A Digital Signature Scheme Based on Random Split of St-Gen Codes

Danilo Gligoroski[1] and Simona Samardjiska[2]

[1]Department of Telematics, The Norwegian University of Science and Technology (NTNU), Trondheim, Norway,
[2]Faculty of Computer Science and Engineering, UKIM, Skopje, Macedonia
`danilog@item.ntnu.no, simona.samardjiska@finki.ukim.mk`

Abstract. Recently we proposed a method for a random split of Staircase-Generator codes (St-Gen codes) to counter the weaknesses found in the previous constructions of public key schemes using St-Gen codes. The initial proposal for the random split addressed only the encryption scheme, and we left the problem of how to apply the random splitting on the signature scheme open. In this work we solve that open problem and describe a digital signature scheme based on random split of St-Gen codes.

Keywords: Public Key Cryptography, Code Based Signatures, St-Gen Codes, List Decoding

1 Introduction

One of the schemes believed to be quantum secure is the McEliece public key scheme [4], published in 1978. Its security is based on the NP-hardness of the problem of decoding random linear codes.

Recently, an encryption and signature variant of the McEliece scheme based on Staircase-Generator codes was introduced in [2,3]. For the public keys produced with these codes a distinguisher was proposed by Sendrier and Tillich [8], and recently a very similar distinguishing strategy was presented as a full and practical key recovery attack by Moody and Perlner [5]. In [5] there is presented also a practical signature forgery attack based on an ISD attack.

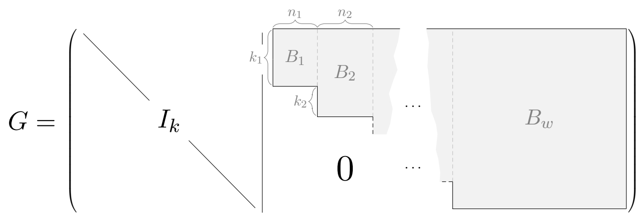
In order to thwart the distinguishing and ISD attacks of [8,5] against the encryption scheme defined in [2,3] we introduced the concept of *"Valid Error Split"* in [7]. By using that concept we showed how to transform the St-Gen encryption scheme from [2,3] into a randomly split St-Gen scheme. In our security analysis we showed that the probability of the attacker obtaining conditions under which the attacks [8,5] can be mounted, becomes negligible.

The problem of transforming the St-Gen signature scheme into a randomly split St-Gen signature scheme remained open. In this paper we solve that problem. We provide an initial security analysis of the scheme as well as some concrete parameters and instances.

2 Definition of Staircase-Generator Codes and Random Split of Their Generator Matrix

We repeat in brief some of the notions given in [2,3,6,7]. We denote by $\mathcal{C} \subseteq \mathbb{F}_2^n$ a binary (n, k) code of length n and dimension k. We denote the generator matrix of the code by G, and $wt(\mathbf{x})$ denotes the Hamming weight of the word $\mathbf{x}$.

Definition 1. *Let $k_i, n_i \in \mathbb{N}$, and let $k = k_1 + k_2 + \cdots + k_w$ and $n = k + n_1 + n_2 + \cdots + n_w$. Further, let B_i be a random binary matrix of dimension $\sum_{j=1}^{i} k_j \times n_i$. A linear binary (n, k) code $\mathcal{C}$ with the following generator matrix in standard form:*



(1)

is called Staircase-Generator code (St-Gen code).

Definition 2. *Let ℓ be a positive integer and let $p_d \in \mathbb{F}_2[x_1, x_2, \ldots, x_\ell]$ be a multivariate polynomial of degree $\geqslant 2$. We say that E_ℓ is an* error set *if it is the solution set of p_d, i.e. $E_\ell = \{\mathbf{e} \in \mathbb{F}_2^\ell \mid p_d(\mathbf{e}) = 0\}$. We will refer to p_d as the defining polynomial.*

We define the density *of the error set E_ℓ to be $D(E_\ell) = |E_\ell|^{1/\ell}$. We will refer to the integer $\ell > 0$ as the* granulation *of E_ℓ. In [2] it was proven that if two error sets $E_{\ell_1} \subseteq \mathbb{F}_2^{\ell_1}$, $E_{\ell_2} \subseteq \mathbb{F}_2^{\ell_2}$, have the same density ρ, then $D(E_{\ell_1} \times E_{\ell_2}) = \rho$.*

The decoding of St-Gen codes relies on the technique of *list decoding*. The following Proposition from [2] determines the parameters of a St-Gen code that provide an efficient decoding.

