Lattice-based cryptography: Episode V: the ring strikes back

Daniel J. Bernstein
University of Illinois at Chicago

---

**Crypto 1999 Nguyen**: “At Crypto ’97, Goldreich, Goldwasser and Halevi proposed a public-key cryptosystem based on the closest vector problem in a lattice, which is known to be NP-hard. We show that ... the problem of decrypting ciphertexts can be reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.”
Lattice-based cryptography: Episode V: the ring strikes back

Daniel J. Bernstein
University of Illinois at Chicago

1999 Nguyen: “At Crypto '97, Goldreich, Goldwasser and
Halevi proposed a public-key cryptosystem based on the closest
vector problem in a lattice, which is known to be NP-hard. We
show that the problem of decrypting ciphertexts can be
reduced to a special closest vector problem which is much easier
than the general problem. As an application, we solved four out
of the five numerical challenges proposed on the Internet by the
authors of the cryptosystem. At least two of those four
challenges were conjectured to be intractable. We discuss ways
to prevent the flaw, but conclude that, even modified, the scheme
cannot provide sufficient security without being impractical.”

Fix would “probably need dimension ≥ 400” for security:
“Public key ≈ 1.8 Mbytes.”

system breakable with 20MB keys.
Lattice-based cryptography:
Episode V:
the ring strikes back
Daniel J. Bernstein
University of Illinois at Chicago

At Crypto '97, Goldreich, Goldwasser and Halevi proposed a public-key cryptosystem based on the closest vector problem in a lattice, which is known to be NP-hard. We show that the problem of decrypting ciphertexts can be reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

Fix would “probably need dimension ≥ 400” for security: “Public key ≈ 1.8 Mbytes.”

Lattice-based cryptography: Episode V: the ring strikes back
Daniel J. Bernstein
University of Illinois at Chicago

At Crypto '97, Goldreich, Goldwasser and Halevi proposed a public-key cryptosystem based on the closest vector problem in a lattice, which is known to be NP-hard. We show that the problem of decrypting ciphertexts can be reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

Fix would “probably need dimension \( \geq 400 \)” for security:

“Public key \( \approx 1.8 \) Mbytes”.  

Crypto 1998 Nguyen–Stern: 
“Provably secure” Ajtai–Dwork system breakable with 20MB keys.
reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.”

Fix would “probably need dimension \( \geq 400 \)” for security: “Public key \( \approx 1.8 \) Mbytes”.

**Crypto 1998 Nguyen–Stern:** “Provably secure” Ajtai–Dwork system breakable with 20MB keys.
reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.”

Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.

**Crypto 1998 Nguyen–Stern:**
“Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers. Typical parameters: 1MB key for $>2^{128}$ *post-quantum* security.
reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges posed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical."

Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.

**Crypto 1998 Nguyen–Stern:** “Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers. Typical parameters: 1MB key for $>2^{128}$ post-quantum security.

2017.05: Lattice student adds the following text to Wikipedia page: “Lattice-based cryptography”:

“Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”
reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the cryptosystem. Those four were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.


Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers.

Typical parameters: 1MB key for $2^{128}$ post-quantum security.

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”
reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.

**Crypto 1998 Nguyen–Stern:** “Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers. Typical parameters: 1MB key for $2^{128}$ post-quantum security.

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”
Fix would “probably need dimension \( \geq 400 \)” for security: “Public key \( \approx 1.8 \) Mbytes”.

**Crypto 1998 Nguyen–Stern:**
“Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers. Typical parameters: 1MB key for \( >2^{128} \) *post-quantum* security.

---

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently *the primary* candidates for post-quantum cryptography.”
Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.

**Crypto 1998 Nguyen–Stern:**
“Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers.

Typical parameters: 1MB key for $2^{128}$ post-quantum security.

---

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”:

“Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”

— [citation needed]
Fix would “probably need dimension $\geq 400$” for security: “Public key $\approx 1.8$ Mbytes”.

**Crypto 1998 Nguyen–Stern:**
“Provably secure” Ajtai–Dwork system breakable with 20MB keys.

Compare to 1978 McEliece code-based cryptosystem: much more stable security story through dozens of attack papers.

Typical parameters: 1MB key for $>2^{128}$ post-quantum security.

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”

— [citation needed]

2016.07: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.
and “probably need dimension \( \geq 400 \) for security: public key \( \approx 1.8 \) Mbytes”.


Compare to 1978 McEliece based cryptosystem: more stable security story through dozens of attack papers.

Typical parameters: 1MB key for \( \gtrsim 2^{128} \) post-quantum security.

2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”

— [citation needed]

2016.07: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.
Fix would "probably need dimension $\geq 400$" for security: "Public key $\approx 1.8$ Mbytes".

- **McEliece system**: much more stable security story through dozens of attack papers. Typical parameters: 1MB key for $2^{128}$ post-quantum security.

