Integer Reconstruction Public-Key Encryption

Houda Ferradi[2] and David Naccache[1]

[1] Département d'informatique, École normale supérieure, Paris, France
45 rue d'Ulm, 75230, Paris CEDEX 05, France
david.naccache@ens.fr
[2] NTT Secure Platform Laboratories
3–9–11 Midori-cho, Musashino-shi, Tokyo 180–8585, Japan
ferradi.houda@lab.ntt.co.jp

Abstract In [AJPS17], Aggarwal, Joux, Prakash & Santha described an elegant public-key cryptosystem (AJPS-1) mimicking NTRU over the integers. This algorithm relies on the properties of Mersenne primes instead of polynomial rings.

A later ePrint [BCGN17] by Beunardeau et al. revised AJPS-1's initial security estimates. While lower than initially thought, the best known attack on AJPS-1 still seems to leave the defender with an exponential advantage over the attacker [dBDJdW17]. However, this lower exponential advantage implies enlarging AJPS-1's parameters. This, plus the fact that AJPS-1 encodes only a single plaintext bit per ciphertext, made AJPS-1 impractical. In a recent update, Aggarwal et al. overcame this limitation by extending AJPS-1's bandwidth. This variant (AJPS-ECC) modifies the definition of the public-key and relies on error-correcting codes.

This paper presents a different high-bandwidth construction. By opposition to AJPS-ECC, we do not modify the public-key, avoid using error-correcting codes and use backtracking to decrypt. The new algorithm is *orthogonal* to AJPS-ECC as both mechanisms may be concurrently used *in the same ciphertext* and cumulate their bandwidth improvement effects. Alternatively, we can increase AJPS-ECC's information rate by a factor of 26 for the parameters recommended in [AJPS17].

The obtained bandwidth improvement and the fact that encryption and decryption are reasonably efficient, make our scheme an interesting post-quantum candidate.

1 Introduction

In a recent paper [AJPS17], Aggarwal, Joux, Prakash, & Santha described an elegant public-key cryptosystem (AJPS-1) mimicking NTRU over the integers. AJPS-1 relies on a specific arithmetic property of Mersenne numbers[3] instead of polynomial rings.

A later ePrint [BCGN17] by Beunardeau et al. revised AJPS-1's initial security estimates. While lower than initially thought, the best known attack

[3] Recall that a Mersenne prime is a prime of the form $2^n - 1$ for $n \in \mathbb{N}$.

against $\mathsf{AJPS\text{-}1}$'s complexity assumption[4] still seems to leave the defender with an exponential advantage over the attacker [dBDJdW17]. However, this lower exponential advantage implies enlarging $\mathsf{AJPS\text{-}1}$'s parameters. This, plus the fact that $\mathsf{AJPS\text{-}1}$ only encodes a single plaintext bit per ciphertext, made $\mathsf{AJPS\text{-}1}$ impractical. In a recent update, Aggarwal et al. overcame this limitation by extending $\mathsf{AJPS\text{-}1}$'s bandwidth. This variant ($\mathsf{AJPS\text{-}ECC}$) modifies the definition of the public-key and relies on error-correcting codes.

Our paper presents a different high-bandwidth construction. By opposition to $\mathsf{AJPS\text{-}ECC}$, we do not modify the public-key, avoid the use of error-correcting codes and use backtracking to decrypt. The new algorithm is *orthogonal* to $\mathsf{AJPS\text{-}ECC}$ as both mechanisms may be concurrently implemented *in the same ciphertext* and cumulate their bandwidth improvement effects[5]. Alternatively, we can increase $\mathsf{AJPS\text{-}ECC}$'s information rate by a factor of 26 for the parameters recommended in [AJPS17].

The obtained bandwidth improvement and the fact that encryption and decryption are reasonably efficient, make our scheme an interesting post-quantum candidate.

1.1 The Original AJPS-1 Cryptosystem

We denote by $\|x\|$ the Hamming weight of x and let $\mathfrak{H}_{n,h}$ be the set of all n-bit strings of Hamming weight h.