Proposition 1 ([2]). *Let $\mathcal{C}$ be any binary (n, k) code and $E \subset \mathbb{F}_2^n$ be an error set of density ρ. Let $\mathbf{w}$ be any word of length n, $W_E = \{\mathbf{w} + \mathbf{e} \mid \mathbf{e} \in E\}$ and let $\mathcal{C}_{W_E}$ denote the set of codewords in W_E. Suppose there exists a codeword $\mathbf{c} \in W_E$. Then the expected number of codewords in $W_E \setminus \{\mathbf{c}\}$ is approximately $\rho^n 2^{k-n}$ for large enough n and k.*

Let E_ℓ be an error set with density ρ where ℓ divides n and $m = n/\ell$. We recall Alg. 1 from [2], that is an efficient algorithm for decoding a code $\mathcal{C}$, that corrects errors from the set E_ℓ^m.

Algorithm 1 Decoding

Input: Vector $\mathbf{y} \in \mathbb{F}_2^n$, and generator matrix G of the form (1).

Output: A list $L_w \subset \mathbb{F}_2^k$ of valid decodings of $\mathbf{y}$.

Procedure:

Let $K_i = k_1 + \cdots + k_i$. Represent $\mathbf{x} \in \mathbb{F}_2^k$ as $\mathbf{x} = \mathbf{x}_1 \parallel \mathbf{x}_2 \parallel \cdots \parallel \mathbf{x}_w$ where each $\mathbf{x}_i$ has length k_i. Similarly, represent $\mathbf{y} \in \mathbb{F}_2^n$, as $\mathbf{y} = \mathbf{y}_0 \parallel \mathbf{y}_1 \parallel \mathbf{y}_2 \parallel \cdots \parallel \mathbf{y}_w$, where each $\mathbf{y}_i$ has length n_i and $|\mathbf{y}_0| = k$. We further identify $\mathbf{y}_0$ with $\mathbf{y}_0 = \mathbf{y}_0[1] \parallel \mathbf{y}_0[2] \parallel \cdots \parallel \mathbf{y}_0[w]$, where each $\mathbf{y}_0[i]$ is of length k_i.

During decoding, we will maintain lists $L_1, L_2, \ldots, L_w$ of possible decoding candidates of length K_i.

Step 0: Set a temporary list $T_0 = L_0$ to contain all possible decodings of the first k_1 coordinates of $\mathbf{y}$:

$$T_0 \leftarrow \{\mathbf{x}' = \mathbf{y}_0[1] + \mathbf{e} \mid \mathbf{e} \in E^{k_1/\ell}\}.$$

Step $1 \le i \le w$*:* Perform list-decoding to recover a list of valid decodings:

For each candidate $\mathbf{x}' \in T_{i-1} \subset \mathbb{F}_2^{K_i}$, add to L_i all the candidates for which $\mathbf{x}'B_i + \mathbf{y}_i \in E^{n_i/\ell}$:

$$L_i \leftarrow \{\mathbf{x}' \in T_{i-1} \mid \mathbf{x}'B_i + \mathbf{y}_i \in E^{n_i/\ell}\}. \tag{2}$$

If $i < w$ then create the temporary list T_i of candidates of length K_{i+1} from L_i:

$$T_i \leftarrow \{\mathbf{x}' \parallel (\mathbf{y}_0[i+1] + \mathbf{e}) \mid \mathbf{x}' \in L_i,\ \mathbf{e} \in E^{k_{i+1}/\ell}\}. \tag{3}$$

Return: L_w.

A key parameter of the public-key encryption scheme where we split the staircase-generator matrix is the number of splits s, that determines the number of summands the generator matrix of the code is split in. This parameter further determines the nature of the error used during encryption.

Definition 3. *Let $E_\ell \subset \mathbb{F}_2^\ell$ be an error set of granulation ℓ and let s denote the number of splits. The s-tuple $ErrorSplit = (e_1, \ldots, e_s)$, where $e_i \in \mathbb{F}_2^\ell, i \in \{1, \ldots, s\}$ is called* A Valid Error Split *for E_ℓ if the sum of its elements permuted with any permutation $\sigma_i \in \mathcal{S}_\ell$ is an element of E_ℓ i.e. it holds that $e = \sum_{i=1}^{s} \sigma_i(e_i) \in E_\ell$. The set of all valid error splits is denoted as $ValidErrorSplits$ and its size with V i.e. $V = |ValidErrorSplits|$.*

Example 1. Let $\ell = 4$, $E_\ell = \{(0,0,0,0), (0,0,1,1), (0,1,0,1), (0,1,1,0), (1,0,1,1), (1,1,0,0), (1,1,0,1), (1,1,1,0), (1,1,1,1)\}$ and $s = 4$. The 4-tuple $ErrorSplit = ((1,0,0,0), (1,1,1,1), (1,1,1,1), (1,1,0,1))$ is *a valid error split for E_ℓ* because the sum of all its elements permuted by any of all possible $4! = 24$ permutations always gives an element in E_ℓ.

