Krouk.
- 1989 Stern.
- 1989 Dumer.
- 1990 Coffey–Goodman.
- 1990 van Tilburg.
- 1991 Dumer.
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.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.
- 0.298: 2015 Laarhoven–de Weger.

Lattice crypto: more attack avenues; even less understanding.

Code-based cryptography

Some papers studying attacks against 1978 McEliece system:

- 1962 Prange.
- 1981 Omura.
- 1988 Leon.
- 1989 Krouk.
- 1989 Stern.
- 1989 Dumer.
- 1990 Coffey–Goodman.
- 1990 van Tilburg.
- 1991 Dumer.
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.
0.378: 2013 Zhang–Pan–Hu.
0.337: 2014 Laarhoven.
0.298: 2015 Laarhoven–de Weger.

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

Approx \(c\) for some algorithms believed to take time \(2^{(c+o(1))N}\):
- 2008 Nguyen–Vidick.
- 2010 Micciancio–Voulgaris.
- 2013 Zhang–Pan–Hu.
- 2014 Laarhoven.

Lattice crypto: more attack avenues; even less understanding.

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1962</td>
<td>Prange</td>
</tr>
<tr>
<td>1981</td>
<td>Omura</td>
</tr>
<tr>
<td>1988</td>
<td>Lee–Brickell</td>
</tr>
<tr>
<td>1988</td>
<td>Leon</td>
</tr>
<tr>
<td>1989</td>
<td>Krouk</td>
</tr>
<tr>
<td>1989</td>
<td>Stern</td>
</tr>
<tr>
<td>1989</td>
<td>Dumer</td>
</tr>
<tr>
<td>1990</td>
<td>Coffey–Goodman</td>
</tr>
<tr>
<td>1990</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1991</td>
<td>Dumer</td>
</tr>
<tr>
<td>1991</td>
<td>Coffey–Goodman–Farrell</td>
</tr>
</tbody>
</table>

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
- 1994 van Tilburg
- 1994 Canteaut–Sendrier
- 1998 Canteaut–Sendrier
- 2008 Bernstein–Lange–Peters
- 2009 Bernstein (post-quantum)
- 2009 Bernstein–Lange–Peters–van Tilborg
- 2009 Bernstein–Lange–Peters–van Tilborg
- 2009 Bernstein–Lange–Peters–van Tilborg
- 2009 Bernstein–Lange–Peters
- 2009 Bernstein–Lange–Peters
- 2010 Bernstein–Lange–Peters
- 2011 May–Meurer–Thomae
- 2011 Becker–Coron–Joux
- 2012 Becker–Joux–May–Meurer
- 2012 Becker–Joux–May–Meurer
- 2012 Becker–Joux–May–Meurer
- 2012 Becker–Joux–May–Meurer
Best SVP algorithms known today: $2^{\Theta(N)}$.

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

- 2008 Nguyen–Vidick.
- 2010 Micciancio–Voulgaris.
- 2013 Zhang–Pan–Hu.
- 2014 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 Leon.
- 1989 Krouk.
- 1989 Stern.
- 1989 Dumer.
- 1990 Coffey–Goodman.
- 1990 van Tilburg.
- 1991 Dumer.
- 1993 Chabanne–Courteau.
- 1993 Chabaud.
- 1994 van Tilburg.
- 1994 Canteaut–Chabanne.
- 1998 Canteaut–Chabanne.
- 2009 Bernstein (post-quantum).
- 2009 Finiasz–Sendrier.
- 2010 Bernstein–Lange–van Tilburg.
- 2011 May–Meurer–Thomae.
Best SVP algorithms known today: $2^{\Theta(N)}$.

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

- 2008 Nguyen–Vidick.
- 2010 Micciancio–Voulgaris.
- 2013 Zhang–Pan–Hu.
- 2014 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 Leon.
- 1989 Krouk.
- 1989 Stern.
- 1989 Dumer.
- 1990 Coffey–Goodman.
- 1990 van Tilburg.
- 1991 Dumer.
- 1993 Chabanne–Courteau.
- 1993 Chabaud.
- 1994 van Tilburg.
- 1994 Canteaut–Chabanne.
- 1998 Canteaut–Chabaud.
- 2009 Bernstein (post-quantum).
- 2009 Finiasz–Sendrier.
- 2010 Bernstein–Lange–Peters.
- 2011 May–Meurer–Thomae.
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.