The original $\mathsf{AJPS\text{-}1}$ scheme is defined by the following sub-algorithms:

- **Setup**$(1^\lambda) \rightarrow \mathsf{pp}$. Chooses the public parameters $\mathsf{pp} = \{n, h\}$ so that $p = 2^n - 1$ is an n-bit Mersenne prime achieving some λ-bit security level.
- **KeyGen**$(\mathsf{pp}) \rightarrow \{\mathsf{sk}, \mathsf{pk}\}$. Picks $\{F, G\} \in_R \mathfrak{H}^2_{n,h}$ and returns:

$$\begin{cases} \mathsf{sk} & \leftarrow G \\ \mathsf{pk} & \leftarrow H = F/G \bmod p \end{cases}$$

- **Enc**$(\mathsf{pp}, \mathsf{pk}, m \in \{0,1\}) \rightarrow C$. Picks $\{A, B\} \in_R \mathfrak{H}^2_{n,h}$, and computes:

$$C \leftarrow (-1)^m (AH + B) \bmod p$$

- **Dec**$(\mathsf{pp}, \mathsf{sk}, C) \rightarrow \{\perp, 0, 1\}$, computes $d = \|GC \bmod p\|$ and returns:

$$\begin{cases} 0 & \text{if } d \leq 2h^2, \\ 1 & \text{if } d \geq n - 2h^2, \\ \perp & \text{otherwise} \end{cases}$$

[4] The Mersenne Low Hamming Ratio Assumption

[5] This requires that the $\{h, n\}$ parameters of both schemes coincide. We conjecture that such a meeting point exists.

The intuition behind the decryption formula is the observation that when $m = 0$ we get:

$$W = GC = G(AH + B) = FA + GB \Rightarrow W \text{ is of low Hamming weight}$$

We refer the reader to [AJPS17] for more details about this cryptosystem.

To increase bandwidth, Aggarwal et al. introduced the AJPS-ECC variant described hereafter.

1.2 The AJPS-ECC Cryptosystem

AJPS-ECC requires an ancillary error correction scheme $\{\mathcal{D}, \mathcal{E}\}$.

AJPS-ECC is formally defined by the following sub-algorithms:

– **Setup**$(1^\lambda) \to \mathsf{pp}$. As in AJPS-1.
– **KeyGen**$(\mathsf{pp}) \to \{\mathsf{sk}, \mathsf{pk}\}$. Picks $\{F, G\} \in_R \mathfrak{H}^2_{n,h}$, $R \in_R \{0,1\}^n$ and returns:

$$\begin{cases} \mathsf{sk} & \leftarrow F \\ \mathsf{pk} & \leftarrow \{R, T\} = \{R, F \times R + G \bmod p\} \end{cases}$$

– **Enc**$(\mathsf{pp}, \mathsf{pk}, m \in \{0,1\}^\lambda) \to C$. Picks $\{A, B_1, B_2\} \in_R \mathfrak{H}^3_{n,h}$ and computes the ciphertext:

$$C = \begin{cases} C_1 & \leftarrow \ \ A \times R + B_1 \bmod p \\ C_2 & \leftarrow (A \times T + B_2 \bmod p) \oplus \mathcal{E}(m) \end{cases}$$

– **Dec**$(\mathsf{pp}, \mathsf{sk}, C) \to \{\perp, m\}$ returns:

$$\mathcal{D}((F \times C_1 \bmod p) \oplus C_2).$$

For the sake of clarity, we keep the definition of C_1 unchanged but slightly depart from [AJPS17]'s original formulae by modifying the definitions of T and C_2 as follows:

$$\begin{aligned} T &\leftarrow \ \ F \times R - G \ \bmod p \\ C_2 &\leftarrow (A \times T - B_2 \bmod p) \oplus \mathcal{E}(m) \end{aligned}$$

To understand the intuition behind **Dec** consider the quantity $W = FC_1 - C_2$ corresponding to the particular case $\mathcal{E}(m) = 0$:

$$W = FAR + FB_1 - AT + B_2 = FAR + FB_1 - A(FR - G) + B_2 = FB_1 + GA + B_2$$

As before, we see that $d = \|W\|$ is low. This means that the noise attached to $\mathcal{E}(m)$ after the clean-off operation $(F \times C_1 \bmod p) \oplus C_2$ is low and thus surmountable by the error-correcting code $\{\mathcal{E}, \mathcal{D}\}$.