In [7] we gave the following parameter sets for practical use:

Parameter Set 1. $l = 3$, $s = 2$, $E_3 = \{(0,0,1), (0,1,0), (1,0,0), (0,1,1), (1,0,1), (1,1,0)\}$, $ValidErrorSplits = \Big\{ \big((0,0,0),(0,0,1)\big), \big((0,0,0),(0,1,0)\big), \big((0,0,0),(0,1,1)\big), \big((0,0,0),(1,0,0)\big), \big((0,0,0),(1,0,1)\big), \big((0,0,0),(1,1,0)\big), \big((0,0,1),(0,0,0)\big), \big((0,0,1),(1,1,1)\big), \big((0,1,0),(0,0,0)\big), \big((0,1,0),(1,1,1)\big), \big((0,1,1),(0,0,0)\big), \big((0,1,1),(1,1,1)\big), \big((1,0,0),(0,0,0)\big), \big((1,0,0),(1,1,1)\big), \big((1,0,1),(0,0,0)\big), \big((1,0,1),(1,1,1)\big), \big((1,1,0),(0,0,0)\big), \big((1,1,0),(1,1,1)\big), \big((1,1,1),(0,0,1)\big), \big((1,1,1),(0,1,0)\big), \big((1,1,1),(0,1,1)\big), \big((1,1,1),(1,0,0)\big), \big((1,1,1),(1,0,1)\big), \big((1,1,1),(1,1,0)\big)\Big\}$.
Note that $V = |ValidErrorSplits| = 24$. The defining polynomial for E_3 is $p_d = 1 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3$, and the density $\rho_3 = |E_3|^{1/\ell} = 6^{1/3}$.

Parameter Set 2. $l = 4$, $s = 2$, and $E_4 = \{(0,0,0,1), (0,0,1,0), (0,1,0,0), (1,0,0,0), (0,0,1,1), (0,1,1,0), (0,1,0,1), (1,0,0,1), (1,0,1,0), (1,1,0,0)\}$, $ValidErrorSplits =$
$\Big\{ \big((0,0,0,0),(0,0,0,1)\big), \big((0,0,0,0),(0,0,1,0)\big), \big((0,0,0,0),(0,0,1,1)\big), \big((0,0,0,0),(0,1,0,0)\big), \big((0,0,0,0),(0,1,0,1)\big), \big((0,0,0,0),(0,1,1,0)\big), \big((0,0,0,0),(1,0,0,0)\big), \big((0,0,0,0),(1,0,0,1)\big), \big((0,0,0,0),(1,0,1,0)\big), \big((0,0,0,0),(1,1,0,0)\big), \big((0,0,0,1),(0,0,0,0)\big), \big((0,0,1,0),(0,0,0,0)\big), \big((0,0,1,1),(0,0,0,0)\big), \big((0,0,1,1),(1,1,1,1)\big), \big((0,1,0,0),(0,0,0,0)\big), \big((0,1,0,1),(0,0,0,0)\big), \big((0,1,0,1),(1,1,1,1)\big), \big((0,1,1,0),(0,0,0,0)\big), \big((0,1,1,0),(1,1,1,1)\big), \big((0,1,1,1),(1,1,1,1)\big), \big((1,0,0,0),(0,0,0,0)\big), \big((1,0,0,1),(0,0,0,0)\big), \big((1,0,0,1),(1,1,1,1)\big), \big((1,0,1,0),(0,0,0,0)\big), \big((1,0,1,0),(1,1,1,1)\big), \big((1,0,1,1),(1,1,1,1)\big), \big((1,1,0,0),(0,0,0,0)\big), \big((1,1,0,0),(1,1,1,1)\big), \big((1,1,0,1),(1,1,1,1)\big), \big((1,1,1,0),(1,1,1,1)\big), \big((1,1,1,1),(0,0,1,1)\big), \big((1,1,1,1),(0,1,0,1)\big), \big((1,1,1,1),(0,1,1,0)\big), \big((1,1,1,1),(0,1,1,1)\big), \big((1,1,1,1),(1,0,0,1)\big), \big((1,1,1,1),(1,0,1,0)\big), \big((1,1,1,1),(1,0,1,1)\big), \big((1,1,1,1),(1,1,0,0)\big), \big((1,1,1,1),(1,1,0,1)\big), \big((1,1,1,1),(1,1,1,0)\big)\Big\}$.
Note that $V = |ValidErrorSplits| = 40$. The defining polynomial for E_4 is $p_d = 1 + x_1 + x_2 + x_3 + x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$. Here, the density is $\rho_4 = |E_4|^{1/\ell} = 10^{1/4}$.

