A one-time single-bit fault leaks all previous NTRU-HRSS session keys to a chosen-ciphertext attack

D. J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum

---

cr.yp.to/papers.html#ntrw

Thanks to Lange for pointing out plaintext confirmation as a countermeasure to fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

A one-time single-bit fault
leaks all previous
NTRU-HRSS session keys
to a chosen-ciphertext attack

D. J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum

---

cr.yp.to/papers.html#ntrw

Thanks to Lange for pointing
out plaintext confirmation as a
countermeasure to fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses
`x25519+sntrup761` by default.

2022.07: NIST announces intent
to standardize Kyber (+ sigs).

A one-time single-bit fault leaks all previous NTRU-HRSS session keys to a chosen-ciphertext attack

D. J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum

---

cr.yp.to/papers.html#ntrw

Thanks to Lange for pointing out plaintext confirmation as a countermeasure to fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

A one-time single-bit fault leaks all previous NTRU-HRSS session keys to a chosen-ciphertext attack

D. J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum

---

cr.yp.to/papers.html#ntrw

Thanks to Lange for pointing out plaintext confirmation as a countermeasure to fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance . . . but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

me single-bit fault
previous
HRSS session keys
sen-ciphertext attack

ernstein

ty of Illinois at Chicago;
niversity Bochum

---

to/papers.html#ntrw

---

to Lange for pointing
ntext confirmation as a
measure to fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance ... but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2010–20
NTRU P
US9246
CN1081
US1105

bit fault

ion keys

rtext attack

is at Chicago;

ochum

---

s.html#ntrw

---

for pointing

rmation as a

o fault attacks.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance ... but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2010–2017 patents

NTRU Prime FAQ

US9246675, CN10

CN108173643, KR

US11050557, EP3

k

ago;

trw

g

s a

acks.

1

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance . . . but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2

2010–2017 patents listed in NTRU Prime FAQ: US90941 US9246675, CN107566121, CN108173643, KR10190568 US11050557, EP3698515.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance ... but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance ... but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.

No analysis of other patents.

PQ deployment and standards

2022.04: OpenSSH 9.0 uses `x25519+sntrup761` by default.

2022.07: NIST announces intent to standardize Kyber (+ sigs).

2022.11: Google announces that all internal Google networking uses `x25519+ntruhrss701`.

"Kyber has high performance ... but still lacks some clarification from NIST about its Intellectual Property status", i.e., patents.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

oyment and standards

: OpenSSH 9.0 uses
+sntrup761 by default.

: NIST announces intent
ardize Kyber (+ sigs).

: Google announces that
nal Google networking
5519+ntruhrss701.

has high performance ...
lacks some clarification
ST about its Intellectual
y status", i.e., patents.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU

2017 HR
proposal
has "a t
20 years

nd standards

H 9.0 uses
61 by default.

nounces intent
per (+ sigs).

announces that
e networking
uhrss701.

erformance . . .
e clarification
its Intellectual
i.e., patents.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU-HRSS se

2017 HRSS paper
proposal for OW-C
has "a track recor
20 years of cryptar

ds

ault.

ntent

s).

that

ng

.

ce . . .

tion

ctual

ts.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.

No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NT
proposal for OW-CPA encry
has "a track record of surviv
20 years of cryptanalysis".

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

2010–2017 patents listed in NTRU Prime FAQ: US9094189, US9246675, CN107566121, CN108173643, KR101905689, US11050557, EP3698515.

2022.11: NIST announces licenses for US9094189, US9246675 *for Kyber v2024 after Kyber v2024 is defined and standardized*.
No analysis of other patents.

For deploying software to protect users *now*, NTRU-HRSS is attractive: small, fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

017 patents listed in
Prime FAQ: US9094189,
675, CN107566121,
73643, KR101905689,
0557, EP3698515.

: NIST announces licenses
094189, US9246675 *for*
2024 after Kyber v2024 is
and standardized.
ysis of other patents.

oying software to protect
w, NTRU-HRSS is
ve: small, fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense
ciphertex
m, reenc

s listed in
: US9094189,
07566121,
R101905689,
698515.

nounces licenses
S9246675 *for*
Kyber v2024 is
ardized.
er patents.

ware to protect
-HRSS is
fast, unpatented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After d
ciphertext C to ob
m, reencrypt m an

189,

39,

censes
for
024 is

.

otect

tented.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain messa m, reencrypt m and reject if

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$.