The reader is referred, again, to [AJPS17] for more details about this cryptosystem and the parameter choices allowing successful decryption and sufficient security. Sticking only to the core idea, we purposely omit the hashing and re-encryption tests performed during the key de-encapsulation process.

2 The New Idea: Randomness Reconstruction

Our idea departs from AJPS-1 in a direction orthogonal to the above.

We set by design $m = 0$ in AJPS-1 or $\mathcal{E}(m) = 0$ in AJPS-ECC and attempt to recover *the randomness*[6] into which information (encapsulated keys and/or plaintext information) will be embedded.

The intuition is that the receiver might be able to recover the randomness if parameters are properly chosen *using his extra knowledge* of G, F and knowing that, in addition, the unknown randomness has a low Hamming weight.

We hence focus the rest of this paper on methods for solving the equations:

$$W = Fx + Gy \quad \text{or} \quad W = Fx + Gy + z \bmod p$$

Where all parameters[7] and unknowns are *randomly* chosen in $\mathfrak{H}_{n,h}$ and where a solution $\{x, y\}$ or $\{x, y, z\}$ *is known to exist*.

We do not introduce any modifications in **Setup** and **KeyGen**, nor do we modify pp or sp[8]. We thus focus on the encapsulation (encryption) and on the de-encapsulation (decryption) processes only.

In a non-KEM version, a plaintext m encoded in the unknowns (x, y or x, y, z) can be directly recovered upon decryption. Such an encryption mode must however be protected against active attacks using padding and randomization that we do not address here.

Note 1: It is tempting but inadvisable to create dependencies between the variables F, G and/or the unknowns x, y, z. Consider an AJPS-ECC where $m \in_R \{0,1\}^\lambda$ and $\{A, B_2\} \in_R \mathfrak{H}^2_{n,h}$ but where $B_1 \leftarrow \mathcal{H}(m)$ is obtained by hashing m into $\mathfrak{H}_{n,h}$. Given m, *anybody* can re-compute B_1 and algebraically infer A, B_2. We hence see in this example that A, B_2 do not add extra entropy as security solely rests upon m.

Note 2: We carefully distinguish between *bandwidth* and *information rate*. An idea, unexplored in AJPS-ECC, may exploit Note 1 to transport more plaintext information in $\{C_1, C_2\}$ *without adding extra security*. To encrypt a τ-bit message μ, pick a key $m \in_R \{0,1\}^\lambda$ and encrypt $c \leftarrow \mathcal{F}_m(\mu)$ using a block cipher $\mathcal{F}$. Set $B_1 \leftarrow \mathcal{H}(m)$. Let $\mathcal{M}$ be any invertible public mapping $\mathcal{M} : \{0,1\}^\tau \to \mathfrak{H}^2_{n,h}$. Encode: $\{A, B_2\} \leftarrow \mathcal{M}(c)$ and form $\{C_1, C_2\}$ using AJPS-ECC. To decrypt, recover m using error-correction, recompute B_1, algebraically recover $\{A, B_2\}$ and retrieve the plaintext μ by:

---

[6] A, B or A, B_1, B_2

[7] Except W

[8] Note that in AJPS-1/ECC given G one can compute F and *vice versa*.

$$\begin{aligned}\mu &\leftarrow \mathcal{F}_m^{-1}(\mathcal{M}^{-1}(A, B_2))\\ &= \mathcal{F}_m^{-1}\left(\mathcal{M}^{-1}(\frac{C_1 - B_1}{R} \bmod p, C_2 - \frac{(C_1 - B_1)T}{R} \bmod p)\right)\\ &= \mathcal{F}_m^{-1}\left(\mathcal{M}^{-1}(\frac{C_1 - \mathcal{H}(m)}{R} \bmod p, C_2 - \frac{(C_1 - \mathcal{H}(m))T}{R} \bmod p)\right)\end{aligned}$$

Because in AJPS-ECC $\{n, h\} = \{756839, 256\}$ the potential encoding capacity of $\mathcal{M}$ can be relatively high:

$$2 \log_2 \binom{756839}{256} = 6631 \text{ bits}$$

This increases AJPS-ECC's information rate by a factor of 26. Again, proper message padding may be necessary to resist active attacks (analysis underway).