---

- **2017.05**: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”:
  “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”
  — [citation needed]

- **2016.07**: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites.

Google sent only a few KB for public keys, ciphertexts.
How can lattice-based crypto work within a few KB?
Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP.

**Uses lattice-based crypto.**
2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.” — [citation needed]

2016.07: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.

Google sent only a few KB for public keys, ciphertexts. How can lattice-based crypto work within a few KB? Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.
2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.” — [citation needed]

2016.07: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.

Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB?

Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.
2017.05: Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”: “Lattice-based constructions are currently the primary candidates for post-quantum cryptography.”

— [citation needed]

2016.07: Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.

Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB?
Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” $\text{SVP}_\gamma$.

2. Use ideal lattices.
   Hope that the extra structure doesn’t damage security.
Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB?

Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.

2. Use ideal lattices. Hope that the extra structure doesn’t damage security.


Define $R = \mathbb{Z}[x]/(x^{503} - 1)$. Elements of $R$ are polynomials $c_0 + c_1x + \cdots + c_{502}x^{502}$ with integer coefficients.

To multiply two polynomials in $R$, multiply them, replace $x^{503}$ with $1$, replace $x^{504}$ with $x$; etc.

e.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400}) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500}$ in $R$. 

Lattice student adds the following text to Wikipedia page “Lattice-based cryptography”:

Lattice-based constructions are currently the primary candidates for post-quantum cryptography.” [citation needed]

Google rolls out large-scale experiment with post-quantum crypto between Chrome and some Google sites. Uses lattice-based crypto.
Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB? Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.

2. Use ideal lattices. Hope that the extra structure doesn’t damage security.


Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$.

Elements of $R$ are $c_0 + c_1 x + c_2 x^2 + \cdots$ with integer coefficients.

To multiply in $R$: multiply polynomials; replace $x^{503}$ with $1$; replace $x^{504}$ with $x$; etc.

e.g.: $(x^{100} + x^{300})(x^{200} + 7 x^{400}) = x^{300} + 8 x^{500} + 7 x^{197} + x^{300} + \cdots$
Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB?
Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP\(_\gamma\).

2. **Use ideal lattices.**
   Hope that the extra structure doesn’t damage security.


Define \( R \) as the ring \( \mathbb{Z}[x]/(x^{503} - 1) \).

Elements of \( R \) are polynomials \( c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502} \) with integer coefficients \( c_j \).

To multiply in \( R \):
multiply polynomials;
replace \( x^{503} \) with 1;
replace \( x^{504} \) with \( x \); etc.

E.g.: \((x^{100} + x^{300})(x^{200} + 7) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500} \) in \( R \).
Google sent only a few KB for public keys, ciphertexts.

How can lattice-based crypto work within a few KB?

Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to “well-studied” SVP. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.

2. Use ideal lattices.
Hope that the extra structure doesn’t damage security.


Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$.

Elements of $R$ are polynomials $c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502}$ with integer coefficients $c_j$.

To multiply in $R$:
 multiply polynomials;
 replace $x^{503}$ with 1;
 replace $x^{504}$ with $x$; etc.
 e.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400})$
 $= x^{300} + 8x^{500} + 7x^{700}$
 $= 7x^{197} + x^{300} + 8x^{500}$ in $R$. 
Google sent only a few KB for public keys, ciphertexts. How can lattice-based crypto work within a few KB? Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to "well-studied" SVP\(\gamma\). See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.

2. Use ideal lattices. Hope that the extra structure doesn't damage security.


Define \( R \) as the ring \( \mathbb{Z}[x]/(x^{503} - 1) \).

Elements of \( R \) are polynomials \( c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502} \) with integer coefficients \( c_j \).

To multiply in \( R \):
- multiply polynomials;
- replace \( x^{503} \) with 1;
- replace \( x^{504} \) with \( x \); etc.
- e.g.: \( (x^{100} + x^{300})(x^{200} + 7x^{400}) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500} \) in \( R \).
Google sent only a few KB for public keys, ciphertexts. How can lattice-based crypto work within a few KB?

Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to "well-studied" SVP\(_\gamma\).
   

2. Use ideal lattices.
   
   Hope that the extra structure doesn’t damage security.


Define \( R \) as the ring \( \mathbb{Z}[x]/(x^{503} - 1) \).

Elements of \( R \) are polynomials
\[
c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502}
\]
with integer coefficients \( c_j \).

To multiply in \( R \):

- multiply polynomials;
- replace \( x^{503} \) with 1;
- replace \( x^{504} \) with \( x \); etc.

\[
e.g.: (x^{100} + x^{300})(x^{200} + 7x^{400}) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500} \text{ in } R.
\]

Define \( q = 2048 \).

Alice’s public key: \( A \in R \) with coefficients in \( \{0, 1, \ldots, q - 1\} \).