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy!

Is NTRU-HRSS secure?

2017 HRSS paper says: NTRU proposal for OW-CPA encryption has "a track record of surviving 20 years of cryptanalysis".

Make various changes, including: "We now show how to turn the above OW-CPA secure encryption into an IND-CCA2-secure KEM"—i.e., include extra defenses to stop chosen-ciphertext attacks.

HRSS uses Fujisaki–Okamoto (FO) transform, specifically one of the variants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of "FrodoKEM parameter sets comfortably match their target security levels with a large margin".

U-HRSS secure?

RSS paper says: NTRU
for OW-CPA encryption
rack record of surviving
of cryptanalysis".

arious changes, including:
w show how to turn the
W-CPA secure encryption
IND-CCA2-secure KEM"—
ude extra defenses to
osen-ciphertext attacks.

ses Fujisaki–Okamoto
nsform, specifically one
ariants from 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of "FrodoKEM parameter sets comfortably match their target security levels with a large margin".

Defense
`ntrw`'s s
defenses

Instead
send cip
where H
Also use

ecure?

says: NTRU
CPA encryption
d of surviving
nalysis".

nges, including:
w to turn the
ecure encryption
2-secure KEM"—
defenses to
rtext attacks.

ki–Okamoto
pecifically one
m 2002 Dent.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of "FrodoKEM parameter sets comfortably match their target security levels with a large margin".

Defense 3 (in the
ntrw's survey of a
defenses): plaintex

Instead of cipherte
send ciphertext (E
where H' is a hash
Also use (E, H') in

RU
ption
ving

ding:
the
yption
EM”—
o
cks.

to
one
ent.

4

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of “FrodoKEM parameter sets comfortably match their target security levels with a large margin”.

5

Defense 3 (in the numbering
`ntrw`'s survey of attacks and
defenses): plaintext confirm

Instead of ciphertext $E(m)$,
send ciphertext $(E(m), H'(m$
where H' is a hash function.
Also use (E, H') in reencryp

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of "FrodoKEM parameter sets comfortably match their target security levels with a large margin".

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

Defense 1: After decrypting ciphertext C to obtain message m, reencrypt m and reject if $\neq C$.

This stops chosen-ciphertext attacks that probe variants of a legitimate C to see which variants decrypt to the same m.

If encryption is randomized, first derandomize it: obtain random bits as $H(m)$. Make sure m has high entropy! See recent collapse of "FrodoKEM parameter sets comfortably match their target security levels with a large margin".

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

1: After decrypting
xt C to obtain message
crypt m and reject if $\neq C$.

ps chosen-ciphertext
that probe variants of a
te C to see which variants
to the same m.

ption is randomized, first
mize it: obtain random
$H(m)$. Make sure m has
ropy! See recent collapse
loKEM parameter sets
ably match their target
levels with a large margin".

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

Current

2019 NT
adopts c
2017 Sa

Modifie
plaintex
relies on

decrypting
btain message
nd reject if $\neq C$.

-ciphertext
e variants of a
e which variants
ne m.

ndomized, first
btain random
ke sure m has
recent collapse
rameter sets
n their target
n a large margin".

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

Current NTRU-HR

2019 NTRU-HRSS
adopts changes pr
2017 Saito–Xagaw

Modified proposal
plaintext confirm
relies on another c

age
$f \neq C$.
t
of a
ariants
first
om
has
llapse
ets
get
nargin".

5

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

6

Current NTRU-HRSS is diff

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamaka

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 3 (in the numbering from `ntrw`'s survey of attacks and defenses): plaintext confirmation.

Instead of ciphertext $E(m)$, send ciphertext $(E(m), H'(m))$ where H' is a hash function. Also use (E, H') in reencryption.

