Designs, Codes and Cryptography (2025) 93:2503–2519
https://doi.org/10.1007/s10623-025-01603-1

How to lose some weight: a practical template syndrome decoding attack

Sebastian Bitzer[1] · Jeroen Delvaux[2] · Elena Kirshanova[2] · Sebastian Maaßen[3] · Alexander May[3] · Antonia Wachter-Zeh[1]

Received: 15 September 2024 / Revised: 20 January 2025 / Accepted: 21 February 2025 /
Published online: 7 March 2025

Abstract
We study the hardness of the Syndrome Decoding problem, the base of most code-based cryptographic schemes, such as Classic McEliece, in the presence of side-channel information. We use ChipWhisperer equipment to perform a template attack on Classic McEliece running on an ARM Cortex-M4, and accurately classify the Hamming weights of consecutive 32-bit blocks of the secret error vector $\mathbf{e} \in \mathbb{F}_2^n$. With these weights at hand, we optimize Information Set Decoding algorithms. Technically, we demonstrate how to speed up information set decoding via a dimension reduction, additional parity-check equations, and an improved information set search, all derived from the Hamming-weight information. Consequently, using our template attack, we can practically recover an error vector $\mathbf{e} \in \mathbb{F}_2^n$ in dimension $n = 2197$ in a matter of seconds. Without side-channel information, such an instance has a complexity of around 88 bit. We also estimate how our template attack affects the security of the proposed McEliece parameter sets. Roughly speaking, even an error-prone leak of our Hamming weight information leads for $n = 3488$ to a security drop of 89 bits.

Keywords Code-based cryptography · Side-channel attacks · Classic McEliece · Information set decoding

✉ Sebastian Bitzer
sebastian.bitzer@tum.de

Jeroen Delvaux
jeroen.delvaux@tii.ae

Elena Kirshanova
elena.kirshanova@tii.ae

Sebastian Maaßen
sebastian.maassen@ruhr-uni-bochum.de

Alexander May
alex.may@rub.de

Antonia Wachter-Zeh
antonia.wachter-zeh@tum.de

[1] Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany

[2] Technology Innovation Institute, Abu Dhabi, UAE

[3] Ruhr University Bochum, Bochum, Germany

Mathematics Subject Classification 94A60 · 94B05 · 68Q25

1 Introduction

Hardness of syndrome decoding. Central to most code-based schemes, in particular those that advanced to the 4th Round of the NIST Post-Quantum Standardization Process [2, 4, 26], lies the *Syndrome Decoding* (SD) *problem*: given a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, where $\mathbb{F}_2$ denotes the binary finite field, a syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$, and an integer $w < n$, find the error vector $\mathbf{e}$ such that $\mathbf{He} = \mathbf{s}$ and $|\mathbf{e}| < w$, where $|\cdot|$ denotes the Hamming weight.

An algorithm for solving this problem for a uniformly random $\mathbf{H}$ leads to a message or key recovery attack for the aforementioned schemes. Therefore, the syndrome decoding problem has received a significant amount of attention, resulting in various methods to solve it: Information Set Decoding (ISD) [25, 28, 30], Statistical Decoding [1, 8], and, recently, Sieving-style algorithms [13, 22]. Despite this extensive theoretical effort, the problem remains tractable for relatively small dimensions. Concretely, in the setting of Classic McEliece (e.g., $w \approx \frac{n}{5\log_2 n}$ and $k \approx 0.8n$), the largest solved instance reported today [27] on decodingchallenge.org [3] is for $n = 1409$, and it already requires an optimized GPU implementation of an advanced information set decoding algorithm [25], together with significant computational resources.[1]

Side-channel attacks. For the practical security of code-based schemes, it is important that the syndrome decoding problem also offers sufficient robustness against realistic side-channel attacks using leaks of the secret error vector $\mathbf{e} \in \mathbb{F}_2^n$. Compared to the comprehensive study of the syndrome decoding problem's classical security, its side-channel resistance has received much less attention.

Initial theoretical work of Horlemann et al. [23] classifies different leakages and shows how to incorporate them into ISD algorithms to solve the syndrome decoding problem faster. One of the leakages considered in [23, Sect. 4] is *known Hamming weights of error blocks*.

In this leakage setting, one knows $\{|\mathbf{e}_i|\}_{i\le t}$, where $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_t)$ and all $\mathbf{e}_i$'s (except, may be the last $\mathbf{e}_t$) are of the same length, i.e., the *word size* of the Central Processing Unit (CPU). For example, for an ARM Cortex-M4, each word $\mathbf{e}_i$ consists of 32 bits. Typical target instructions are *loads*, which move 32-bit words from SRAM to CPU registers, and *stores*, which move 32-bit words from CPU registers to SRAM. When executing such instructions, the power consumption is slightly different for each possible weight $|\mathbf{e}_i|$, and these unique characteristics can be condensed into a so-called *template* [9]. We call the respective modified syndrome decoding problem, which additionally receives $\{|\mathbf{e}_i|\}_{i\le t}$, the *template syndrome decoding* (template SD) problem.

While Horlemann et al. [23] describe a potential template syndrome decoding attack, their attack remains purely theoretical. Neither do the authors realize concrete power trace leaks, nor do they provide an improved information set decoding implementation. Thus, the practical implications of code-based template attacks remain unclear.

Contribution. In this work, we perform for the first time an explicit template attack on a Classic McEliece implementation. To this end, we realize a concrete power trace leak, from which we derive with high accuracy (but still error-prone) the desired Hamming weight information $\{|\mathbf{e}_i|\}_{i\le t}$.

[1] See https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/WzgqEmAfnk8 for a discussion on hardness predictions for this instance.

We then improve information set decoding by using and enhancing the techniques of Horlemann et al. [23]. Building on information set decoding software from Esser, May, and Zweydinger [16], we provide a concrete implementation of these improvements.

With our (erroneous but easily correctable) leakage, we run our template information set decoding and retrieve the secret $\mathbf{e} \in \mathbb{F}_2^n$. Concretely, we are able to solve the template syndrome decoding problem for Classic McEliece in dimension $n = 2197$ in a matter of seconds. Without template, such an instance has complexity around 88 bits. In more detail, our results are as follows.