Algorithm 2 Key Generation

Parameters: Let $\ell | n$, $m = n/\ell$ and $E \subset \mathbb{F}_2^{\ell}$ be an error set of granulation ℓ and density ρ. Let s be the number of splits.

Key generation:
The following matrices make up the private key: - A generator matrix G of a binary (n, k) code of the form (1).
- An invertible matrix $S \in \mathbb{F}_2^{k \times k}$.
- An array of permutation matrices $P_1, P_2, \dots, P_s$ created as follows:

1. Select a permutation π on $\{1, 2, \dots, m\}$, and let $P \in \mathbb{F}_2^{n \times n}$ be the permutation matrix induced by π, so that for any $\mathbf{y} = \mathbf{y}_1 \parallel \mathbf{y}_2 \parallel \dots \parallel \mathbf{y}_m \in (\mathbb{F}_2^{\ell})^m$:

$$\mathbf{y}P = \mathbf{y}_{\pi(1)} \parallel \mathbf{y}_{\pi(2)} \parallel \dots \parallel \mathbf{y}_{\pi(m)}, \tag{4}$$

i.e., P only permutes the m substrings of $\mathbf{y}$ of length ℓ.

2. For $i := 1$ to s:

– Select randomly m permutations $\sigma_j^i \in \mathcal{S}_\ell$, $j \in \{1, \dots, m\}$.
– Let P_i be defined by

$$\mathbf{y}P_i = \sigma_1^i(\mathbf{y}_{\pi(1)}) \parallel \sigma_2^i(\mathbf{y}_{\pi(2)}) \parallel \dots \parallel \sigma_m^i(\mathbf{y}_{\pi(m)}),$$

where $\sigma_j^i(\mathbf{x}) = \sigma_j^i(x_1, x_2, \dots, x_\ell)$.

3. The public key is formed as follows:

– Generate uniformly at random $s - 1$ matrices $G_1, \dots, G_{s-1}$ of size $k \times n$ over $\mathbb{F}_2$.
– Set $G_s = G + G_1 + \dots + G_{s-1}$.
– For all $i \in \{1, 2, \dots, s\}$, set $G_{\text{pub}}^i = SG_iP_i$.

Public key: $G_{\text{pub}}^1, \dots, G_{\text{pub}}^s$.

Private key: S, G and $P_1, P_2, \dots, P_s$.

Algorithm 3 Valid Error Splits (ℓ, E_ℓ, s)

Input: Granulation ℓ, error set E_ℓ, number of splits s.
Output: A set $ValidErrorSplits$ of all possible valid error splits.

```
1: Set $ValidErrorSplits \leftarrow \emptyset$
2: for all $(e_1, \dots, e_s) \in (\mathbb{F}_2^{\ell})^s$ do
3:     if $\sum_{i=1}^{s} \sigma_i(e_i) \in E_\ell, \forall(\sigma_1, \dots, \sigma_s) \in (\mathcal{S}_\ell)^s$ then
4:         Add $(e_1, \dots, e_s)$ to $ValidErrorSplits$.
5:     end if
6: end for
7: Return $ValidErrorSplits$.
```

Note that Algorithm 3 is run only once at the time of the initialization of the system with parameters ℓ, E_ℓ, s. Even more, in practice, this set can be pre-calculated and publicly available.

A randomized decoding algorithm, Alg. 4, that is suitable for finding digital signatures was presented in [2]. Since Alg. 4 is a randomized version of the Alg. 1 we need a condition that guarantees that the signing process will find a signature with high probability.

Proposition 2. *Algorithm 4 produces a signature with probability more than* $1/2$ *if the following two conditions hold:*

1. $\mathsf{ExpLimit}_i > (2/\rho)^{n_i}$, *for* $1 \leq i \leq w - 1$;
2. $\rho^{k_w + n_w} 2^{-n_w} > 1$.

A formal description of the signature scheme is given through the algorithms for key generation, error set generation, decoding for signatures, signing and verification. We describe the signing and verification procedures in detail in Alg. 5 and 6, respectively.

Algorithm 4 Decoding for signatures

Input: A vector $\mathbf{y} \in \mathbb{F}_2^n$, and a generator matrix G of the form (1).

Output: A valid decoding $\mathbf{s} \in \mathbb{F}_2^k$ of $\mathbf{y}$.