This is \( 503 \cdot 11 = 5533 \) bits.
Google sent only a few KB for public keys, ciphertexts. How can lattice-based crypto work within a few KB? Combine two ingredients:

1. Do not take key sizes large enough for theorems to connect to "well-studied" \textit{SVP}'s. See, e.g., 2016 Chatterjee–Koblitz–Menezes–Sarkar.

2. Use ideal lattices. Hope that the extra structure doesn't damage security.

---


Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$.

Elements of $R$ are polynomials $c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502}$ with integer coefficients $c_j$.

To multiply in $R$: multiply polynomials; replace $x^{503}$ with 1; replace $x^{504}$ with $x$; etc.

e.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400}) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500}$ in $R$.

Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Define \( R \) as the ring \( \mathbb{Z}[x]/(x^{503} - 1) \).

Elements of \( R \) are polynomials \( c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502} \) with integer coefficients \( c_j \).

To multiply in \( R \):
multiply polynomials;
replace \( x^{503} \) with 1;
replace \( x^{504} \) with \( x \); etc.

E.g.: \((x^{100} + x^{300})(x^{200} + 7x^{400})\)
\[= x^{300} + 8x^{500} + 7x^{700}\]
\[= 7x^{197} + x^{300} + 8x^{500} \text{ in } R.\]

Define \( q = 2048 \).

Alice’s public key: \( A \in R \) with coefficients in \( \{0, 1, \ldots, q - 1\} \).
This is \( 503 \cdot 11 = 5533 \) bits.
Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$. Elements of $R$ are polynomials $c_0 + c_1 x + c_2 x^2 + \cdots + c_{502} x^{502}$ with integer coefficients $c_j$.

To multiply in $R$: multiply polynomials; replace $x^{503}$ with 1; replace $x^{504}$ with $x$; etc.

E.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400}) = x^{300} + 8x^{500} + 7x^{700} = 7x^{197} + x^{300} + 8x^{500}$ in $R$.

Define $q = 2048$. Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients: e.g., all coefficients in $\{-1, 0, 1\}$.

Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$.

Elements of $R$ are polynomials $c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502}$ with integer coefficients $c_j$.

To multiply in $R$:
multiply polynomials;
replace $x^{503}$ with 1;
replace $x^{504}$ with $x$; etc.

e.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400})$
$= x^{300} + 8x^{500} + 7x^{700}$
$= 7x^{197} + x^{300} + 8x^{500}$ in $R$.

Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients:
e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$:
multiply $A$ by $b$ in $R$; add $c$;
reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients:
e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$:
multiply $A$ by $b$ in $R$; add $c$;
reduce each coefficient modulo $q$
to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$.
This is also 5533 bits.


Define $R$ as the ring $\mathbb{Z}[x]/(x^{503} - 1)$.

Elements of $R$ are polynomials $c_0 + c_1x + c_2x^2 + \cdots + c_{502}x^{502}$ with integer coefficients $c_j$.

To multiply in $R$:
multiply polynomials;
replace $x^{503}$ with 1;
replace $x^{504}$ with $x$; etc.
e.g.: $(x^{100} + x^{300})(x^{200} + 7x^{400})$
$= x^{300} + 8x^{500} + 7x^{700}$
$= 7x^{197} + x^{300} + 8x^{500}$ in $R$.
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in \{0, 1, \ldots, q - 1\}. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients: e.g., all coefficients in \{-1, 0, 1\}.

Bob computes $Ab + c \mod q$: multiply $A$ by $b$ in $R$; add $c$; reduce each coefficient modulo $q$ to the range \{0, 1, \ldots, q - 1\}.

Bob sends $Ab + c \mod q$. This is also 5533 bits.

Alice generated $A = 3 \frac{a}{d}$ in $R = \mathbb{Z} / q\mathbb{Z}$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$.

"Quotient NTRU" (new name), used in original NTRU design:
Define \( q = 2048 \).

Alice’s public key: \( A \in R \) with coefficients in \( \{0, 1, \ldots, q - 1\} \).
This is \( 503 \cdot 11 = 5533 \) bits.

Bob generates random \( b, c \in R \) with small coefficients:
e.g., all coefficients in \( \{-1, 0, 1\} \).

Bob computes \( Ab + c \mod q \):
multiply \( A \) by \( b \) in \( R \); add \( c \);
reduce each coefficient modulo \( q \)
to the range \( \{0, 1, \ldots, q - 1\} \).

Bob sends \( Ab + c \mod q \).
This is also 5533 bits.
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients:

e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$: multiply $A$ by $b$ in $R$; add $c$; reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$.