1. We use ChipWhisperer equipment to measure the power consumption of an open-source implementation [11] of Classic McEliece running on an ARM Cortex-M4, or at least a decapsulation subroutine that checks whether $|\mathbf{e}| = w$. Using $48k$ traces for template building, and $12k$ for matching, the weights $\{|\mathbf{e}_i|\}$ we recover are correct with a probability of around 97 %.
2. We modify the ISD algorithms of Prange [28] and Dumer [14] by incorporating the template. Specifically, we show how to encode the knowledge of weights of error blocks into the parity-check matrix $\mathbf{H}$. Then, using such modified $\mathbf{H}$, we show how to decrease the expected running time of the above ISD algorithms, again exploiting the leakage.
3. We develop a preprocessing algorithm which is capable of computing suitable Hamming-weight information from noisy measurement data. Utilizing inherent redundancy and naturally present reliability information, we make template ISD algorithms resilient to measurement errors.
4. We provide efficient and parallelized implementations of the modified ISD algorithms. With our software we are able to solve the $n = 2197$ instance from [3] in a matter of 10 s on AMD EPYC 7742 using 200 threads. Based on our implementation, we estimate the hardness of larger McEliece instances under this template attack.

Related work. Closely related to the template syndrome decoding is *regular syndrome decoding* introduced in [5]. In regular SD, for each block $\mathbf{e}_i$ of $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_t)$ it holds that $|\mathbf{e}_i| = 1$. Note that regular SD is a special case of Template SD. Recent work of Esser and Santini [17] studies the hardness of regular SD, and some of their ideas apply to our setting, e.g., the construction of new parity-check equations, see also [16].

Another template attack on Classic McEliece was presented by Grosso et al. in [21]. The authors of [21] aim at the same leakage, namely, $\{|\mathbf{e}_i|\}_{i \le t}$ but they retrieve it from the matrix–vector multiplication $\mathbf{H} \cdot \mathbf{e}$ that computes the syndrome. In our template attack, similar to [21], we discard the columns of $\mathbf{H}$ that correspond to the zero-weight blocks in the template. Contrary to [21], in our work we show how to make use of the *non-zero* weight blocks to speed-up ISD algorithms, and we implement our ISD algorithms in order to actually retrieve the secret.

Another side-channel attack exploiting failures of the decoding procedure in McEliece decryption is studied in [24]. The authors show how to learn the positions of 1's in the secret vector by querying the decoder with modified syndromes. Similar to our work, the authors combine the obtained information with ISD algorithms and estimate their attack performance. In contrast, we implement our (modified) ISD routines, report on concrete runtimes for feasible instance and then give estimates for large dimensions.

In summary, in contrast to [21] and [24] we do not only *estimate* the effects on ISD, but we retrieve Hamming weight side-channel information, correct errors, provide improved ISDs via dimension reduction and additional parity-check equations, and *practically solve* an $n = 2197$-dimensional template SD instance in a matter of seconds.

Artifacts. Our software for template ISD algorithms as well as scripts to generate the figures are available in [6].

2 Template ISD

Notations. Let $|\mathbf{x}|$ denote the Hamming weight of $\mathbf{x}$, and let $[i, j)$ be the interval of consecutive integers $\{i, i+1, \ldots, j-1\}$. By S_n we denote the group of all permutations on sets of size n. By $\mathbb{I}_n$ we denote the identity matrix of rank n.
Coding Theory. A binary linear code is a k-dimensional subspace of $\mathbb{F}_2^n$. Thus it is the kernel of some $(n-k)$-rank matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, i.e., for every codeword $\mathbf{c}$ it holds that $\mathbf{Hc} = 0$. We consider random linear codes meaning that the entries of $\mathbf{H}$ are independently uniformly distributed in $\mathbb{F}_2$. For a vector $\mathbf{x} \in \mathbb{F}_2^n$, its syndrome is defined as $\mathbf{s} = \mathbf{Hx} \in \mathbb{F}_2^{n-k}$.
Problem definitions. In the Classic McEliece KEM [2], the decryption process receives as input a syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$ and recovers the secret message $\mathbf{e}$ by calling an efficient syndrome decoder using the McEliece secret key. Once $\mathbf{e}$ is retrieved, the decryption checks if $|\mathbf{e}| = w$, where w is the decoding capacity of the syndrome decoder. The parameter w is a fixed public parameter. Classic McEliece decryption only returns $\mathbf{e}$, if $\mathbf{e}$ passes the check $|\mathbf{e}| = w$.

Without knowledge of the secret key, message recovery attacks on Classic McEliece require solving the Syndrome Decoding (SD) problem.

Definition 1 (*Syndrome Decoding (SD)*) Let $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ be a parity-check matrix, $\mathbf{e}$ an error vector of Hamming weight w, and $\mathbf{s} = \mathbf{He} \in \mathbb{F}_2^{n-k}$ the corresponding syndrome. SD asks to find the unique weight-w $\mathbf{e} \in \mathbb{F}_2^n$ satisfying $\mathbf{He} = \mathbf{s}$.

The side-channel attack we consider in this work creates a template for the function that computes $|\mathbf{e}|$. In the ideal scenario, such a template allows the attacker to learn the blockwise weight of $\mathbf{e}$. We call the SD problem that, in addition, receives the blockwise weight *template Syndrome Decoding*.

Definition 2 (*Template SD*) Let $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ be a parity-check matrix of a code and $\mathbf{s} = H\mathbf{e} \in \mathbb{F}_2^{n-k}$, for some $\mathbf{e}$ of Hamming weight w. Let further $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_t)$ with $\mathbf{e}_i \in \mathbb{F}_2^b$ for $i = [1, t)$, $\mathbf{e}_t \in \mathbb{F}_2^{n-b\cdot(t-1)}$, and $w_i = |\mathbf{e}_i|$. Template ISD asks to find $\mathbf{e}$ given $\mathbf{H}$, $\mathbf{s}$, and $\{w_i\}_{i\le t}$.

Definition 3 (*Guess*) We call any vector $\{\hat{w}_i\}_{i\le t} \in \mathbb{N}_0^t$ a *guess*. The *accuracy* of a guess is the percentage of correctly identified weights, i.e. $\frac{|\{i\in[1,t]|\hat{w}_i=w_i\}|}{t}$. A guess is *error-free* if it has accuracy 1; otherwise, it is *error-prone*. Notice that, in general, error-prone guesses do not satisfy $\sum_{i=1}^t \hat{w}_i = w$.