Procedure:
The notation is the same as in Alg. 1, with the addition of the variables $\mathsf{ExpLimit}_i \leq \rho^{n_i}$. The decoding proceeds in two phases:

Phase 1: Find a valid decoding $\mathbf{x}'$ of $\mathbf{y}_0[1]$ with respect to B_1 and $\mathbf{y}_1$. That is, find an $\mathbf{x}' \in \mathbb{F}_2^{k_1}$ so that $\mathbf{x}'B_1 + \mathbf{y}_1 \in E^{n_1/\ell}$, trying at most $\mathsf{ExpLimit}_1$ candidates. Expand $\mathbf{x}'$ into at most $\mathsf{ExpLimit}_2$ candidates of length $k_1 + k_2$ by appending the sum of $\mathbf{y}_0[2]$ with random errors from $E^{k_2/\ell}$, until you find a valid decoding with respect to B_2 and $\mathbf{y}_2$ (if no valid candidate can be found, start over with a new initial $\mathbf{x}'$). Continue this process for $(B_3, \mathbf{y}_3)$, $(B_4, \mathbf{y}_4), \ldots$

Phase 2: Once you have found a candidate that is valid for $B_1, B_2, \ldots, B_{w-1}$ and $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{w-1}$, switch to the list-decoding algorithm described in Alg. 1 for the last block, i.e. for $i = w - 1$.

Return: $\mathbf{s} \leftarrow L_w$.

Algorithm 5 Signing

Input: A value $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_s) \in (\mathbb{F}_2^n)^s$ to be signed. The private key: S, G and $P_1, P_2, \ldots, P_s$.

Output: A valid signature $\boldsymbol{\sigma} \in \mathbb{F}_2^k$, so that
$\mathbf{e}_i = \boldsymbol{\sigma} G^i_{\text{pub}} + \mathbf{z}_i$, $i = 1, \ldots, s$, where $\mathbf{e}_i = (e_{1,i}, \ldots, e_{\frac{n}{l},i})$ and where $\forall j \in \{1, \ldots, \frac{n}{l}\}$, $(e_{j,1}, \ldots, e_{j,s}) \in ValidErrorSplits$.

Procedure:

```
1: Set $\mathbf{y}_i = \mathbf{z}_i P_i^{-1}$
2: Set $\mathbf{y} = \sum_{i=1}^{s} \mathbf{y}_i$
3: Decode $\mathbf{y}$ using Alg. 4, to get a valid decoding $\mathbf{s}$.
4: Set the signature $\boldsymbol{\sigma} = \mathbf{s}S^{-1}$.
5: Return $\boldsymbol{\sigma}$
```

Algorithm 6 Verification

Input: A pair $(\mathbf{z}, \boldsymbol{\sigma}) \in (\mathbb{F}_2^n)^s \times \mathbb{F}_2^k$, and the public key $G^1_{\text{pub}}, \ldots, G^s_{\text{pub}}$.
Output: Accept or Reject.

```
1: Set $\mathbf{e}_i = \boldsymbol{\sigma} G^i_{\text{pub}} + \mathbf{z}_i$, for $i = 1, \ldots, s$
2: if $\forall j \in \{1, \ldots, \frac{n}{l}\}$,
   $(e_{j,1}, \ldots, e_{j,s}) \in ValidErrorSplits$ then
3:     Return Accept
4: else
5:     Return Reject
6: end if
```

3 Security analysis

The initial proposals both for encryption and signatures that use St-Gen codes [2], are vulnerable to an Information Set Decoding (ISD) attack [8,5]. Additionally, as it is shown in [5], the signature scheme is vulnerable against a very efficient forgery attack. In the following two sections we analyze the security of the signature scheme introduced in this paper against the forgeries attacks.

3.1 Resistance against ISD forgery attacks

First we give the Alg. 7 which is an adapted ISD attack for signature forgery (given as Alg. 1 in [5]) for the original St-Gen scheme with granularity $\ell = 2$ and the error set $E = \{00, 01, 10\}$.

Since in the original St-Gen signature scheme the error set E is such that when the first bit is fixed to 0, there are possibilities the second bit to be either 0 or 1, by choosing $e'_1 = 0$ in Step 3 of Alg. 7, we have that in Step 4 the computed error vector e will have valid pairs from the error set E for ℓk bits. To have a successful forgery, it remains to have valid errors from the set