This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R=q$ for small random $a, d$ (with suitable invertibility):

i.e., $dA - 3a \mod q = 0$. 
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients: e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$: multiply $A$ by $b$ in $R$; add $c$; reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$. This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$. 
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients:

e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$:
multiply $A$ by $b$ in $R$; add $c$;
reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$.
This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility):
i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.
Alice computes $dC \mod q$,
i.e., $3ab + dc \mod q$. 
Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients: e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$: multiply $A$ by $b$ in $R$; add $c$; reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$. This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.

Alice computes $dc \mod q$, i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$, using smallness of $a, b, d, c$.

Alice computes $dc$, deduces $c$, deduces $b$. 

Alice generates $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.

Alice computes $dc \mod q$, i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$, using smallness of $a, b, d, c$.

Alice computes $dc$, deduces $c$, deduces $b$. 

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.

Alice computes $dc \mod q$, i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$, using smallness of $a, b, d, c$.

Alice computes $dc$, deduces $c$, deduces $b$. 

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility): i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.

Alice computes $dc \mod q$, i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$, using smallness of $a, b, d, c$.

Alice computes $dc$, deduces $c$, deduces $b$.
Define $q = 2048$.

Alice's public key: $A \in R$ with
coefficients in $\{0, 1, \ldots, q - 1\}$.

This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$
with small coefficients:
all coefficients:
coefficients in $\{-1, 0, 1\}$.

Bob computes $Ab + c \mod q$:
multiply $A$ by $b$ in $R$; add $c$;
reduce each coefficient modulo $q$
to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $Ab + c \mod q$.
This is also 5533 bits.

"Quotient NTRU" (new name),
used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$
for small random $a, d$
(with suitable invertibility):
i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.
Alice computes $dC \mod q$,
i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$,
using smallness of $a, b, d, c$.
Alice computes $dc$,
deduces $c$, deduces $b$.

"Product NTRU" (new name),
2010 Lyubashevsky–Peikert–Regev:
Everyone knows random $G \in R$.

Alice generated $A = aG + d \mod q$
for small random $a, d$.


Define $q = 2048$.

Alice’s public key: $A \in R$ with coefficients in $\{0, 1, \ldots, q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b, c \in R$ with small coefficients:
e.g., all coefficients in $\{-1, 0, 1\}$.

Bob computes $A b + c \mod q$:
multiply $A$ by $b$ in $R$; add $c$; reduce each coefficient modulo $q$ to the range $\{0, 1, \ldots, q - 1\}$.

Bob sends $A b + c \mod q$. This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility):
i.e., $dA - 3a \mod q = 0$.

Alice receives $C = A b + c \mod q$.

Alice computes $dC \mod q$,
i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$,
using smallness of $a, b, d, c$.

Alice computes $dc$,
deduces $c$, deduces $b$.

“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$.
Alice generated $A = aG + d \mod q$ for small random $a, d$. 

Define $q = 2048$.

Alice's public key: $A \in \mathbb{R}$ with coefficients in $\{0; 1; \ldots; q - 1\}$. This is $503 \cdot 11 = 5533$ bits.

Bob generates random $b; c \in \mathbb{R}$ with small coefficients:

- e.g., all coefficients in $\{-1; 0; 1; \ldots\}$.

Bob computes $A b + c \mod q$:

- multiply $A$ by $b$ in $\mathbb{R}$;
- add $c$;
- reduce each coefficient modulo $q$ to the range $\{0; 1; \ldots; q - 1\}$.

Bob sends $A b + c \mod q$.

This is also 5533 bits.

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3 a/d$ in $R/q$ for small random $a, d$

(with suitable invertibility):

- i.e., $d A - 3 a \mod q = 0$.

Alice receives $C = A b + c \mod q$.

Alice computes $d C \mod q$,

- i.e., $3 a b + d c \mod q$.

Alice reconstructs $3 a b + d c$,

using smallness of $a, b, d, c$.

Alice computes $d c$,

- deduces $c$, deduces $b$.

“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in \mathbb{R}$.

Alice generated $A = a G + d \mod q$ for small random $a, d$. 

Alice computes $d C \mod q$,

- i.e., $3 a b + d c \mod q$.

Alice reconstructs $3 a b + d c$,

using smallness of $a, b, d, c$.

Alice computes $d c$,

- deduces $c$, deduces $b$. 

“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$
(with suitable invertibility):
i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.
Alice computes $dC \mod q$,
i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$,
using smallness of $a, b, d, c$.
Alice computes $dc$,
deduces $c$, deduces $b$.

“Product NTRU” (new name),
2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$.
Alice generated $A = aG + d \mod q$ for small random $a, d$.
“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$
(with suitable invertibility):

i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.

Alice computes $dC \mod q$,

i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$,
using smallness of $a, b, d, c$.

Alice computes $dc$,

deduces $c$, deduces $b$.

“Product NTRU” (new name),  
2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$.

Alice generated $A = aG + d \mod q$ for small random $a, d$.