2.1 The Bivariate Cryptosystem

- **Setup**$(1^\lambda) \rightarrow \mathsf{pp}$ and **KeyGen**$(\mathsf{pp}) \rightarrow \{\mathsf{sk}, \mathsf{pk}\}$ are identical to AJPS-1.
- **Enc**$(\mathsf{pp}, \mathsf{pk}) \rightarrow C$. Picks $\{A, B\} \in_R \mathfrak{H}_{n,h}^2$ and computes:

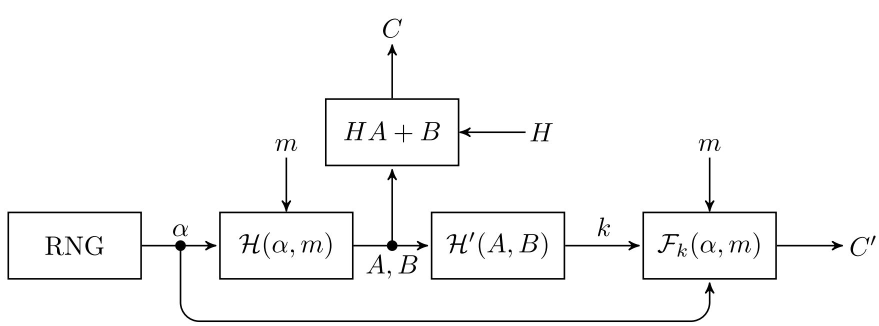
  $$C \leftarrow AH + B \bmod p$$

- **Dec**$(\mathsf{pp}, \mathsf{sk}, C) \rightarrow \{\perp, \{A, B\}\}$ returns:

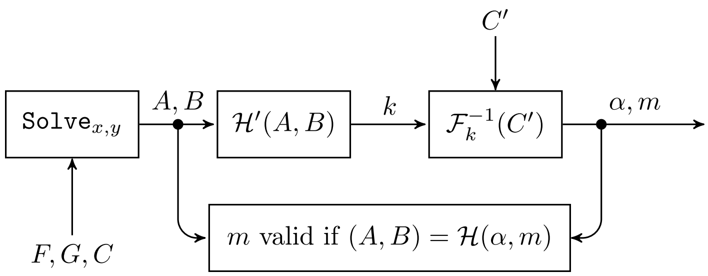
  $$\{A, B\} \leftarrow \mathtt{Solve}_{x,y}[GC = Fx + Gy \bmod p]$$

If $\{A, B\} \neq \perp$ use $\{A, B\}$ as KEM entropy for further encryption.

Let μ be a plaintext and $\mathcal{R}$ a redundancy function. Compute $m \leftarrow \mathcal{R}(\mu, \rho)$ where ρ is random. A typical KEM[9] is shown here:



[9] Where $\mathcal{H}, \mathcal{H}'$s are hash functions and $\mathcal{F}$ is a block-cipher.



Retrieve m from $\mu \leftarrow \mathcal{R}^{-1}(m)$.

2.2 The Trivariate Cryptosystem

- **Setup**$(1^\lambda) \to \mathsf{pp}$ and **KeyGen**$(\mathsf{pp}) \to \{\mathsf{sk}, \mathsf{pk}\}$ are identical to AJPS-ECC but with the modified formula $T \leftarrow F \times R - G \bmod p$.
- **Enc**$(\mathsf{pp}, \mathsf{pk}) \to C$. Picks $\{A, B_1, B_2\} \in_R \mathfrak{H}^3_{n,h}$ and computes the ciphertext:

$$C = \begin{cases} C_1 & \leftarrow A \times R + B_1 \bmod p \\ C_2 & \leftarrow A \times T - B_2 \bmod p \end{cases}$$

- **Dec**$(\mathsf{pp}, \mathsf{sk}, C) \to \{\perp, \{A, B_1, B_2\}\}$ returns:

$$\{A, B_1, B_2\} \leftarrow \mathtt{Solve}_{x,y,z}[FC_1 - C_2 = Fy + Gx + z \bmod p]$$

If $\{A, B_1, B_2\} \neq \perp$ use $\{A, B_1, B_2\}$ as KEM entropy for further encryption.

Note 3: As noted before, the trivariate version may accommodate in the encryption formula an *independent* $\mathcal{E}(m)$ and thus cumulate the bandwidth improvements due to both mechanisms. This requires that the $\{h, n\}$ values of both schemes coincide and the enforcement of the condition $n \leq 16h^2$, not addressed here. We conjecture that such a meeting point exists.

