SecureComm
Security and Privacy in Communication Networks

Vulnerability Analysis of Mersenne Number-Based Post-Quantum Cryptosystems in Secure Communication Networks

Mengce Zheng, Abderrahmane Nitaj

ZWU; UNICAEN, CNRS, LMNO

Online, SecureComm 2026

July 2026

Outline

1.1 Motivation

Why this matters

- Communication infrastructures are migrating to post-quantum cryptography.
- Schemes modulo a Mersenne number $p = 2^n - 1$ are promoted as novel candidates.
- They are efficient; but security rests on the assumed hardness of specific number-theoretic problems.

Open question

Are such post-quantum crypto candidates *actually* secure?

1.2 The AJPS Cryptosystem at a Glance

Basic scheme

- Pick sparse binary f, g of Hamming weight $w \approx \sqrt{n}$.
- Public key $h \equiv f/g \pmod{p}$; private key (f, g).
- Encrypt bit $m \in \{0, 1\}$:

$$c = (-1)^m (ah + b).$$

- Decrypt: $d = \mathrm{HW}(cg)$; decide $m = 0$ if $d \leq 2w^2$.
- Condition: $n > 4w^2$.

KEM variant

- Pick random r and set (r, t) with $t \equiv rf + g \pmod{p}$.
- Public key (r, t) with above t; private key f.
- Encrypt message $m \in \{0, 1\}^w$:

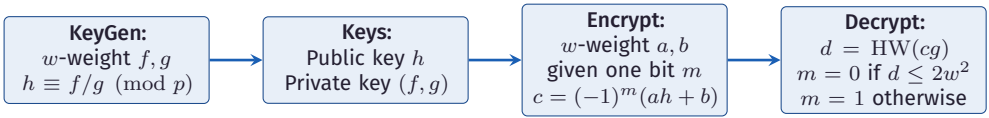
$$(c_1, c_2) = (ar + b_1,\ (at + b_2) \oplus \mathcal{E}(m)).$$

- Decrypt via the decoding algorithm $\mathcal{D}(c_1 f \oplus c_2)$.
- Stricter condition: $n > 10w^2$.

1.3 The AJPS Encrypt / Decrypt Flow

Sparse integer keys

- AJPS: sparse *integer* keys f, g (not NTRU polynomials).
- Basic: pk : $h = f/g$, needs $n > 4w^2$ + Hamming weight evaluation.
- KEM: pk : (r, t), needs $n > 10w^2$ + error-correcting code $(\mathcal{E}, \mathcal{D})$.



1.4 Problems & Threat Model

MLHRSP

Given Mersenne prime $p = 2^n - 1$, weight w: given $h \equiv f/g \pmod{p}$ for sparse f, g, recover (f, g).

MLHCSP

Given Mersenne prime $p = 2^n - 1$, weight w: given (r, t) with $t \equiv rf + g \pmod{p}$, recover (f, g).

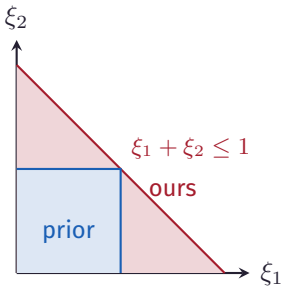
Threat model

- Adversary only *eavesdrops* on the channel.
- Intercepts public keys or ciphertexts.
- No active interaction; no chosen ciphertext.
- Attacks use only the public transcript.

1.5 Prior Attacks & Limitation

Prior attacks

- **Beunardeau et al.:** lattice attack $O(2^{2w})$.
- **de Boer et al.:** meet-in-the-middle (LSH).
- **Zheng–Yan:** solve bivariate equation, gain $\approx w^{3/2}$.
- **Coron–Gini:** KEM break $\approx 2^{-1.75w}$, needs all unknowns $< p^{2/3}$.



Common limitation

Tight explicit upper bounds on key sizes underestimate the weak-key space.

1.6 Our Contributions

- **Continued fraction attacks**: heuristic-free, via convergents of h/p or r/p.
- **Lattice-based attacks**: tailored lattice solving of modular equations; handles *unbalanced* parameters.
- **Theory + experiments**: precise success probabilities & time complexity; verified experiments for larger n and w.

Comparison with existing attacks (constraint column highlighted).