Bob sends $B = Gb + e \mod q$ and $C = m + Ab + c \mod q$ where $b, c, e$ are small and each coefficient of $m$ is 0 or $q/2$. 
“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$
(with suitable invertibility):
i.e., $dA - 3a \mod q = 0$.

Alice receives $C = Ab + c \mod q$.
Alice computes $dC \mod q$,
i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$,
using smallness of $a, b, d, c$.
Alice computes $dc$,
deduces $c$, deduces $b$.

“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$.
Alice generated $A = aG + d \mod q$
for small random $a, d$.

Bob sends $B = Gb + e \mod q$
and $C = m + Ab + c \mod q$
where $b, c, e$ are small and each
coefficient of $m$ is 0 or $q/2$.

Alice computes $C - aB \mod q$,
i.e., $m + db + c - ae \mod q$.

Alice reconstructs $m$,
using smallness of $d, b, c, a, e$. 
“Quotient NTRU” (new name),
used in original NTRU design:
Alice generated \( A = 3a/d \) in \( R/q \)
for small random \( a, d \)
(suitable invertibility):
\[ dA - 3a \mod q = 0. \]
Alice receives \( C = Ab + c \mod q \).
Alice computes \( dC \mod q \),
\[ 3ab + dc \mod q. \]
Alice reconstructs \( 3ab + dc \),
using smallness of \( a, b, d, c \).
Alice computes \( dc \),
deduces \( c \), deduces \( b \).

“Product NTRU” (new name),
2010 Lyubashevsky–Peikert–Regev:
Everyone knows random \( G \in R \).
Alice generated \( A = aG + d \mod q \)
for small random \( a, d \).
Bob sends \( B = Gb + e \mod q \)
and \( C = m + Ab + c \mod q \)
where \( b, c, e \) are small and each
coefficient of \( m \) is 0 or \( q/2 \).
Alice computes \( C - aB \mod q \),
i.e., \( m + db + c - ae \mod q \).
Alice reconstructs \( m \),
using smallness of \( d, b, c, a, e \).

Lattice view: Define \( L \) as
the set of pairs \( (v;w) \in R \times R \)
such that
\[ vG - w \mod q = 0. \]
“Quotient NTRU” (new name), used in original NTRU design:

Alice generated $A = 3a/d$ in $R/q$ for small random $a, d$ (with suitable invertibility):

$$dA - 3a \mod q = 0.$$  

Alice receives $C = Ab + c \mod q$.

Alice computes $dC \mod q$, i.e., $3ab + dc \mod q$.

Alice reconstructs $3ab + dc$, using smallness of $a; b; d; c$.

Alice computes $dc$, deduces $c$, deduces $b$.

“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$.

Alice generated $A = aG + d \mod q$ for small random $a, d$.

Bob sends $B = Gb + e \mod q$ and $C = m + Ab + c \mod q$ where $b, c, e$ are small and each coefficient of $m$ is 0 or $q/2$.

Alice computes $C - aB \mod q$, i.e., $m + db + c - ae \mod q$.

Alice reconstructs $m$, using smallness of $d, b, c, a, e$.

Lattice view: Define $L$ as the set of pairs $(v;w) \in R \times R$ such that $vG - w \mod q = 0$. 

8

9

10
“Quotient NTRU” (new name),
used in original NTRU design:
Alice generated
\( A = 3a = d \) in \( R = \mathbb{Z}_q \)
for small random \( a, d \)
(with suitable invertibility):
\[ i.e., \quad dA - 3a \mod q = 0. \]
Alice receives \( C = Ab + c \mod q \).
Alice computes \( dc \mod q \),
i.e., \( 3ab + dc \mod q \).
Alice reconstructs \( 3ab + dc \),
using smallness of \( a, b, d, c \).
Alice computes \( dc \),
deduces \( c \),
deduces \( b \).

“Product NTRU” (new name),
2010 Lyubashevsky–Peikert–Regev:
Everyone knows random \( G \in \mathbb{Z}_q \).
Alice generated \( A = aG + d \mod q \)
for small random \( a, d \).
Bob sends \( B = Gb + e \mod q \)
and \( C = m + Ab + c \mod q \)
where \( b, c, e \) are small and each
coefficient of \( m \) is 0 or \( q/2 \).
Alice computes \( C - aB \mod q \),
i.e., \( m + db + c - ae \mod q \).
Alice reconstructs \( m \),
using smallness of \( d, b, c, a, e \).