This stops chosen-ciphertext attacks that exploit structure of the public-key encryption function E to convert $E(m)$ for secret m into, e.g., $E(m+1)$. Attacker has no way to convert $H'(m)$ into $H'(m+1)$ for "unstructured" H'.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

3 (in the numbering from
survey of attacks and
): plaintext confirmation.

of ciphertext $E(m)$,
hertext $(E(m), H'(m))$
H' is a hash function.
(E, H') in reencryption.

ps chosen-ciphertext
that exploit structure of
ic-key encryption function
nvert $E(m)$ for secret m
., $E(m+1)$. Attacker
way to convert $H'(m)$ into
1) for "unstructured" H'.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implic
adequat
confirma
chosen-c

numbering from
ttacks and
xt confirmation.

ext $E(m)$,
$E(m), H'(m))$
n function.
n reencryption.

-ciphertext
it structure of
ryption function
) for secret m
1). Attacker
vert $H'(m)$ into
structured" H'.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

ls implicit rejectio
adequate substitut
confirmation as a
chosen-ciphertext

g from
d
ation.

n)))

tion.

e of
nction
et m
ker
h) into
d" H'.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an
adequate substitute for plain
confirmation as a defense ag
chosen-ciphertext attacks?

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?

Issue 2: Proof is tight only in ROM; can this be exploited?

Current NTRU-HRSS is different

2019 NTRU-HRSS proposal adopts changes proposed by 2017 Saito–Xagawa–Yamakawa.

Modified proposal **removes plaintext confirmation** and relies on another defense.

Defense 4, implicit rejection (from 2017 Hofheinz–Hövelmanns–Kiltz, generalizing 2012 Persichetti): instead of having a KEM reject an invalid ciphertext C, have it output $H''(r, C)$ where r is a random string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

NTRU-HRSS is different

TRU-HRSS proposal
changes proposed by
ito–Xagawa–Yamakawa.

d proposal **removes**
t confirmation and
another defense.

4, implicit rejection (from
ofheinz–Hövelmanns–Kiltz,
zing 2012 Persichetti):
of having a KEM reject
d ciphertext C, have
t $H''(r, C)$ where r is a
string stored in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

2007 Ko
"Anyone
should t
dropping
had bee
security
someone
experien
never ha
if he hac
because

See also

RSS is different

S proposal
oposed by
wa–Yamakawa.

removes
nation and
defense.

t rejection (from
övelmanns–Kiltz,
Persichetti):
a KEM reject
ext C, have
where r is a
red in secret key.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

2007 Koblitz, rega
"Anyone working i
should think very
dropping a validat
had been put in to
security problems.
someone with Kra
experience and exp
never have made s
if he hadn't been
because of his 'pro

See also 2019 surv

erent

awa.

(from
–Kiltz,
i):
ject
e
s a
et key.

7

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

8

2007 Koblitz, regarding HM
"Anyone working in cryptog
should think very carefully b
dropping a validation step th
had been put in to prevent
security problems. Certainly
someone with Krawczyk's
experience and expertise wo
never have made such a blu
if he hadn't been over-confi
because of his 'proof' of sec

See also 2019 survey of failu

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography should think very carefully before dropping a validation step that had been put in to prevent security problems. Certainly someone with Krawczyk's experience and expertise would never have made such a blunder if he hadn't been over-confident because of his 'proof' of security."

See also 2019 survey of failures.

Is implicit rejection really an adequate substitute for plaintext confirmation as a defense against chosen-ciphertext attacks?

SXY+HRSS answer: Here's a **proof** of IND-CCA2 security from OW-CPA + implicit rejection.

Issue 1: Proof is only in QROM; are there non-QROM attacks?
Issue 2: Proof is tight only in ROM; can this be exploited?
Issue 3, my focus today: Are there chosen-ciphertext attacks beyond the IND-CCA2 model?

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography should think very carefully before dropping a validation step that had been put in to prevent security problems. Certainly someone with Krawczyk's experience and expertise would never have made such a blunder if he hadn't been over-confident because of his 'proof' of security."

See also 2019 survey of failures.

Should think very carefully before dropping plaintext confirmation.

it rejection really an
e substitute for plaintext
ation as a defense against
ciphertext attacks?

RSS answer: Here's a
f IND-CCA2 security from
A + implicit rejection.