The block size b, and, therefore, also the template's length depends on the target architecture's specifications. All the presented techniques are compatible with various block sizes, such as $b = 8, b = 32$, or $b = 64$. Increasing the block size reduces the available side-channel information, thereby increasing the cost of solving Template SD. Conversely, guesses with $b = 8$ typically contain sufficient information to make recovering $\mathbf{e}$ trivial for a wide range of parameters. In this work, we focus on $b = 32$, which aligns with the architecture of the ARM Cortex-M4 processor. Hence, our guesses will be of length $t = \lceil n/32 \rceil$.

Example 1 (*Running example* $n = 2197$) Our running example uses the parameters $n = 2197$, $k = 1758$, and $w = 37$. Therefore, a guess is a string of length $t = 69$, its ith entry indicating the weight of the ith 32-bit block of $\mathbf{e}$.

3 Algorithms for template ISD

3.1 Permutation-based template ISD—improving Prange

Let us start with the fundamental information set decoding algorithm due to Prange [28], given in Algorithm 1. Prange's algorithm permutes the columns of $\mathbf{H}$, which is equivalent to permuting the positions of 1's in $\mathbf{e}$.

Let $\pi \in S_n$ be a permutation and let $\pi(\mathbf{H}) = (\mathbf{Q} \mid \cdot)$ be the result of applying the permutation π to $\mathbf{H}$ such that $\mathbf{Q} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ is invertible (this event occurs with constant probability). Multiplying by $\mathbf{Q}^{-1}$ from the left both $\pi(H)$ and $\mathbf{s}$ leads to an equivalent SD instance written in systematic form:

$$\mathbf{H}'\pi(\mathbf{e}) = \mathbf{s}', \quad \text{where} \quad \mathbf{H}' = \mathbf{Q}^{-1}\mathbf{H} = (\mathbb{I}_{n-k} \mid \cdot), \text{ and } \mathbf{s}' = \mathbf{Q}^{-1}\mathbf{s}.$$

If $\pi(\mathbf{e})$ has weight 0 on the last k coordinates, then $|\mathbf{s}'| = w$. This means that the first $(n-k)$ coordinates of $\pi(\mathbf{e})$ are identical to $\mathbf{s}'$, and hence $\mathbf{e}$ can be reconstructed.

Algorithm 1: Prange

Input : $\mathbf{H}$, $\mathbf{s}$, w
Output: $\mathbf{e}$

1 **repeat**
2 Sample $\mathbf{P} \in S_n$
3 Let $\mathbf{H}' = \mathbf{Q}^{-1}\mathbf{HP}$ be the systematic form of $\mathbf{HP}$
4 $\mathbf{s}' = \mathbf{Q}^{-1}\mathbf{s}$
5 **until** $|\mathbf{s}'| = w$
6 **return** $\mathbf{P}^{-1} \cdot (\mathbf{s}', 0^k)$.

Dimension reduction. As already noticed in the work of Grosso et al. [21], any weight-0 block does not contribute to the solution $\mathbf{e}$. Let m_0 denote the number of error-free blocks of length b. Then $b \cdot m_0$ columns of $\mathbf{H}$ do not contribute and can be eliminated, leading to a modified parity-check matrix $\bar{\mathbf{H}} \in F_2^{(n-k)\times\bar{n}}$ with $\bar{n} = n - b \cdot m_0$ columns. This in turn reduces the dimension of the solution $\mathbf{e}$ from n to $\bar{n} = n - b \cdot m_0$, while leaving its weight w unchanged.

Improved permutation. The idea of the permutation π in Prange's algorithm is to move all w 1-entries of $\mathbf{e}$ upfront to the first $n-k$ coordinates. Having weight w_i for the ith block, we permute an amount of vectors proportional to w_i upfront. Concretely, in Algorithm 2, we use the following *template-optimized permutation*.

Let P be a permutation matrix and $v_i \in \mathbb{Z}$ with $0 \le v_i \le b$ and $\sum_{i=1}^{t} v_i = n-k$. Further, denote the permuted error vector as

$$\mathbf{P}\,\mathbf{e} = (\mathbf{e}', \mathbf{e}'') = (\mathbf{e}'_1, \ldots, \mathbf{e}'_t, \mathbf{e}''_1, \ldots, \mathbf{e}''_t)$$

with $\mathbf{e}'_i \in \mathbb{F}_2^{v_i}$ and $\mathbf{e}''_i \in \mathbb{F}_2^{b-v_i}$. Then, $\mathbf{P}$ is a template permutation if $\mathbf{e}'_i$ and $\mathbf{e}''_i$ originate from $\mathbf{e}_i$ for all i. The success probability $\Pr(\sum_i |\mathbf{e}''_i| = 0)$ is determined by the v_i. In [23], a greedy algorithm for optimizing v_i is given. We observe that this optimal choice corresponds to the setting $v_i \approx \frac{w_i}{w} \cdot (n-k)$, i.e., the number of columns is chosen *proportional* to the weight of the block. This proportional assignment of columns generalizes the puncturing of [21]: columns of $\mathbf{H}$ with $w_i = 0$ are implicitly ignored by taking 0 columns from the blocks with $w_i = 0$. In Algorithm 2, the procedure that samples a template permutation as described above is called `TemplatePerm`.

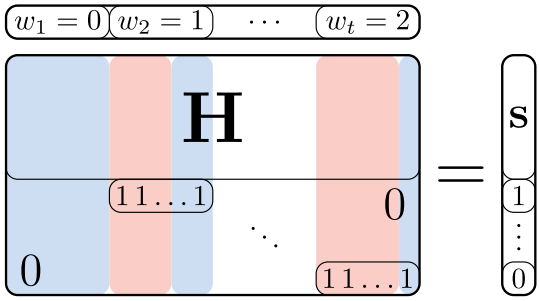

Fig. 1 Illustration of our improved template ISD method

In practice, $v_i = \frac{w_i}{w} \cdot (n-k)$ cannot be used directly due to rounding issues and the restriction $v_i \leq b$. In our implementation, we minimize $|v_i - \frac{w_i}{w} \cdot (n-k)|$.