	Type 1: MLHRSP		Type 2: MLHCSP (bit)		Type 3: MLHCSP (KEM)	
	prior	ours	prior	ours	prior	ours
Prob.	2^{-2w}	$\sqrt{\pi}\, w^{3/2} 2^{-2w-1}$	2^{-2w}	$w^{3/2} 2^{-2w}$	$2^{-1.75w}$	$w^3 2^{-1.75w}$
Const.	$f, g < \sqrt{p}$	$fg < p$	$f, g < \sqrt{p}$	$fg < p$	$a, b_1, b_2 < p^{2/3}$	$ab_1b_2 < p^2$
Time	poly($\log p$)	poly($\log p$)	poly($\log p$)	poly($\log p$)	poly($\log p$)	poly($\log p$)

2.1 Continued Fraction Attack I: MLHRSP

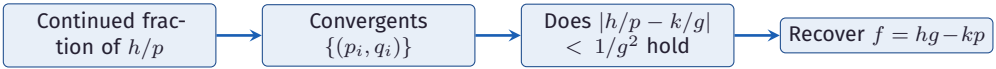
Key idea

Focus on $f = hg - kp$; search ratio k/g among convergents of h/p.

Theorem 2

If $fg < p$ (i.e. $\xi_1 + \xi_2 < 1$), recover (f, g) in poly(n) from $h \equiv f/g \pmod{p}$.

- If $fg < p$: $|h/p - k/g| < 1/g^2 \Rightarrow$ unique convergent.
- Then try $f = hg - kp$ and verify $\mathrm{HW}(f) = \mathrm{HW}(g) = w$.



2.2 Continued Fraction Attack II: MLHCSP-bit

Key idea

Focus on $t = rf + g - kp$; use ratio p_i/q_i among convergents of r/p.

Theorem 3

Given $r, t < p$ with $t \equiv rf + g \pmod{p}$, recover (f, g) in poly(n) if $fg < p$.

- Holds in low Hamming weight setting where f, g have small w.
- Reduces to locating (f, g) via convergents of r/p plus a short search.
- Recovers the secrets in polynomial time, *without* prior size estimates.

2.3 Attack Algorithm for MLHCSP-bit

Require: $p = 2^n - 1$, r, t, w
Ensure: f, g: $fg < p$, $t \equiv rf + g$, $\mathrm{HW}(f) = \mathrm{HW}(g) = w$
Convergents p_i/q_i of r/p for $i = 0, \ldots, l$
for $i = 1$ **to** l **do**
    $r_i \leftarrow (-1)^i(q_i r - p_i p)$
end for
for $i = 1$ **to** $l - 1$ **do**
    $\bar{a} \leftarrow \lfloor (-1)^i t q_{i+1}/p \rceil$, $\hat{a} \leftarrow \lceil q_{i+1}/q_i + q_{i+1} r_{i+1}/p \rceil$, $\bar{b} \leftarrow \lfloor (-1)^i t q_i/p \rceil$, $\hat{b} \leftarrow \lceil 2 - q_i r_{i+1}/p \rceil$
    for $\tilde{a} = -\hat{a}$ **to** $\hat{a}$ **do**
        $a \leftarrow \bar{a} + \tilde{a}$
        for $\tilde{b} = -\hat{b}$ **to** $\hat{b}$ **do**
            $b \leftarrow \bar{b} + \tilde{b}$, $f \leftarrow aq_i - bq_{i+1}$, $g \leftarrow t - (-1)^i(ar_i + br_{i+1}) \pmod p$
            if $\mathrm{HW}(f) = \mathrm{HW}(g) = w$ **and** $t \equiv rf + g$ **then**
                return (f, g)
            end if
        end for
    end for
end for

3.1 Lattice Attack I: MLHCSP-bit

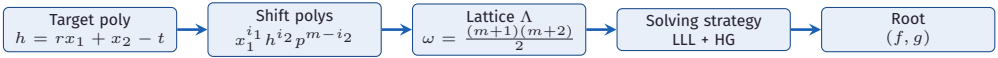
Polynomial model

Build a bivariate polynomial $h(x_1, x_2) = rx_1 + x_2 - t$ solved at (f, g).