Lattice view: Define \( L \) as
the set of pairs \( (v, w) \in \mathbb{Z} \times \mathbb{Z} \)
such that \( vG - w \mod q = 0 \).
“Product NTRU” (new name),
2010 Lyubashevsky–Peikert–Regev:
Everyone knows random $G \in R$.
Alice generated $A = aG + d \mod q$ for small random $a, d$.
Bob sends $B = Gb + e \mod q$
and $C = m + Ab + c \mod q$ where $b, c, e$ are small and each coefficient of $m$ is 0 or $q/2$.
Alice computes $C - aB \mod q$,
i.e., $m + db + c - ae \mod q$.
Alice reconstructs $m$, using smallness of $d, b, c, a, e$.

Lattice view: Define $L$ as the set of pairs $(v, w) \in R \times R$
such that $vG - w \mod q = 0$. 
“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in R$. Alice generated $A = aG + d \mod q$ for small random $a, d$.

Bob sends $B = Gb + e \mod q$ and $C = m + Ab + c \mod q$ where $b, c, e$ are small and each coefficient of $m$ is 0 or $q/2$.

Alice computes $C - aB \mod q$, i.e., $m + db + c - ae \mod q$.

Alice reconstructs $m$, using smallness of $d, b, c, a, e$.

Lattice view: Define $L$ as the set of pairs $(v, w) \in R \times R$ such that $vG - w \mod q = 0$.

e.g. $(a, A - d) \in L$. $(0, A)$ is close to a lattice point.

Try to find close lattice point. Breaks both Product NTRU and Quotient NTRU.
“Product NTRU” (new name), 2010 Lyubashevsky–Peikert–Regev:

Everyone knows random $G \in \mathbb{R}$.

Alice generated $A = aG + d \pmod{q}$ for small random $a, d$.

Bob sends $B = Gb + e \pmod{q}$ and $C = m + Ab + c \pmod{q}$ where $b, c, e$ are small and each coefficient of $m$ is 0 or $q/2$.

Alice computes $C - aB \pmod{q}$, i.e., $m + db + c - ae \pmod{q}$.

Alice reconstructs $m$, using smallness of $d, b, c, a, e$.

Lattice view: Define $L$ as the set of pairs $(v, w) \in \mathbb{R} \times \mathbb{R}$ such that $vG - w \pmod{q} = 0$.

e.g. $(a, A - d) \in L$.

$(0, A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

(“Ring-LWE”: arbitrary reuse.)

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.
“Product NTRU” (new name),
2010 Lyubashevsky–Peikert–Regev:
Everyone knows random $G \in R$.
Alice generated $A = aG + d \mod q$
for small random $a, d$.

Bob sends $B = Gb + e \mod q$
and $C = m + Ab + c \mod q$
where $b, c, e$ are small and each
coefficient of $m$ is 0 or $q/2$.

Alice computes $C - aB \mod q$,
i.e., $m + db + c - ae \mod q$.

Alice reconstructs $m$,
using smallness of $d, b, c, a, e$.

Lattice view: Define $L$ as
the set of pairs $(v, w) \in R \times R$
such that $vG - w \mod q = 0$.
e.g. $(a, A - d) \in L$.
$(0, A)$ is close to a lattice point.

Try to find close lattice point.
Breaks both Product NTRU
and Quotient NTRU.

Try to exploit reuse of $b$
for faster Product NTRU attack.
(“Ring-LWE”: arbitrary reuse.)

Try to exploit $A = 3a/d$ structure
for faster Quotient NTRU attack.

2013 Lyubashevsky–Peikert–Regev:
All of the algebraic
and algorithmic tools (including
quantum computation) that we
employ ::: can also be brought
to bear against SVP and other
problems on ideal lattices. Yet
despite considerable effort
no significant progress in attacking
these problems has been made.
The best-known algorithms for
ideal lattices perform essentially
no better than their generic
counterparts, both in theory and
in practice.
Lattice view: Define $L$ as the set of pairs $(v, w) \in \mathbb{R} \times \mathbb{R}$ such that $vG - w \mod q = 0$.

E.g. $(a, A - d) \in L$.

$(0, A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

(“Ring-LWE”: arbitrary reuse.)

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.

Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ ... can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”
Lattice view: Define $L$ as the set of pairs $(v, w) \in R \times R$ such that $vG - w \mod q = 0$.

e.g. $(a, A - d) \in L$.

$(0, A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

(“Ring-LWE”: arbitrary reuse.)

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.

2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ ... can also be brought to bear against SVP and other problems on ideal lattices. Yet, despite considerable effort, no significant progress in attacking these problems has been made.

The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”
Lattice view: Define $L$ as the set of pairs $(v, w) \in R \times R$ such that $vG - w \mod q = 0$.

E.g. $(a, A - d) \in L$.

(0, A) is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

(“Ring-LWE”: arbitrary reuse.)

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.

2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ ... can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”
Lattice view: Define $L$ as the set of pairs $(v, w) \in \mathbb{R} \times \mathbb{R}$ such that $vG - w \mod q = 0$.

e.g. $(a; A - d) \in L$.