The following sections explain how to instantiate $\mathtt{Solve}_{x,y}$. The routine $\mathtt{Solve}_{x,y,z}$ is obtained *mutatis mutandis*.

3 Instantiating $\mathtt{Solve}_{x,y}$ Using Backtracking

The intuition behind $\mathtt{Solve}_{x,y}$ is the following: assume that we are given the quantity $W = GC = AF + BG \bmod p$ where $\|W\| \cong 2h^2$. Because multiplication modulo p is (somewhat) weight-preserving, we can test the hypothesis that the i-th bit of A is equal to one by looking at the quantity Δ:

$$\Delta = \|W\| - \|W - 2^i F \bmod p\|$$

Intuitively, a good guess should result in a weight decrease of $\simeq h$ whereas a wrong guess should re-blur W by triggering random carry propagations. Evidently, because there may be false positives during this process, we must be able to backtrack. To reduce the false positive error probability, n must be large enough with respect to h. The exact same idea applies to $\texttt{Solve}_{x,y,z}$.

3.1 Prerequisites & Subroutines

We start by introducing three necessary prerequisites.

The ancillary function Confirm: Our algorithms require an ancillary function Confirm-ing a candidate solution $\{x, y\}$. e.g. given a candidate x, $\texttt{Confirm}(x)$ may solve $GC = Fx + Gy \bmod p$ for y and return $\{y, \mathsf{True}\}$ if $\|x\| = \|y\| = h$. Because in some cases several solutions may exist, a simpler implementation may just compare $\mathcal{H}(x, y)$ to a confirmation digest τ provided with the ciphertext and return $\{y, \mathsf{True}\}$ if the purported solution hashes into τ. If $\mathcal{H}(x, y) \neq \tau$ then $\texttt{Confirm}(x)$ returns $\{\perp, \mathsf{False}\}$.

Dealing with decoding failures: Because we may discard seemingly uninteresting (but actually promising) exploration paths, backtracking may fail to decode W. As it seems complex to formally compute the algorithm's success probability, we estimated it by simulation. To deal with decryption failures we re-attempt backtracking after index randomization *i.e.* pick t random permutations $\{\phi_0, \ldots, \phi_{t-1}\}$ of $\mathbb{Z}_k$ and re-run $\mathcal{B}_2(W, \emptyset, 0, \phi_j)$ t times hoping that at least one of the t runs will succeed[10]. A more brutal approach consists in sending t encapsulated keys to increase the probability exponentially. This exponential probability gain only handicaps the information rate by a constant factor[11].

Determining the backtracking aperture Γ: Backtracking is parametrized by a constant Γ controlling the aperture of the exhausting process (*i.e.* the marginal tolerance allowing to exclude a search path from further investigation). Simulations indicate that for any given $\{n, h\}$ there is a Γ_{optimal} value minimizing the failure probability. We did not attempt a formal analysis of the dependency between $\{n, h\}$ and Γ_{optimal} but estimated Γ_{optimal} for various $\{n, h\}$ pairs using simulations. For instance for $n = 19937, h = 72$, we get[12] the following decryption probabilities for $t = 1$:

Γ	50	51	52	53	54	55	56	57	58	59	60
Probability	24%	20%	26%	30%	32%	20%	22%	18%	14%	9%	4%

[10] Note that $\mathcal{B}_1(W, \emptyset, 0) = \mathcal{B}_2(W, \emptyset, 0, \text{ID})$.

[11] Link $B_0, \ldots, B_{t-1}$ in a way allowing the recovery of all the B_i if one of them is known (*e.g.* define $B_i = \mathcal{F}_i(\text{seed})$ where $\mathcal{F}_k(m)$ is a block-cipher encrypting into $\mathfrak{H}_{n,h}$). Use the A_i to transport entropy or information. One successful decryption reveals the seed $\Rightarrow$ open all the B_is $\Rightarrow$ all the t information containers A_i.

[12] 50 decryption simulations per entry.