Additional parity-check equations. Note that $|\mathbf{e}_i| = w_i$ implies that the sum of the entries of $\mathbf{e}_i$ is $w_i \mod 2$, see also [16]. Hence, for $w_i > 0$, one can extend the parity-check matrix by appending a row of the shape

$$(0, \ldots, 0, 1, \ldots, 1, 0, \ldots 0),$$

where the all-1 block is at the positions $[i \cdot b, (i+1) \cdot b)$. The syndrome $\mathbf{s}$ is extended by appending $w_i \mod 2$. Each appended check increases the co-dimension of the code by one to eventually $n - k + t - m_0$. This makes it easier for ISD to find a permutation that puts all weight to the first co-dimension many positions.

We summarize all modifications to the parity-check matrix and the optimized permutations in Fig. 1. Columns in blocks with error weight $w_i = 0$ are punctured. For $w_i \neq 0$, an additional check is appended to the parity-check matrix and the syndrome. Note that the parity of the block weight determines the corresponding syndrome bit. For each block, the number of columns chosen for permutation upfront (colored red) is set proportionally to the error weight.

Algorithm 2: Template Prange

Input : $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathbf{s}$, $\{w_i\}_{i \leq t}$; $\sum_i w_i = w$
Output: $\mathbf{e}$

Let $m_0 := |\{i \leq t \mid w_i = 0\}|$, $\bar{n} = n - m_0 b$, $\bar{k} = k + m_0 - t$.
Obtain $\bar{\mathbf{H}} \in \mathbb{F}_2^{(n-k)\times \bar{n}}$ by removing zero blocks.
Obtain $\bar{\mathbf{H}} \in \mathbb{F}_2^{(n-\bar{k})\times \bar{n}}$, $\bar{\mathbf{s}} \in \mathbb{F}_2^{n-\bar{k}}$ by adding checks.
repeat
    $\mathbf{P} \leftarrow$ `TemplatePerm`(w)
    Let $\mathbf{H}' = \mathbf{Q}^{-1}\bar{\mathbf{H}}\mathbf{P}$ be the systematic form of $\bar{\mathbf{H}}\mathbf{P}$
    $\mathbf{e}' = \mathbf{Q}^{-1}\bar{\mathbf{s}}$.
until $|\mathbf{e}'| = w$
return $\mathbf{P}^{-1} \cdot (\mathbf{e}', 0^k)$.

Table 1 Bit complexity estimates for ISD with and without template information for Classic McEliece parameters

Parameter set			Prange	Dumer	Template Prange	Template Dumer
n	k	w				
3488	2720	64	173	151	84	68
4608	3360	96	217	193	94	79
6960	5413	119	297	268	122	103
8192	6528	128	334	303	134	114

The columns which correspond to template ISD give the averaged runtime over 10,000 runs in case of random assignments for w_i's

Theorem 1 *Let $\{w_i\}_{i\le t}$ be a an error-free guess with m_0 many zeros. The expected number of permutations of Algorithm 2 for solving Template SD is*

$$\prod_{i=1}^{t}\binom{b}{w_i}\binom{\lfloor \frac{w_i}{w}(n-k+t-m_0)\rceil}{w_i}^{-1}.$$

Proof Our Algorithm 2 finds a good permutation if for all t blocks of length b, all w_i-many 1's from the ith block will be moved upfront to the first $n-\bar{k}=n-k+t-m_0$ coordinates. As from each block we take $\lfloor \frac{w_i}{w}(n-k+t-m_0)\rceil$ many positions, the expected number of required permutation follows. □

Example 2 (*Running example* $n = 2197$) According to [18], the concrete complexity of Algorithm 1 for $n = 2197$, $k = 1758$, $w = 37$ is estimated as 116 bits. Dimension reduction by weight-0 blocks reduces the complexity of this instance to 71 bits. With improved permutation and additional parity check equations from Algorithm 2, the complexity further decreases to 62 bits. Estimates for larger McEliece instances can be found in Table 1.

3.2 Enumeration-based template ISD—improving Dumer

Recall from Sect. 3.1 that for a parity-check matrix $\mathbf{H}\in\mathbb{F}_2^{(n-k)\times n}$ Prange's algorithm finds a permutation that shifts all w 1-entries of $\mathbf{e}$ upfront to the first $n-k$ entries. That is why Prange is called a permutation-based ISD.

Instead, enumeration-based ISD algorithms, such as [14, 19, 25], choose small parameters p, ℓ and permute $\mathbf{e}$ such that weight $w-2p$ lands on the first $n-k-\ell$ coordinates, and the remaining weight $2p$ lands on the last $k+\ell$ coordinates. On the one hand, such a permutation is way more likely to find the secret. On the other hand, we now have to enumerate a search space of size $\binom{k+\ell}{2p}$, in Dumer's algorithm in a meet-in-the-middle fashion. For the typical parameters used by McEliece-like cryptosystems, including Classic McEliece [2], such a tradeoff pays off, i.e., the benefit of faster finding a suitable permutation outweighs the cost of enumeration.

In this work, we chose to adapt Dumer's algorithm to the Template ISD setting. In the parameter range that we practically solve, Dumer's algorithm is known to perform best, whereas more advanced algorithms like [25] are taking over for n of cryptographic size [16].

Although we now choose Dumer as an enumeration-based ISD algorithm, the benefits from the Template ISD still contribute to a large extent to the search for a suitable permutation. Namely, analogous to Sect. 3.1, we obtain the template version of Dumer using the following modifications and improvements:

Algorithm 3: Template Dumer

Input : $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathbf{s}$, $\{w_i\}_{i\le t}$; $\sum_i w_i = w, b, p, \ell$
Output: $\mathbf{e}$

1 Let $m_0{:=}|\{i \le t \mid w_i = 0\}|$, $\bar{n} = n - m_0 \cdot b$, $\bar{k} = k + m_0 - t$, $k' = \bar{n} - (n - \bar{k}) + \ell$.
2 Obtain $\bar{\mathbf{H}} \in \mathbb{F}_2^{(n-k)\times \bar{n}}$ by removing zero blocks.
3 Obtain $\bar{\mathbf{H}} \in \mathbb{F}_2^{(n-\bar{k})\times \bar{n}}$, $\bar{\mathbf{s}} \in \mathbb{F}_2^{n-\bar{k}}$ by adding checks.
4 **repeat**
5 $P \leftarrow$ `TemplatePerm`$(w - 2p)$
6 Let $\mathbf{H}' = \mathbf{Q}^{-1}\bar{\mathbf{H}}\mathbf{P}$ be the quasi-systematic form of $\bar{H}P$, $(\mathbf{s}', \mathbf{s}'') = \mathbf{Q}^{-1}\bar{\mathbf{s}} \in \mathbb{F}_2^{n-\bar{k}-\ell} \times \mathbb{F}_2^{\ell}$.
7 **for** *all collisions* $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^{k'/2}$ *with weight* p **do**
8 Compute $\mathbf{e}'{:=}\mathbf{H}_1\mathbf{e}_1 + \mathbf{H}_2\mathbf{e}_2 + \mathbf{s}'$. ▷ via Meet-in-the-Middle
9 **until** $|\mathbf{e}'| = w - 2p$
10 **return** $\mathbf{P}^{-1} \cdot (\mathbf{e}', \mathbf{e}_1, \mathbf{e}_2)$.