$(0; A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

```
("Ring-LWE": arbitrary reuse.)
```

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.

---

2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ ... can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”

---

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:

STOC 2009 Gentry

“Fully homomorphic encryption using ideal lattices”.

PKC 2010 Smart–Vercauteren.

Eurocrypt 2011 Gentry–Halevi.

etc.

Multilinear maps: e.g.,

Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.
Define $L$ as the set of pairs $(v, w) \in \mathbb{R} \times \mathbb{R}$ such that $vG - w \pmod{q} = 0$.

e.g. $(a; A - d) \in L$.

$(0; A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

("Ring-LWE": arbitrary reuse.)

Try to exploit $A = 3a/d$ structure for faster Quotient NTRU attack.

2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ . . . can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:

STOC 2009 Gentry

“Fully homomorphic encryption using ideal lattices.”

PKC 2010 Smart–Vercauteren.

Eurocrypt 2011 Gentry–Halevi.

etc.

Multilinear maps:

Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices.”
Lattice view: Define $L$ as the set of pairs $(v;w) \in \mathbb{R} \times \mathbb{R}$ such that $vG - w \mod q = 0$.

e.g. $(a;A - d) \in L$.

$(0;A)$ is close to a lattice point.

Try to find close lattice point.

Breaks both Product NTRU and Quotient NTRU.

Try to exploit reuse of $b$ for faster Product NTRU attack.

("Ring-LWE": arbitrary reuse.)

Try to exploit $A = \frac{3a}{d}$ structure for faster Quotient NTRU attack.

2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ … can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:

STOC 2009 Gentry

“Fully homomorphic encryption using ideal lattices”.

PKC 2010 Smart–Vercauteren.

Eurocrypt 2011 Gentry–Halevi.

etc.

Multilinear maps: e.g.,

Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.
2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) that we employ ... can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.”

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi. etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

“All of the algebraic and algorithmic tools (including quantum computation) that we ... can also be brought against SVP and other problems on ideal lattices. Yet considerable effort, no significant progress in attacking problems has been made. Well-known algorithms for lattices perform essentially no better than their generic counterparts, both in theory and practice.”

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption: STOC 2009 Gentry “Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.

Multilinear maps: e.g., Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.
etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is 
broken by quantum algorithms
for typical “cyclotomic rings”.

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.
etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

Many more NTRU variants (often not crediting NTRU).
All of the algebraic and algorithmic tools (including quantum computation) that we employ can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best-known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice.

Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.
etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is
broken by quantum algorithms for typical “cyclotomic rings”.
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption: STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.
etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.
Many more NTRU variants (often not crediting NTRU).


Multilinear maps: e.g., Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is **broken** by quantum algorithms for typical “cyclotomic rings”.

First stage in attack: SODA 2016 Biasse–Song fast quantum algorithm to compute $gR \mapsto ug$ with $u \in R^*$. Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum $R \mapsto R^*$ algorithm.
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.

equal.
Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is **broken** by quantum algorithms for typical “cyclotomic rings”.

First stage in attack:
SODA 2016 Biasse–Song
fast quantum algorithm to compute $g R \mapsto u g$ with $u \in R^*$.

Builds upon STOC 2014
Eisenträger–Hallgren–Kitaev–Song
quantum $R \mapsto R^*$ algorithm.

Older pre-quantum algorithms take subexponential time.
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption: STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.

Eurocrypt 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi “Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is **broken** by quantum algorithms for typical “cyclotomic rings”.

First stage in attack: SODA 2016 Biasse–Song
fast quantum algorithm to compute $gR \mapsto ug$ with $u \in R^*$.

Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song
quantum $R \mapsto R^*$ algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd
fast pre-quantum algorithm for typical cyclotomic ring
to compute $ug \mapsto$ short $g$. 
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
- STOC 2009 Gentry
- "Fully homomorphic encryption using ideal lattices".
- PKC 2010 Smart–Vercauteren.
- Eurocrypt 2011 Gentry–Halevi.
- etc.

Multilinear maps: e.g.,
- Eurocrypt 2013 Garg–Gentry–Halevi "Candidate multilinear maps from ideal lattices".

STOC 2009 Gentry system is **broken** by quantum algorithms for typical “cyclotomic rings”.

First stage in attack:
- SODA 2016 Biasse–Song
  - fast quantum algorithm to compute \( gR \mapsto ug \) with \( u \in R^* \).

Builds upon STOC 2014
- Eisenträger–Hallgren–Kitaev–Song
  - quantum \( R \mapsto R^* \) algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10
- Campbell–Groves–Shepherd
  - fast pre-quantum algorithm for typical cyclotomic ring
to compute \( ug \mapsto \text{short } g \).
Many more NTRU variants (often not crediting NTRU).