3.2 The Backtracking Algorithms

The deterministic backtracking algorithm $\mathcal{B}_1$ subtracts left-shifted Fs from W to obtain candidate ws having smaller and smaller weights. $\mathcal{B}_1$ maintains a set of integers R containing the bit positions of x discovered so far. The deterministic algorithm is called by $\{A, B\} \leftarrow \mathcal{B}_1(W, \emptyset, 0)$ and the randomized version is called by $\mathcal{B}_2(W, \emptyset, 0, \phi_j)$ where ϕ_js are random permutations of $\mathbb{Z}_k$. Code is available from the authors upon request.

ALGORITHM 1
Backtracking $\mathcal{B}_1(w, R, e)$

Input: w, R, e. The values $G, F, h, n, p = 2^n - 1, C$ are global and invariant.
Output: $\{x, y\} = \{A, B\}$ such that $W = CG = Fx + Gy \bmod p$ or Failure.

```
if e = n then return Failure
else
    if #R = h then
        x ← ∑_{i∈R} 2^i
        {s, y} ← Confirm(x)
        if s then return {x, y}
w̄ ← w − 2^e × F mod p
if |‖w‖ − ‖w̄‖| ≤ h − Γ then
    B_1(w̄, R ∪ {e}, e + 1)
else
    B_1(w, R, e + 1)
```

ALGORITHM 2
Backtracking $\mathcal{B}_2(w, R, e, \phi)$

Input: w, R, e, ϕ. The values $G, F, h, n, p = 2^n - 1, C$ are global and invariant.
Output: $\{x, y\} = \{A, B\}$ such that $W = CG = Fx + Gy \bmod p$ or Failure.

```
if e = n then return Failure
else
    if #R = h then
        x ← ∑_{i∈R} 2^{φ(i)}
        {s, y} ← Confirm(x)
        if s then return {x, y}
w̄ ← w − 2^{φ(e)} × F mod p
if |‖w‖ − ‖w̄‖| ≤ h − Γ then
    B_2(w̄, R ∪ {e}, e + 1, φ)
else
    B_2(w, R, e + 1, φ)
```

Note 4: We conjecture that working with a fixed Γ during the entire backtracking process handicaps the algorithm. When the process starts the weight of W is high, hence the probability to strike-out h bits by subtraction is high. However as subtractions make w sparser aperture should intuitively decrease. It may hence make sense to explore algorithms in which the constant Γ_{optimal} is replaced by a function $\Gamma(\|w\|)$.

Best candidate search: $\mathcal{B}_1$ and $\mathcal{B}_2$ explore all the paths starting by an *a priori* promising Δ. However, $\mathcal{B}_1$ and $\mathcal{B}_2$ do not explore the most promising paths first. A more complex backtracking strategy ($\mathcal{B}_3$) trying with priority the paths starting by a Δ as close as possible to h was developed as well (information available from the authors upon request). We do not include this algorithm here for the sake of concision.

3.3 Eccentric Reconstruction Strategies

Backtracking might be improved in a variety of ways. As examples, we list here a number of research ideas that we did not explore in detail.

Brittle encryption formulae: We may modify the bivariate encryption formula to $C \leftarrow HA + 3B$ and enforce by design that H, A, B do not contain the binary sequence `11`. This means that the bit positions representing $3B$ will be "colored" by a pattern `11` making their isolation and identification easier. If $n \gg h$ we may even attempt to brutally reset all the isolated ones in W and divide the result by $3G$ to directly obtain B. For the trivariate version one may use:

$$C = \begin{cases} C_1 & \leftarrow AR + B_1 \mod p \\ C_2 & \leftarrow AT - 3B_2 \mod p \end{cases}$$

resulting in the decryption formula $W = FC_1 - C_2 = FB_1 + GA + 3B_2$. Here as well, we banish the pattern `11` from G, F, A, B_1, B_2. We may thus attempt to identify in W the binary patterns `11`, hinting the probable presence of B_2 to ease decoding. Note that the pattern `11` may result naturally from the multiplication, the addition or the reduction and hence mislead the decoder (backtrack). Similarly, an `11` due to $3B_2$ may disappear due to addition (backtrack).