1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1962</td>
<td>Prange</td>
</tr>
<tr>
<td>1981</td>
<td>Omura</td>
</tr>
<tr>
<td>1988</td>
<td>Lee–Brickell</td>
</tr>
<tr>
<td>1988</td>
<td>Leon</td>
</tr>
<tr>
<td>1989</td>
<td>Krouk</td>
</tr>
<tr>
<td>1989</td>
<td>Stern</td>
</tr>
<tr>
<td>1989</td>
<td>Dumer</td>
</tr>
<tr>
<td>1990</td>
<td>Coffey–Goodman</td>
</tr>
<tr>
<td>1990</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1991</td>
<td>Dumer</td>
</tr>
<tr>
<td>1991</td>
<td>Coffey–Goodman–Farrell</td>
</tr>
<tr>
<td>1993</td>
<td>Chabanne–Courteau</td>
</tr>
<tr>
<td>1993</td>
<td>Chabaud</td>
</tr>
<tr>
<td>1994</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1994</td>
<td>Canteaut–Chabanne</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Chabaud</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Sendrier</td>
</tr>
<tr>
<td>2008</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein–Lange–Peters–vtilborg</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein (post-quantum)</td>
</tr>
<tr>
<td>2009</td>
<td>Finiasz–Sendrier</td>
</tr>
<tr>
<td>2010</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2011</td>
<td>May–Meurer–Thomae</td>
</tr>
<tr>
<td>2011</td>
<td>Becker–Coron–Joux</td>
</tr>
<tr>
<td>2012</td>
<td>Becker–Joux–May–Meurer</td>
</tr>
<tr>
<td>2013</td>
<td>Bernstein–Jeffery–Lange–Meurer</td>
</tr>
<tr>
<td>2015</td>
<td>May–Ozerov</td>
</tr>
<tr>
<td>2020</td>
<td>Bernstein–Lange–Peters–vtilborg</td>
</tr>
<tr>
<td>2021</td>
<td>Becker–Joux–May–Meurer</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1962</td>
<td>Prange</td>
</tr>
<tr>
<td>1981</td>
<td>Omura</td>
</tr>
<tr>
<td>1988</td>
<td>Lee–Brickell</td>
</tr>
<tr>
<td>1988</td>
<td>Leon</td>
</tr>
<tr>
<td>1989</td>
<td>Krouk</td>
</tr>
<tr>
<td>1989</td>
<td>Stern</td>
</tr>
<tr>
<td>1989</td>
<td>Dumer</td>
</tr>
<tr>
<td>1990</td>
<td>Coffey–Goodman</td>
</tr>
<tr>
<td>1990</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1991</td>
<td>Dumer</td>
</tr>
<tr>
<td>1991</td>
<td>Coffey–Goodman–Farrell</td>
</tr>
<tr>
<td>1993</td>
<td>Chabanne–Courteau</td>
</tr>
<tr>
<td>1993</td>
<td>Chabaud</td>
</tr>
<tr>
<td>1994</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1994</td>
<td>Canteaut–Chabanne</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Chabaud</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Sendrier</td>
</tr>
<tr>
<td>2008</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein–Lange–Peters–van Tilborg</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein (post-quantum)</td>
</tr>
<tr>
<td>2009</td>
<td>Finiasz–Sendrier</td>
</tr>
<tr>
<td>2010</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2011</td>
<td>May–Meurer–Thomae</td>
</tr>
<tr>
<td>2011</td>
<td>Becker–Coron–Joux</td>
</tr>
<tr>
<td>2012</td>
<td>Becker–Joux–May–Meurer</td>
</tr>
<tr>
<td>2013</td>
<td>Bernstein–Jeffery–Lange–Meurer (post-quantum)</td>
</tr>
<tr>
<td>2015</td>
<td>May–Ozerov</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>1962</td>
<td>Prange</td>
</tr>
<tr>
<td>1981</td>
<td>Omura</td>
</tr>
<tr>
<td>1988</td>
<td>Lee–Brickell</td>
</tr>
<tr>
<td>1988</td>
<td>Leon</td>
</tr>
<tr>
<td>1989</td>
<td>Krouk</td>
</tr>
<tr>
<td>1989</td>
<td>Stern</td>
</tr>
<tr>
<td>1989</td>
<td>Dumer</td>
</tr>
<tr>
<td>1990</td>
<td>Coffey–Goodman</td>
</tr>
<tr>
<td>1990</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1991</td>
<td>Dumer</td>
</tr>
<tr>
<td>1991</td>
<td>Coffey–Goodman–Farrell</td>
</tr>
<tr>
<td>1993</td>
<td>Chabanne–Courteau</td>
</tr>
<tr>
<td>1993</td>
<td>Chabaud</td>
</tr>
<tr>
<td>1994</td>
<td>van Tilburg</td>
</tr>
<tr>
<td>1994</td>
<td>Canteaut–Chabanne</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Chabaud</td>
</tr>
<tr>
<td>1998</td>
<td>Canteaut–Sendrier</td>
</tr>
<tr>
<td>2008</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein–Lange–Peters–van Tilborg</td>
</tr>
<tr>
<td>2009</td>
<td>Bernstein (post-quantum)</td>
</tr>
<tr>
<td>2009</td>
<td>Finiasz–Sendrier</td>
</tr>
<tr>
<td>2010</td>
<td>Bernstein–Lange–Peters</td>
</tr>
<tr>
<td>2011</td>
<td>May–Meurer–Thomae</td>
</tr>
<tr>
<td>2011</td>
<td>Becker–Coron–Joux</td>
</tr>
<tr>
<td>2012</td>
<td>Becker–Joux–May–Meurer</td>
</tr>
<tr>
<td>2013</td>
<td>Bernstein–Jeffery–Lange–Meurer (post-quantum)</td>
</tr>
<tr>
<td>2015</td>
<td>May–Ozerov</td>
</tr>
</tbody>
</table>
1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2015 May–Ozerov.
1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.

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.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.
2015 May–Ozerov.

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

Key size needed for $2^b$ security vs. best pre-quantum attack known today:
$\left(C_0 + o(1)\right)b^2(\lg b)^2$. 
1993 Chabanne–Courteau.
1993 Chabaud.
1994 van Tilburg.
1994 Canteaut–Chabanne.
1998 Canteaut–Chabaud.
2009 Bernstein (post-quantum).
2009 Finiasz–Sendrier.
2010 Bernstein–Lange–Peters.
2011 May–Meurer–Thomae.

2015 May–Ozerov.

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

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

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

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.

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:

\[
(4C_0 + o(1)) b^2 (\lg b)^2.
\]
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.
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.
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.”
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”;
\ldots “Simon’s algorithm”;
\ldots “Shor’s algorithm”;
\ldots “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”.

Key size needed for 2\textsuperscript{b} security vs. best attack known in 1978: \( (C_0 + o(1)) b^2 \lg b^2 \).

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

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”.

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”.

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

Instructions work of quantum-computer engineering.

Instructions: "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".

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.
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.