Dimension reduction: 0-weight blocks from the guess $\{w_i\}_{i\le t}$ are removed, let m_0 be their number. Such a dimension reduction helps to significantly speed up permutation search, and it also decreases the search space for enumeration by $m_0 b$.

Additional parity checks: Encoding all $w_i \ge 1$ into additional check equations increases the co-dimension from $n - k$ to $n - k + t - m_0$. This speeds up permutation search further and slightly reduces the enumeration cost.

Improved permutation: Similar to Section 3.1, we permute upfront proportionally to the weights w_i to improve the permutation. For this, we set $v_i \approx \frac{w_i \cdot c}{w-2p}(n - \bar{k} - \ell)$, where $c = \frac{w-2p}{w}$ is a re-scaling factor. We do not exploit non-zero weights for enumeration.

The resulting algorithm Template Dumer is given in Algorithm 3.

Theorem 2 (Template Dumer) *Let $k'{:=}\bar{n} - (n - \bar{k}) + \ell$ with $\bar{n}$ and $\bar{k}$ as in Algorithm 3. Then, the number of iterations that Template Dumer ISD performs on average is the inverse of the success probability*

$$\binom{k'/2}{p}^2 \binom{k'}{2p}^{-1} \sum_{p_1+\cdots+p_t=2p} \prod_{i=1}^{t} \binom{\lfloor \frac{w_i}{w}(n-\bar{k}-\ell)\rceil}{w_i - p_i} \binom{b}{w_i}^{-1}, \tag{1}$$

where each iteration has a meet-in-the-middle cost of $2\binom{k'/2}{p} + \binom{k'/2}{p}^2 \cdot 2^{-\ell}$.

Proof Since $\lfloor \frac{w_i}{w}(n - \bar{k} - \ell)\rceil$ positions of the ith block are moved upfront, this contributes p_i errors to the last k' positions with probability $\binom{\lfloor \frac{w_i}{w}(n-\bar{k}-\ell)\rceil}{w_i - p_i}\binom{b}{w_i}^{-1}$. Similar to [23], the probability of $2p$ errors in the last k' positions is obtained by summing over all possibilities $p_1 + \ldots + p_t = 2p$. Further, the error needs to split evenly in the last k' positions. Randomizing the order of these coordinates, this probability is $\binom{k'/2}{p}^2\binom{k'}{2p}^{-1}$. The meet-in-the-middle step requires enumerating $2\binom{k'/2}{p}$ vectors $\mathbf{e}_1, \mathbf{e}_2$, leading to $\binom{k'/2}{p}^2 2^{-\ell}$ collisions on average. □

Example 3 (*Running example $n = 2197$*) For $n = 2197$, $k = 1758$, $w = 37$, we pick $\ell = 16$ and $p = 2$ for Algorithm 3. The performance differs between guesses. On average, 2^{20}

iterations are sufficient, each with a Meet-in-the-Middle cost of processing approximately 2^{16} vectors. Taking into account the cost of partial Gaussian elimination, we obtain an estimated cost of 2^{47} binary operations. Estimates for larger McEliece instances can be found in Table 1.

4 Dealing with Noisy Guesses

In practice, Hamming weights are prone to misclassification due to various noise sources, ranging from the thermal agitation of charge carriers to the unwanted power-consumption contributions from untargeted system components. Hence, the solver should tolerate misclassifications to some extent to be practically viable. Such robustness is not directly available from Algorithm 2, which, while extending the matrix $\mathbf{H}$, assigns wrong syndrome entries for blocks with incorrectly estimated weights. An erroneous syndrome causes the algorithm to fail in finding the error vector. As was observed earlier, the additional check rows do, however, significantly impact the solver's performance. Hence, it is undesirable to discard all of them right away.

This section demonstrates how noisy measurement information can be efficiently integrated into template ISD algorithms. The proposed techniques are compatible with Template Prange and Template Dumer, as well as other template solvers.

Inherent redundancy.

Let us assume that template information is available for all blocks, but a single one. Since the overall weight of the error vector is known, one can recover the error weight of the missing block as $w_j = w - \sum_{i \neq j} w_i$.

This inherent redundancy allows for a simple method for coping with small misestimations. Given a full measurement template vector $\hat{\mathbf{w}}$, one can use the knowledge of the overall Hamming weight of the error vector to check consistency. In the low-noise setting, $\sum_i \hat{w}_i = w$ implies $\hat{\mathbf{w}} = \mathbf{w}$ with high probability. In case of $\sum_i \hat{w}_i \neq w$, $\hat{\mathbf{w}}$ and $\mathbf{w}$ usually differ in a single block. For $\sum_i \hat{w}_i = w - 1$, one can find the error vector using at most t trials. In each trial, the estimated weight of one block is increased. This modified sequence of block weights $\hat{\mathbf{w}}$ is used as input for the Template ISD algorithm. Since an incorrect guess leads to wrong entries in the syndrome w.r.t. the extended parity-check matrix, the trial will only terminate with the correct error vector $\mathbf{e}$ if the assumed input sequence is correct. Distinguishing whether the input matches the actual weights is possible only by performing a sufficient number of iterations. Let $T_{\texttt{TISD}}(\hat{\mathbf{w}})$ denote the expected number of iterations required by the solver to recover $\mathbf{e}$ given the modified input. Then, Markov's inequality implies that in each trial, it is sufficient to perform $T_{\texttt{TISD}}(\hat{\mathbf{w}})$ iterations to find the error vector with probability $1/2$. Incorrect trials are terminated upon reaching this maximum number of iterations.