Proof is only in QROM;
e non-QROM attacks?
Proof is tight only in
an this be exploited?
my focus today: Are
osen-ciphertext attacks
the IND-CCA2 model?

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography
should think very carefully before
dropping a validation step that
had been put in to prevent
security problems. Certainly
someone with Krawczyk's
experience and expertise would
never have made such a blunder
if he hadn't been over-confident
because of his 'proof' of security."

See also 2019 survey of failures.

Should think very carefully before
dropping plaintext confirmation.

2018 Be
implicit
random-
invalid c
the patt
plaintext
an earlie
current
any adva
defense
"seems
recomme
dual-def
given th
different

n really an
te for plaintext
defense against
attacks?

er: Here's a
A2 security from
cit rejection.

nly in QROM;
OM attacks?
ight only in
exploited?
today: Are
ertext attacks
CA2 model?

8

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography should think very carefully before dropping a validation step that had been put in to prevent security problems. Certainly someone with Krawczyk's experience and expertise would never have made such a blunder if he hadn't been over-confident because of his 'proof' of security."

See also 2019 survey of failures.

Should think very carefully before dropping plaintext confirmation.

9

2018 Bernstein–Pe
implicit rejection "
random-looking se
invalid ciphertexts
the pattern of vali
plaintext confirma
an earlier stage of
current proofs do
any advantages fo
defense constructi
"seems difficult to
recommendation a
dual-defense const
given that the defe
different aspects o

ntext
gainst

a
from
n.

OM;
ks?
in
?
e
cks
el?

8

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography should think very carefully before dropping a validation step that had been put in to prevent security problems. Certainly someone with Krawczyk's experience and expertise would never have made such a blunder if he hadn't been over-confident because of his 'proof' of security."

See also 2019 survey of failures.

Should think very carefully before dropping plaintext confirmation.

9

2018 Bernstein–Persichetti:
implicit rejection "produces
random-looking session keys
invalid ciphertexts, "so it hi
the pattern of valid cipherte
plaintext confirmation "stop
an earlier stage of the attac
current proofs do not "show
any advantages for the dual-
defense construction" **but** it
"seems difficult to justify a
recommendation against the
dual-defense construction"
given that the defenses "tar
different aspects of attacks"

2007 Koblitz, regarding HMQV:
"Anyone working in cryptography should think very carefully before dropping a validation step that had been put in to prevent security problems. Certainly someone with Krawczyk's experience and expertise would never have made such a blunder if he hadn't been over-confident because of his 'proof' of security."

See also 2019 survey of failures.

Should think very carefully before dropping plaintext confirmation.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

oblitz, regarding HMQV:
e working in cryptography
hink very carefully before
g a validation step that
n put in to prevent
problems. Certainly
e with Krawczyk's
ce and expertise would
ve made such a blunder
dn't been over-confident
of his 'proof' of security."

2019 survey of failures.

think very carefully before
g plaintext confirmation.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

An attac

DRAM
Often st
Google s
each sto
DRAM,
keys cor

rding HMQV:
in cryptography
carefully before
ion step that
o prevent
Certainly
wczyk's
pertise would
such a blunder
over-confident
oof' of security."

vey of failures.

carefully before
confirmation.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

An attack against

DRAM hardware i
Often stored bits a
Google statistics =
each storing a 256
DRAM, will have
keys corrupted eac

QV:
raphy
efore
nat

uld
nder
dent
urity."

ures.

before
tion.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

An attack against NTRU-HF

DRAM hardware is unreliabl
Often stored bits are corrupt
Google statistics $\Rightarrow$ 10^9 use
each storing a 256-bit key in
DRAM, will have 50000–140
keys corrupted each year.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow 10^9$ users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

2018 Bernstein–Persichetti:
implicit rejection "produces random-looking session keys" for invalid ciphertexts, "so it hides the pattern of valid ciphertexts"; plaintext confirmation "stops an earlier stage of the attack"; current proofs do not "show any advantages for the dual-defense construction" **but** it "seems difficult to justify a recommendation against the dual-defense construction" given that the defenses "target different aspects of attacks".