Algorithm 7 Forgeries with ISD against original St-Gen scheme (adapted Alg. 1 in [5])

```
Input: A value $\mathbf{z} \in \mathbb{F}_2^n$ to be signed and a $k \times n$ public key matrix $G_{pub}$.
Output: A forged signature $\boldsymbol{\sigma} \in \mathbb{F}_2^k$.

1. Permute the bits of $\mathbf{z}$ into $\mathbf{z}'$ by a specially constructed random permutation matrix $P'$. The permutation is
   special in the way that it randomly picks $k$ pairs of bits, and takes every first bit in those pairs and puts it
   in the first $k$ bits of $\mathbf{z}'$. The permutation of the other bits of $\mathbf{z}$ is completely random.
2. Try to find a forged signature $\boldsymbol{\sigma}$ by establishing the following relation between $\mathbf{z}'$ and $\boldsymbol{\sigma}$:
       $\mathbf{z}' = \mathbf{z}P' = (\boldsymbol{\sigma} G_{pub} + e)P'$
       $= \boldsymbol{\sigma} G_{pub}P' + eP'$
       $= \boldsymbol{\sigma}(A|B) + (e_1'|e_2')$
       $= (\boldsymbol{\sigma} A + e_1')|(\boldsymbol{\sigma} B + e_2')$,
   where $A$ and $e_1'$ are are the first $k$ columns of the permuted generator matrix $G_{pub}P'$ and permuted error
   vector $eP'$, respectively.
2: If $A$ is not invertible, go to Step 1.
3: Set $e_1' = 0$ and compute
       $\boldsymbol{\sigma} = (\boldsymbol{\sigma} A)A^{-1}$.
4: Compute the error vector as
       $e = \mathbf{z} - \boldsymbol{\sigma} G_{pub}$.
5: If the error vector is properly formed i.e. it is composed of 2-bit substrings from the proper generalized error
   set $E$, then return $\boldsymbol{\sigma}$. Otherwise go back to Step 1 and start over with a new permutation $P'$.
```

E on the remaining $n - \ell k$ bits. The probability of that event is

$$Pr = \left(\frac{|E|}{2^\ell}\right)^{\frac{n-\ell k}{\ell}} = \left(\frac{\rho}{2}\right)^{n-\ell k}.$$

For the concrete value $\ell = 2$ the probability to have a successful forgery is $(\frac{\sqrt{3}}{2})^{n-2k}$ i.e., for all the practical parameters given in [2] it is very high and is around 2^{-8}.

Let us now analyze Alg. 8 which is an analog ISD forgery attack adapted for signing and verification with random split of the generator matrix given in Alg. 5 and 6.

There are several crucial differences between the attacks in Alg. 7 and in Alg. 8. First, the permutation matrices P' in Step 1 are produced differently. In Alg. 7 the permutation is special in the way that it randomly picks k pairs of bits, and takes every first bit in those pairs and puts them in the first k bits of $\mathbf{z}'$. This is beneficial for the attacker to do since the ℓ-bit substrings are never permuted internally. On the other hand, for the new signature scheme, picking some fixed position in the ℓ-bit substrings does not give any advantage since those substrings are permuted locally during the key-generation phase. As a consequence of the first difference, comes the second difference: fixing the value of $e_1' = 0$ in Step 3 does not give any advantage to the attacker in Alg. 8. Thus, the value of e_1' is randomly guessed. The third difference is in Step 4 where now instead of one error vector e we have to compute s error vectors $\mathbf{e}_i, i = 1, s$. The fourth and most important difference is in Step 5 where the successful forgery has to produce $\frac{n}{\ell}$ error s-tuples $(e_{j,1}, \dots, e_{j,s})$ that all belong to $ValidErrorSplits$. By assuming that there are always elements in the $ValidErrorSplits$ that as first component has the values $e_{j,1}$ as part of the guessed e_1',

Algorithm 8 Forgeries with ISD against signature scheme of Alg. 5 and 6

```
Input: A value $\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_s) \in (\mathbb{F}_2^n)^s$ to be signed and the public key of $s$ matrices of size $k \times n$: $G^1_{\text{pub}}, \dots, G^s_{\text{pub}}$.
Output: A forged signature $\boldsymbol{\sigma} \in \mathbb{F}_2^k$.

1. Permute the bits of $\mathbf{z}_1$ into $\mathbf{z}'_1$ by a random permutation matrix $P'$.
2. Try to find a forged signature $\boldsymbol{\sigma}$ by establishing the following relation between $\mathbf{z}'_1$ and $\boldsymbol{\sigma}$:
       $\mathbf{z}'_1 = \mathbf{z}_1 P' = (\boldsymbol{\sigma} G^1_{\text{pub}} + e)P'$
            $= \boldsymbol{\sigma} G^1_{\text{pub}} P' + eP'$
            $= \boldsymbol{\sigma}(A|B) + (e'_1|e'_2)$
            $= (\boldsymbol{\sigma} A + e'_1)|(\boldsymbol{\sigma} B + e'_2)$,
   where $A$ and $e'_1$ are the first $k$ columns of the permuted generator matrix $G^1_{\text{pub}}P'$ and permuted error vector $eP'$, respectively.
2: If $A$ is not invertible, go to Step 1.
3: Guess $e'_1$ and compute
       $\boldsymbol{\sigma} = ((\boldsymbol{\sigma} A + e'_1) - e'_1)A^{-1}$.
4: For all $i \in \{1, \dots, s\}$ compute error vectors
       $\mathbf{e}_i = \mathbf{z}_1 - \boldsymbol{\sigma} G^i_{\text{pub}} = (e_{1,i}, \dots, e_{\frac{n}{l},i})$.
5: Check whether $\forall j \in \{1, \dots, \frac{n}{l}\}$, $(e_{j,1}, \dots, e_{j,s}) \in ValidErrorSplits$. If yes, then return $\boldsymbol{\sigma}$, otherwise go back to Step 1 and start over with a new permutation $P'$.
```