The same procedure applies for $\sum_i \hat{w}_i = w + 1$ and can be generalized to arbitrary differences between $\mathbf{w}$ and $\hat{\mathbf{w}}$. This is efficient in the low-noise setting, where few combinations result in correct block weight estimates. However, as the noise increases, the number of relevant test patterns grows exponentially. In fact, at some point, the combined cost of trying ISD for different combinations exceeds the complexity of simply running ISD on the original SDP instance. In the following, we provide a simple but general method for dealing with noisy templates that smoothly interpolates between perfect side-channel information and the plain classical SDP.

Noise model.

To evaluate the noise tolerance of our solver in a reproducible manner, we establish a simple, single-parameter noise model. A measured sample y_i is modeled as the sum of the

block weight w_i and independent Gaussian noise n_i, where standard deviation σ is the only parameter:

$$y_i = w_i + n_i, \text{ with } n_i \sim \mathcal{N}(0, \sigma)\ \forall i.$$

Hence, the density of the measurement y_i conditioned on a given weight w_i is given by

$$f(y_i \mid w_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y_i - w_i)^2/(2\sigma^2)}.$$

This model relies on three fairly common assumptions:

1. The noiseless measured value is linearly correlated with the weight w_i. This is a central assumption of Correlation Power Analysis (CPA) [7].
2. Independent Gaussian noise [9, 21] is suitable because noise phenomena on the transistor level, such as Johnson-Nyquist noise, are inherently continuous, thereby transforming the discrete variable w_i into a continuous variable y_i.
3. For simplicity, the measurement is modeled by a single variable or sample y_i, whereas in practice, a side-channel trace usually contains multiple samples correlated to w_i, and information obtained from these samples is aggregated.

Using Bayes' rule, one can compute

$$\Pr(w_i = \ell \mid y_i) = \frac{f(y_i \mid \ell) \cdot \Pr(w_i = \ell)}{\sum_x f(y_i \mid x) \cdot \Pr(w_i = x)},$$

where the prior can be calculated as $\Pr(w_i = \ell) = \binom{b}{\ell}\binom{n-b}{w-\ell}/\binom{n}{w}$. Given $\Pr(w_i \mid y_i)$, the maximum-a-posteriori (MAP) estimate for the error weight in the ith block is given by

$$\hat{w}_i = \arg\max_{\ell} \Pr(w_i = \ell \mid y_i).$$

Example 4 (*Running Example* $n = 2197$) Figure 2 illustrates the probability density functions for the assumed model and the resulting optimal decision boundaries. We consider our running example with $n = 2197$, $w = 37$, $b = 32$ and assume a template noise level $\sigma = 0.3$.

From Fig. 2, it can be observed that the maximum-a-posteriori (MAP) decision boundaries, which are optimal in the sense that they minimize the error rate, do not lie precisely in the middle between the w_i. This is due to the inclusion of the prior $\Pr(w_i)$, which decreases in w_i. Let $\bar{w}_\ell$ denote the MAP-decision boundary between weight ℓ and $\ell + 1$. Then, $\bar{w}_\ell$ can be computed as

$$\bar{w}_\ell = \ell + \tfrac{1}{2} + \sigma^2 \cdot \ln\left(\frac{\Pr(w_i = \ell)}{\Pr(w_i = \ell + 1)}\right) \quad \forall \ell \in \{0, \ldots, b-1\}.$$

Further, we define $\bar{w}_{-1} = -\infty$ and $\bar{w}_b = \infty$. Then, the error rate is computed as

$$\Pr(w_i \neq \hat{w}_i) = \sum_{\ell=0}^{b} \Pr(w_i = \ell) \cdot \left(Q\left(\frac{\bar{w}_\ell - \ell}{\sigma}\right) + Q\left(\frac{\ell - \bar{w}_{\ell-1}}{\sigma}\right)\right),$$

where the Q-function $Q(x)$ denotes the probability of a standard normal random variable being larger than x.

Example 5 (*Running Example* $n = 2197$) For $n = 2197$, $w = 37$, $b = 32$ and a noise level of $\sigma = 0.3$, the error rate can be computed as $\Pr(w_i \neq \hat{w}_i) = 0.059$. Figure 2 offers a visual interpretation of this value; $\Pr(w_i \neq \hat{w}_i)$ is equal to the area of the surface marked in red,

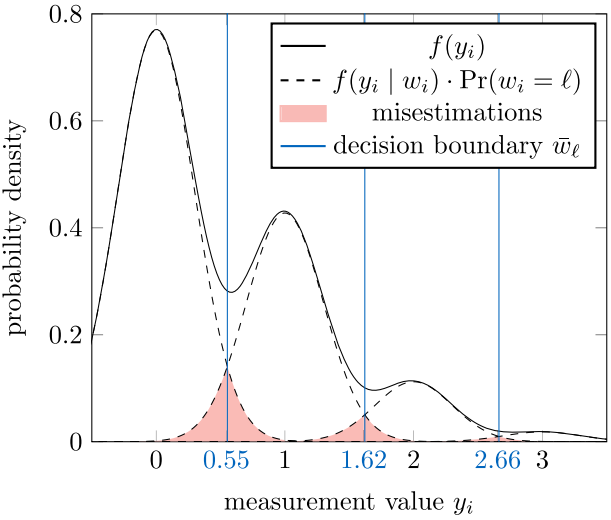

Fig. 2 Illustration of the probability densities and the decision boundaries for template noise level $\sigma = 0.3$. The curves were generated for the running example with parameters $n = 2197$ and $w = 37$ with a block size of $b = 32$

which corresponds to values of y_i that lead to misestimations. While $\Pr(w_i \neq \hat{w}_i) = 0.059$ may seem small at first glance, it still implies that a guess of length 69 is expected to contain around 4 wrong MAP estimates. Note that this is higher than the error rate of 0.03 observed in our practical experiments; see Sect. 5. The error rate of 0.03 corresponds to a noise level of roughly $\sigma = 0.25$.

Note that the MAP estimates are hard-decision estimates in the sense that $\hat{w}_i \in [0, b]$. Hence, even under the optimal decision rule, they do not include the complete information provided by the template measurement. This is illustrated by the fact that the probability that the resulting hard-decision estimation is correct differs depending on how close y_i is to a decision boundary; measurement values that lie closer to the decision boundaries hold less information than those further away from the border. We quantify this reliability information included in y_i as $r_i := \Pr(w_i = \hat{w}_i \mid y_i)$, i.e., as the probability that the estimate is correct. In the following, we provide a preprocessing method that allows the inclusion of this reliability information in Template ISD.