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow 10^9$ users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

rnstein–Persichetti:
rejection "produces
looking session keys" for
iphertexts, "so it hides
ern of valid ciphertexts";
t confirmation "stops
r stage of the attack";
proofs do not "show
antages for the dual-
construction" **but** it
difficult to justify a
endation against the
ense construction"
at the defenses "target
aspects of attacks".

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow 10^9$ users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What ca

Incompa
can re-a

ersichetti:

“produces
ession keys” for
, “so it hides
d ciphertexts”;
tion “stops
the attack”;
not “show
r the dual-
on” **but** it
justify a
gainst the
truction”
enses “target
f attacks”.

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow 10^9$ users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What can we do i

Incompatible new
can re-add plainte

” for
des
xts”;
s
k”;
’
-
t
e
get
.

10

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow 10^9$ users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

11

What can we do in response

Incompatible new NTRU-HR
can re-add plaintext confirm

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow$ 10^9 users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow$ 10^9 users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow$ 10^9 users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?

An attack against NTRU-HRSS

DRAM hardware is unreliable. Often stored bits are corrupted. Google statistics $\Rightarrow$ 10^9 users, each storing a 256-bit key in DRAM, will have 50000–140000 keys corrupted each year.

Main point of the `ntrw` paper: implicit rejection doesn't do its job if r is corrupted. Attacker detects invalid ciphertexts: changing r changes decryption output. See paper for application to NTRU-HRSS and full attack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?
Point fingers and do nothing?

ck against NTRU-HRSS

hardware is unreliable.
ored bits are corrupted.
statistics $\Rightarrow$ 10^9 users,
ring a 256-bit key in
will have 50000–140000
rupted each year.

int of the `ntrw` paper:
rejection doesn't do its job
rrupted. Attacker detects
iphertexts: changing r
decryption output. See
r application to NTRU-
nd full attack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?
Point fingers and do nothing?

Classic N

2022.10:
recomm
confirma
eliminat
U.S. pat

NTRU-HRSS

s unreliable.

are corrupted.

$\Rightarrow 10^9$ users,

5-bit key in

50000–140000

ch year.

`ntrw` paper:

doesn't do its job

Attacker detects

: changing r

n output. See

on to NTRU-

ack software.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?
Point fingers and do nothing?

Classic McEliece f

2022.10: Classic M

recommends dropp

confirmation "to p

eliminate any conc

U.S. patent 99124

RSS

e.
ted.
rs,
n
0000

er:
its job
etects
g *r*
See
RU-
re.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?
Point fingers and do nothing?

Classic McEliece followup

2022.10: Classic McEliece recommends dropping plaint
confirmation "to proactively
eliminate any concerns regar
U.S. patent 9912479".

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?

Use ECC in crypto libraries?

Use ECC in applications?

Programming language? OS?

Require SECDED ECC DRAM?

Point fingers and do nothing?

Classic McEliece followup

2022.10: Classic McEliece recommends dropping plaintext confirmation "to proactively eliminate any concerns regarding U.S. patent 9912479".

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?

Use ECC in crypto libraries?

Use ECC in applications?

Programming language? OS?

Require SECDED ECC DRAM?

Point fingers and do nothing?

Classic McEliece followup

2022.10: Classic McEliece recommends dropping plaintext confirmation "to proactively eliminate any concerns regarding U.S. patent 9912479".

Warns that this allows the `ntrw` attack whenever r is corrupted.

Describes ECC as a defense.

What can we do in response?

Incompatible new NTRU-HRSS can re-add plaintext confirmation.

Can fix corruption by applying an error-correcting code (ECC):

- `ntrw`'s `libsecded` software; or
- SECDED ECC DRAM hardware.

Many benefits beyond this attack.

Specify ECC in secret-key format?
Use ECC in crypto libraries?
Use ECC in applications?
Programming language? OS?
Require SECDED ECC DRAM?
Point fingers and do nothing?

Classic McEliece followup

2022.10: Classic McEliece recommends dropping plaintext confirmation "to proactively eliminate any concerns regarding U.S. patent 9912479".

Warns that this allows the `ntrw` attack whenever r is corrupted. Describes ECC as a defense.

Introduces principle of factoring "any generic transformation aiming at a goal beyond IND-CCA2" out of KEM specifications, to simplify design and review.