the probability of successful forgery is upper bounded by

$$\left(\frac{|V|}{2^{\ell s}}\right)^{\frac{n}{\ell}-1}. \tag{5}$$

3.2 Resistance against algebraic ISD forgery attacks

The attack described in the previous section does not exploit the structure of the generator matrix, but rather the specific structure of the error set. Here, we show that such an attack, and even a more general one, can always be modeled as a system of equations.

Recall that we have defined the error set of our scheme using a defining polynomial p_d in ℓ variables, whose roots are exactly the possible errors used in the encoding and decoding algorithms of the scheme. Therefore, the set $ValidErrorSplits$ can also be defined using the polynomial p_d as the set of all s-tuples $(e_1, \dots, e_s)$, that satisfy the equation

$$p_d(\sum_{i=1}^{s} \sigma_i(e_i)) = 0$$

for every permutation $\sigma_i \in \mathcal{S}_\ell$.

This means that the verification process will accept a signature $(\mathbf{z}, \mathbf{x}) \in (\mathbb{F}_2^n)^s \times \mathbb{F}_2^k$, if for $\mathbf{e}_i = \mathbf{x}G^i_{\text{pub}} + \mathbf{z}_i,$, $i = 1, \dots, s$

$$p_d(\sum_{i=1}^{s} \sigma_i(e_{j,i})) = 0, \quad \forall j \in \{1, \dots, \frac{n}{\ell}\}, \ \forall \sigma_i \in \mathcal{S}_\ell. \tag{6}$$

Taking $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ to be an unknown signature for a message $\mathbf{z}$, (6) can be seen as a system of equations whose solution is a valid signature.

While the modeling as a system of equations is fairly natural, if the defining polynomial is of degree at least 2, solving the system 6 is in general a hard problem.

A good strategy to speed up the process of solving the system is to use linearization, i.e. to add linear equations to the system that hold with some reasonably high probability. For example, the simplest linear equation that can be added is $x_i = 0$ or $x_i + 1 = 0$ which corresponds to fixing the value of some variable x_i. In the case of our scheme, since the defining polynomial p_d acts on relatively small number $s\ell$ of variables, it is possible to analyze it, and find the linear equations that hold with the highest probability. Note that the lower the nonlinearity of p_d is, the higher the probability of a linear equation.

For a moment assume that the system 6 has a unique solution. Denote by p the probability that some linear equation

$$P(e_{j+1,1}, \ldots, e_{j+1,s}, e_{j+2,1}, \ldots, e_{j+2,s}, \ldots, e_{j+\ell,1}, \ldots, e_{j+\ell,s}) = 0 \tag{7}$$

on a block of $s\ell$ variables holds. It can be shown that once the equations (7) for every $j \in \{1, \ldots, \frac{n}{l}\}$ are added to the system (6), the system becomes easy to solve. However, the probability that (7) holds for every j becomes $p^{\frac{n}{l}}$. For large enough n, this probability is very small, so it is very unlikely that the system will have a solution with the added linear equations. Even if we randomly pick only $\frac{k}{l}$ blocks (which could be enough to solve the system), the probability that the system will have a solution, $p^{\frac{k}{l}}$, is still negligible.

The situation is very different if the system (6) has many solutions, i.e. if there exist many valid signatures for a given message. Denote by N_{vs} the number of valid signatures for a given message. Then the probability that the equations (7) for some k randomly chosen $j \in \{1, \ldots, \frac{n}{l}\}$ are valid for at least one of the possible signatures becomes

$$Pr_A = 1 - (1 - p^{\frac{k}{l}})^{N_{vs}} \tag{8}$$

Using Proposition 1, we can estimate the number of valid signatures by

$$N_{vs} = |E|^{\frac{n}{\ell}} 2^{k-n}, \tag{9}$$

and thus the probability of a successful algebraic attack becomes

$$Pr_A = 1 - (1 - p^{\frac{k}{l}})^{|E|^{\frac{n}{\ell}} 2^{k-n}} \tag{10}$$

The probability (10) can be well approximated using Poisson approximation with parameter $\lambda = p^{\frac{k}{l}} N_{vs}$, since for realistic parameters the number of valid signatures is large enough. Thus we have