Preprocessing for reliable ISD with noisy templates.

The proposed assignment method is described in Algorithm 4. The basic idea is to generate a guess suitable for ISD from a measurement vector by distinguishing between reliable and unreliable blocks.

One begins by calculating the reliability of each block. Then, one determines the permutation π, which sorts these reliabilities in decreasing order. The t blocks of the measurement vector $\mathbf{y}$ are processed according to this order. For the first, most reliable blocks, one performs a MAP estimation of the block weight. Using the blockwise MAP estimate as an entry of the guess allows assigning an additional parity check to the corresponding block. This is continued as long as the product of the reliabilities of the blocks exceeds a threshold θ. Denote as $\mathcal{J}$ the set of most reliable blocks that are assigned the MAP estimate by Algorithm

Algorithm 4: Preprocessing of Noisy Templates

Input : $\mathbf{H}, \mathbf{s}, w, \mathbf{y}, \theta$
Output: $\mathbf{e}$

$r_i \leftarrow \max_\ell \Pr(w_i = \ell \mid y_i)\ \forall i$ ▷ reliability of MAP decisions
$\pi \leftarrow \text{argsort}(\mathbf{r})$ ▷ sort in decreasing order of reliability
$P_{\text{succ}} \leftarrow 1$
for $j = 1, \ldots, t$ **do**
  $i \leftarrow \pi(j)$ ▷ get j-th most reliable block
  if $P_{\text{succ}} \cdot r_i \geq \theta$ **then**
    $P_{\text{succ}} \leftarrow P_{\text{succ}} \cdot r_i$
    $\hat{w}_i \leftarrow \arg\max_\ell \Pr(w_i = \ell \mid y_i)$ ▷ MAP estimates for reliable blocks
    if $\hat{w}_i \neq 0$ **then**
      add check to H in i-th block, extend $\mathbf{s}$
  else
    $\hat{w}_i = \mathbb{E}[w_i \mid y_i]$ ▷ expected weight for unreliable blocks
add check to $\mathbf{H}$ in unreliable blocks, extend $\mathbf{s}$
$\mathbf{e} \leftarrow$ `TemplateISD`$(\mathbf{H}, \mathbf{s}, \hat{\mathbf{w}})$ ▷ Prange or Dumer
return $\mathbf{e}$.

4. Then,

$$\theta \leq \prod_{j \in \mathcal{J}} r_j = \prod_{j \in \mathcal{J}} \Pr(\hat{w}_j = w_j \mid y_j) \approx \Pr\left(\hat{w}_j = w_j\ \forall j \in \mathcal{J} \mid y_j\ \forall j \in \mathcal{J}\right),$$

where the approximation ignores the weak dependence of the misestimation events in the blocks ordered w.r.t. $\mathbf{y}$. Hence, one can expect all blocks with checks to be correct with a probability of approximately θ. In our experiments, we typically pick $\theta = 0.75$ and observed that the approximation is indeed correct for the medium noise regime (note that in the low and high noise regime, either all or no blocks are picked, so the concrete choice of θ has little impact on the success probability).

The available information is insufficient to assign hard-decision estimates for the remaining less reliable blocks. Instead, soft-decision estimates are used, which indicate the range of the error weight while not necessarily lying in $[0, b]$, unlike their hard-decision counterparts. A suitable choice is the expected weight given y_i as soft-decision estimate, which can be computed as $\mathbb{E}[w_i \mid y_i] = \sum_\ell \ell \cdot \Pr(w_i = \ell \mid y_i)$. For example, $\hat{w}_i = 0.7$ implies that the error weight is 0 or 1, with 1 being more likely. This information can then later be used to distribute the information sets.

On the other hand, the soft-decision estimates do not allow the assignment of parity checks to the individual blocks. One can, however, assign a length-n check over the entire width of the parity-check matrix since the overall error weight w is known [15]. In the template setting, this is equivalent to assigning a single check to all unreliable blocks.

Finally, a Template ISD Algorithm uses the guess $\hat{\mathbf{w}}$ and the suitably extended parity-check matrix $\mathbf{H}$ to recover the error vector. To this end, either Template Prange or Template Dumer can be used. In comparison to the case of a perfect template, only a minor modification is required: For the generated guess $\hat{\mathbf{w}}$, $\sum_{i=1}^{t} \hat{w}_i = w$ is not guaranteed. To consider this, we normalize by $\sum_{i=1}^{t} \hat{w}_i$ instead of w when determining the number of elements per block permuted upfront.

Figure 3 illustrates the performance of Algorithm 4 in combination with Template Prange

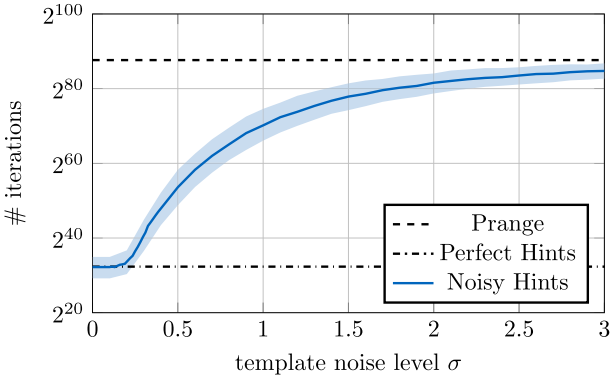

Fig. 3 The cost of solving instances with $n = 2197$, $k = 1758$, $w = 32$, and $b = 32$ using Template Prange with the proposed preprocessing step. The 0.25-quantile, median, and 0.75-quantile of the expected number of iterations for different noise levels σ are depicted. The threshold θ was chosen as $\theta = 0.75$, for which we observe that the fraction of samples for which the solver terminates successfully exceeds $1/2$. The practical experiments described in Sect. 5 achieve approximately a noise level of $\sigma = 0.25$, i.e., fall into the regime in which Algorithm 4 allows for optimal performance despite the measurement noise

for different noise regimes. While other solvers are also applicable, Template Prange highlights the impact of the developed methods most effectively: Algorithms with enumeration, such as Template Dumer, are inherently more resilient to noise.