Dye tracing: In hydrogeology, *dye tracing* is a technique for tracking various flows using dye added to the water source. In other words, dye tracing uses dye as a flow tracer. It is an evolution of the ages-known float tracing method, which consists of throwing a buoyant object into a waterflow to see where it emerges. To simulate the effect of dye tracing, we inject into F's digits a few low-weight binary patterns and track their appearance in W. For instance (toy example), generate an F of weight $h - 10$ not containing any of the ten sequences ℓ_i:

11	101	111	1001	1011	1101	10001	11001	10101	10011

randomly insert those ten ℓ_is into F's blank spaces (insert each ℓ_i once, this will increase the weight of F to $h - 10 + \sum \|\ell_i\| = h - 10 + 26 = h + 16$ and the weight of W to $\simeq 2h^2 + 16h$). To retrieve A, isolate the 10 dyes tracers in W and use majority voting on bit offsets to infer the probable positions of A's bits.

Demodulation: We can attempt to "travel back in time" and infer $\omega = FA + GB \in \mathbb{Z}$ from W, or at least estimate the probability that a candidate bit in W originates from the number's pre-reduced upper half. Given $\omega \in \mathbb{Z}$ decryption[13] is immediate because:

$$A = \omega F^{-1} \bmod G = (W \text{ demod } p) \times F^{-1} \bmod G$$

To demodulate W we work modulo $p = 2^n - 3$ that "colors" the folded MSBs by turning them into LSB `11`s. The process is error-prone[14] but *actually works* for parameters that are large enough. We implemented the idea very brutally, by simply translating each `11` in W into a `1` in the MSB of ω without taking any further precautions. 100 demodulation attempts for $\{k = 6 \times 10^7, h = 55\}$ resulted in 29 successes. Although k is huge, the resulting information rate is not "that" catastrophic as we can pack:

$$2 \log_2 \binom{6 \times 10^7}{55} = 2356 \text{ plaintext bits into the ciphertext.}$$

In other words, each plaintext bit claims 25461 ciphertext bits and is successfully transmitted with probability 29%.

While $h = 55$ is not very large and $k = 6 \times 10^7$ is extremely large, our simulation shows that it is definitely possible to make ingredients meet at a workable parameter combination. We conjecture that with proper analysis and refined demodulation strategies k might be reduced by at least two orders of magnitude. It may also be possible to work modulo $2^k - \pi$ with a more distinguishable color $\pi \neq 3$ despite an extra weight due to a more complex π. $\pi = -1$ is interesting as well as -1 turns folded bits into long chains of `1`s.

Note 5 (important): One of the features preventing lattice-based attacks in AJPS-1/ECC is the emergence of parasitic short vectors due to working modulo $2^n - 1$ (section 5.1. [AJPS17] [15]). We did not evaluate the impact of $\pi \neq 1$ on the number and the norm of parasitic short vectors *and hence on security*.

[13] Take F, G coprime in **Setup**.

[14] Again, "natural" `11`s may be already present in the LSBs of ω, `11` + `01` may destroy an `11`, `10` + `01` may create fake `11`s etc.

[15] ePrint version 20170530:072202.

Caveat

Brittle encryption, dye tracing and demodulation are research directions that we do not claim to be secure.

4 Security & Parameter Sizes

This work did not cover the security of the proposed constructions and focused on the textbook modes in which data is encoded and decoded. Parameter sizes were not recommended and numerical examples are given for illustrative purposes.

A careful balance must be established between ① the security, ② the decodability (backtracking failure probability) and ③ the efficiency of the various `Solve` processes. So far, simulations indicate that there are ways to practically satisfy those three constraints at once.

5 Acknowledgments

The authors thank Waïss Azizian, Sarah Houdaigoui and Quốc Tún Lê for the development and the simulation of different backtracking strategies.

References

AJPS17. Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Miklos Santha. A new public-key cryptosystem via Mersenne numbers. Cryptology ePrint Archive, Report 2017/481, 2017. http://eprint.iacr.org/2017/481.

BCGN17. Marc Beunardeau, Aisling Connolly, Rémi Géraud, and David Naccache. On the hardness of the mersenne low hamming ratio assumption. Cryptology ePrint Archive, Report 2017/522, 2017. http://eprint.iacr.org/2017/522.

dBDJdW17. Koen de Boer, Léo Ducas, Stacey Jeffery, and Ronald de Wolf. Attacks on the ajps mersenne-based cryptosystem. Cryptology ePrint Archive, Report 2017/1171, 2017. https://eprint.iacr.org/2017/1171.