Proposition 1

Given $r, t < p$ with $t \equiv rf + g \pmod{p}$, recover (f, g) in poly(n) if $fg < p$.

- Success probability $\Pr_2 \approx w^{3/2}\, 2^{-2w}$.



3.2 Lattice Attack II: AJPS KEM

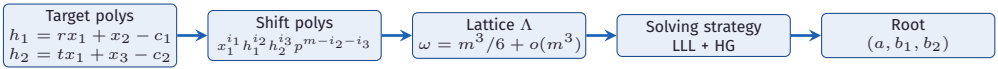
Polynomial model

Build $h_1 = rx_1 + x_2 - c_1$, $h_2 = tx_1 + x_3 - c_2$ solved at (a, b_1, b_2).

Proposition 2

Given c_1, c_2, r, t, recover (a, b_1, b_2) in poly(n) if $ab_1b_2 < p^2$.

- Success probability $\Pr_3 \approx w^3\, 2^{-1.75w}$.



4. Success Probability & Complexity

Gain over prior work

- MLHCSP-bit: $\Pr_2 \approx w^{3/2}\, 2^{-2w} \Rightarrow$ gain factor $r_2 \approx w^{3/2}$.
- KEM: $\Pr_3 \approx w^3\, 2^{-1.75w} \Rightarrow$ gain factor $r_3 \approx w^3$.

Takeaway

All attacks run in poly$(\log p)$ and a larger w does *not* make AJPS safer.

Remark

Both continued fraction and lattice-based methods use approximation of modular relations.

5.1 Implementation & Experimental Setup

Software

SageMath 10.3; full attack implementations in Python/Sage.

Hardware

2.80 GHz CPU, 16 GB RAM, Windows 10 / WSL 2 (notebook-class).

Parameters & trials

- $w \in [30, 70]$; modulus n up to 19937 bits.
- 5 independent trials per configuration.
- 100% success rate for each experiment.

Key observation

Runtime is driven mainly by modulus size n, confirming time in poly$(\log p)$.

5.2 Experiments: Continued Fraction Attacks

Selected results for continued fraction attacks.

Scheme	n	w	ξ_1	ξ_2	$\xi_1 + \xi_2$	Time (s)
MLHRSP	9689	45	0.350	0.650	1.000	3.12
	11213	50	0.100	0.900	1.000	8.52
	19937	70	0.300	0.700	1.000	17.29
MLHCSP (bit)	4253	30	0.897	0.102	0.999	23.86
	9689	45	0.235	0.765	1.000	25.15
	19937	70	0.500	0.500	1.000	310.20

Summary

- Recovery using only public values; no prior knowledge of ξ_i.
- Unbalanced keys cost a bit more, still run in polynomial time.

5.3 Experiments: Lattice-Based Attacks

Selected results for lattice-based attacks.

Scheme	n	w	$\sum \xi$	m	τ	Time (s)
MLHCSP (bit)	9689	45	1.000	3	10	0.34
	11213	50	1.000	3	10	1.61
	19937	70	1.000	3	10	3.79
MLHCSP (KEM)	4253	30	2.000	3	20	0.91
	11213	50	2.000	3	20	3.55
	19937	70	2.000	3	20	54.46

Summary

- Time grows with ω and n, yet efficient even for large n.
- Common root extracted $\Rightarrow$ KEM distinguishing confirmed.

6.1 Implications & Discussion

Speed

100% in seconds on standard hardware.

Scaling

Cost grows only with n (and parameter m).

Improvement

Relaxed conditions and weak key space.

Method nature & limitations

- Continued fraction: heuristic-free. Lattice: Coppersmith heuristic.
- Passive eavesdropping only; no active / side-channel adversary.

Practical implication

AJPS parameter sets are *insecure* under the passive model.

6.2 Conclusion & Future Work

Conclusion

- Several efficient parameter sets are insecure.
- Session key confidentiality might be easier to break.
- Attacks can run on standard hardware in reasonable time.

Future work

- Transform success probability into generic time-complexity estimates via the random partition technique.
- Before deploying Mersenne-number post-quantum crypto, *carefully design the parameter generation guidelines*.

6.3 Q&A

Thank you for your attention

Questions & Discussion?

Mengce Zheng

Online, SecureComm 2026, July 2026

mengce.zheng@gmail.com