$$Pr_A \approx 1 - e^{-\lambda} = 1 - e^{-p^{\frac{k}{l}} N_{vs}}. \tag{11}$$

The crucial parameter for success of the attack is λ, and as $\lambda \to 0$, $Pr_A \to 0$. In order for the signature scheme to be secure against such attacks, λ has to be very small. Informally speaking, this means that the number of valid signatures has to be as small as possible, or in other words the rate of the code should be as small as possible. Clearly, this introduces a trade-off between security and efficiency, and the parameters should be carefully chosen.

3.3 Resistant against ISD distinguishing and key recovery attacks

In the analysis by Moody and Perlner [5] a modification of Stern's algorithm [9] was provided, dedicated to cryptanalysis of the scheme in [2]. We refer the reader to [5] for details, and here we mention that the complexity of the key recovery attack is in general given by

$$ISD_{St} = Pr_{St}^{-1} \cdot Cost_{St}$$

where Pr_{St} is the probability of success, and $Cost_{St}$ the cost of finding the low weight codeword. In [7] we gave a detailed security analysis how and why the random split of the generator matrix prevents from ISD attacks.

The main logical basis for construction of a key recovery attack on the signature scheme is the fact that n_w is relatively small in comparison to n. That structural property of the generator matrix gives a strategy for the attacker: Learn which columns of the public key belong to the B_w part of the private key G.

For the signature scheme with a random split of G the probability of a success of the original Pr_{St} is decreased to the value

$$Pr_{St} \cdot \left(\frac{1}{\ell!}\right)^{\frac{s(k_w+n_w)}{\ell}}. \tag{12}$$

4 Concrete parameter sets and their security

Based on Eq. (5) and (12) in this section we propose two concrete codes from the two parameter sets given in Sec. 2, two of each set, providing security of 128 bits.

We denote by $K = (k_1, \ldots, k_w)$ and $N = (n_1, \ldots, n_w)$ the vectors of values used in the definition of concrete generator matrices as defined in equation (1).

Parameter Set 1:

- Code $(2142, 465)$,
  $w = 27$, $K = (75, 15, 15, \ldots, 15)$, $N = (60, 60, \ldots, 60, 60, 117)$.

Parameter Set 2:

- Code $(2244, 508)$.
  $w = 28$, $K = (76, 16, 16, \ldots, 16)$, $N = (60, 60, \ldots, 60, 60, 116)$.

We have implemented both instances in Magma [1] and we can confirm that these parameter sets are practical.

5 Conclusions

We have presented a signature scheme based on random split of St-Gen codes where we split the generator matrix into s randomly generated matrices. The split strategy is used to thwarts the distinguishing key recovery and ISD forgery attacks on the initial St-Gen signature scheme.

References

1. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993). (Cited on page 9.)
2. Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. McEliece in the world of Escher. Cryptology ePrint Archive, Report 2014/360, 2014. `http://eprint.iacr.org/`. (Cited on pages 1, 2, 4, 5, 6, and 9.)
3. Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. A new code based public key encryption and signature scheme based on list decoding. Presented at ?Workshop on Cybersecurity in a Post-Quantum World,? NIST, Gaithersburg MD, USA, 2015. `http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm`, [Retrieved: October 2015]. (Cited on page 1.)
4. R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44. (Cited on page 1.)
5. Dustin Moody and Ray Perlner. Vulnerabilities of "McEliece in the World of Escher". Cryptology ePrint Archive, Report 2015/966, 2015. `http://eprint.iacr.org/`. (Cited on pages 1, 5, 6, and 9.)
6. Simona Samardjiska and Danilo Gligoroski. Approaching maximum embedding efficiency on small covers using staircase-generator codes. In *Information Theory (ISIT), 2015 IEEE International Symposium on*, pages 2752–2756, June 2015. (Cited on page 1.)
7. Simona Samardjiska and Danilo Gligoroski. An Encryption Scheme based on Random Split of St-Gen Codes. Cryptology ePrint Archive, Report 2016/202, 2016. `https://eprint.iacr.org/2016/202`. (Cited on pages 1, 3, and 9.)
8. Nicolas Sendrier and Jean-Pierre Tillich. Private communication, October 2014. (Cited on pages 1 and 5.)
9. Jacques Stern. A method for finding codewords of small weight. In *Proceedings of the 3rd International Colloquium on Coding Theory and Applications*, pages 106–113, London, UK, UK, 1989. Springer-Verlag. (Cited on page 9.)