Fully homomorphic encryption:
STOC 2009 Gentry
“Fully homomorphic encryption using ideal lattices”.
PKC 2010 Smart–Vercauteren.
Eurocrypt 2011 Gentry–Halevi.
etc.

Multilinear maps: e.g.,
Eurocrypt 2013 Garg–Gentry–Halevi
“Candidate multilinear maps from ideal lattices”.

STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.

First stage in attack:
SODA 2016 Biasse–Song
fast quantum algorithm to compute $gR \leftrightarrow ug$ with $u \in R^*$.

Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum $R \leftrightarrow R^*$ algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd
fast pre-quantum algorithm for typical cyclotomic ring to compute $ug \mapsto$ short $g$. 
STOC 2009 Gentry system is **broken** by quantum algorithms for typical “cyclotomic rings”.

First stage in attack:  
**SODA 2016 Biasse–Song**  
fast quantum algorithm to compute $gR \mapsto ug$ with $u \in R^*$.  
Builds upon **STOC 2014 Eisenträger–Hallgren–Kitaev–Song**  
quantum $R \mapsto R^*$ algorithm.  
Older pre-quantum algorithms take subexponential time.

Second stage of attack:  
**2014.10 Campbell–Groves–Shepherd**  
fast pre-quantum algorithm for typical cyclotomic ring to compute $ug \mapsto \text{short } g$.  

STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.

First stage in attack:
SODA 2016 Biasse–Song fast quantum algorithm to compute \( gR \mapsto ug \) with \( u \in R^* \).

Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum \( R \mapsto R^* \) algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd fast pre-quantum algorithm for typical cyclotomic ring to compute \( ug \mapsto \text{short } g \).

STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.

First stage in attack: SODA 2016 Biasse–Song fast quantum algorithm to compute \( gR \mapsto ug \text{ with } u \in R^\ast \).

Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum \( R \mapsto R^\ast \) algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd fast pre-quantum algorithm for typical cyclotomic ring to compute \( ug \mapsto \text{short } g \).


These attacks exploit structure of cyclotomic rings. Rescue system by switching to another ring?
The STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.

First stage in attack: STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum R ↦→ R∗ algorithm.

Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd fast pre-quantum algorithm for typical cyclotomic ring to compute \( ug \mapsto \text{short } g \).


These attacks exploit structure of cyclotomic rings. Rescue system by switching to another ring? 2014.02 Bernstein: pre-quantum attack strategy; subexponential time for many choices of ring.


2016 Bernstein–Chuengsatiansup–Lange–van Vredendaal “NTRU Prime”: use prime degree, large Galois group, inert modulus; reduce attack surface at low cost.
STOC 2009 Gentry system is broken by quantum algorithms for typical “cyclotomic rings”.

First stage in attack: SODA 2016 Biasse–Song fast quantum algorithm to compute $ug \mapsto ug$ with $u \in R^*$. Builds upon STOC 2014 Eisenträger–Hallgren–Kitaev–Song quantum $R \mapsto R^*$ algorithm. Older pre-quantum algorithms take subexponential time.

Second stage of attack: 2014.10 Campbell–Groves–Shepherd fast pre-quantum algorithm for typical cyclotomic ring to compute $ug \mapsto$ short $g$.


2014.02 Bernstein: pre-quantum attack strategy; subexponential time for many choices of ring.


2016 Bernstein–Chuengsatiansup–Lange–van Vredendaal “NTRU Prime”: use prime degree, large Galois group, inert modulus; reduce attack surface.
<table>
<thead>
<tr>
<th>Page</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Second stage of attack: 2014.10 Campbell–Groves–Shepherd fast pre-quantum algorithm for typical cyclotomic ring to compute $ug \mapsto \text{short } g$.</td>
</tr>
<tr>
<td></td>
<td>These attacks exploit structure of cyclotomic rings. Rescue system by switching to another ring?</td>
</tr>
<tr>
<td>14</td>
<td>2014.02 Bernstein: pre-quantum attack strategy; subexponential time for many choices of ring.</td>
</tr>
<tr>
<td></td>
<td>2016 Bernstein–Chuangsatiansup–Lange–van Vredendaal “NTRU Prime”: use prime degree, large Galois group, inert modulus; reduce attack surface at low cost.</td>
</tr>
</tbody>
</table>
Second stage of attack: 2014.10 Campbell–Groves–Shepherd
fast pre-quantum algorithm for typical cyclotomic ring to compute \( ug \mapsto \text{short } g \).


These attacks exploit structure of cyclotomic rings. Rescue system by switching to another ring?

2014.02 Bernstein: pre-quantum attack strategy; subexponential time for many choices of ring.


2016 Bernstein–Chuengsatiansup–Lange–van Vredendaal “NTRU Prime”: use prime degree, large Galois group, inert modulus; reduce attack surface at low cost.