We consider the parameters of our running example, i.e., $n = 2197$, $k = 1758$, and $w = 37$ for blocksize $b = 32$. The 0.25-quantile, median, and 0.75-quantile of the expected number of Prange iterations are plotted for different noise levels and compared with the cases of perfect and no side-channel information. It can be observed that the attack is very robust against measurement noise with $\sigma \leq 0.3$. The solver's performance in this regime is virtually identical to Template Prange with perfect side-channel information. This is due to the use of reliability information, which allows for identifying potential measurement errors. Further, the inherent redundancy, i.e., the knowledge of the overall error weight w, compensates for individual measurement errors. Note that the practical experiments that we have performed (see Sect. 5) fall into this region. Starting at $\sigma = 0.3$, the required number of iterations increases quickly. In this regime, the number of misestimations is considerable and exceeds the correction capability of the inherent redundancy. For $\sigma = 0.5$, a MAP estimate for the weight of a 32-bit block is incorrect with probability 0.19. This is considerably higher than the error rate of 0.059 at $\sigma = 0.3$. Despite this drastic increase in wrong estimates, using soft-decision weights for unreliable blocks allows for improvement over plain Prange. For $\sigma \to \infty$, the side-channel measurement $\mathbf{y}$ contains no information on $\mathbf{w}$. In this setting, the performance of the templated solver converges to the performance of plain ISD. Hence, using Algorithm 4, Template ISD can interpolate smoothly between the case of no information and perfect information.

5 Side-channel experiments

5.1 Measurement setup

We target an open-source C implementation of McEliece, which is made by Chen and Chou [11], optimized for the ARM Cortex-M4, and unprotected against side-channel attacks. The targeted function is a Hamming-weight computation in the decryption, as specified in Listing 1 [10]. To accelerate our measurements, we do not run the entire decapsulation, and instead communicate via UART with a custom wrapper around function weight_3488. Likewise, although solving $n = 3488$ is computationally feasible, we reduce n to 2197 for faster results. The code is compiled by arm-none-eabi-gcc using O3 optimization. Although the right-shift of the 32-bit word $\mathbf{v}$ in Listing 1 might leak bit-level information, we only aim to recover word-level information, i.e., weights $|\mathbf{v}|$.

```
1 static int weight_3488(uint32_t *v)
2 {
3   int i, w = 0;
4   for (i = 0; i < 3488; i++)
5     w += (v[i>>5] >> (i&31)) & 1;
6   return w;
7 }
```

Listing 1 Targeted C function [10].

The power consumption is measured using ChipWhisperer equipment: a CW308 UFO board, an STM32F405RGT6 target that contains an ARM Cortex-M4, and a Husky oscilloscope. The clock frequency is set to 16MHz and the sampling frequency is set to 128MHz, i.e., there are 8 samples per clock period. To synchronize traces, the wrapper raises a trigger signal right before function weight_3488 is executed. To capture the entire operation, around $202k$ samples suffice. As the Husky has a buffer of around $131k$ samples, we stitch together 2 traces by varying the offset from the trigger. Traces for template building and template matching are taken from the same STM chip, which is fair: to build templates, the attacker can perform unlimited encapsulations to obtain known pairs $(\mathbf{c}, \mathbf{e})$.

5.2 Template building

Given that error $\mathbf{e}$ spans 69 words, each having weight $W \in \{0, 1, 2, 3\}$ with overwhelming probability, $276 = 69 \times 4$ templates are built. For this purpose, we randomly generate $48k$ error vectors $\mathbf{e}$ and measure one trace for each $\mathbf{e}$. To ensure that the templates have similar qualities, we impose $\Pr(W = 0) = \Pr(W = 1) = \Pr(W = 2) = \Pr(W = 3) = 1/4$. This deviation from the McEliece distribution is optional and is only realistic for a 2-device attack. We have not investigated whether the original distribution is better or worse in terms of prediction accuracy: the larger W, the lower the quality of the template, but also the lower the importance of the template. All choose-W-out-of-32 selections are equally likely. For example, words 0x80020040 and 0x01400002 are equally likely in the case of $W = 3$.

For each out of 69 words, we only retain the 100 samples that matter most. All other samples primarily generate classification noise, so it's beneficial to discard them. To make a selection, we extend Welch's t-test [29] from two to four populations as specified below, where M_W is the *sample mean*, V_W is the *sample variance*, and N_W is the number of traces for

each weight W. The samples with the largest absolute values of *test statistic* T are retained.

$$T = \frac{1}{3}\left(\frac{M_0 - M_1}{\sqrt{\frac{V_0}{N_0} + \frac{V_1}{N_1}}} + \frac{M_1 - M_2}{\sqrt{\frac{V_1}{N_1} + \frac{V_2}{N_2}}} + \frac{M_2 - M_3}{\sqrt{\frac{V_2}{N_2} + \frac{V_3}{N_3}}}\right)$$

5.3 Template matching

For each error $\mathbf{e}$ we aim to recover, we collect $12k$ traces and average them into a single trace X. Now, the weights W are non-uniform and follow the McEliece distribution. For each out of 69 words, the distinguisher $D_W = \sum_{i=0}^{99} |T_i| \cdot |M_W - X_i| \in \mathbb{R}^+$ is computed. The weight W for which D_W is the smallest is the best guess. The probability that this guess is correct is around 97 %, corresponding to a noise level of $\sigma = 0.25$ from the perspective of Sect. 4.

6 Practical template SD solving with our algorithms

We implemented Algorithms 2 and 3. The source code of our implementation can be found in [6]. We ran our experiments on the parity-check matrices of Classic McEliece instances with parameters provided by [3], where we generated the solution vectors $\mathbf{e}$ ourselves. We fully recovered the secret error vector for all instances $n \leq 2197$.

In the experiments, we always worked with an error-free guess. Indeed, the actual side-channel attacks gave guesses with an accuracy of 97 % corresponding to a noise level of $\sigma = 0.25$. For our running example $n = 2197$, the measurement resulted in a single mispredicted block: we observed a guess $\hat{\mathbf{w}}$ with $\sum_i \hat{w}_i = w - 1$, which can be corrected with low overhead due to the inherent redundancy. For longer lengths, an accuracy of 97 % implies that more than a single misestimation is to be expected. Indeed, for $n = 3488$, we have observed two transitions from $w_i = 2$ to $\hat{w}_i = 1$ and a transition $w_i = 3$ to $\hat{w}_i = 2$. Nevertheless, Algorithm 4 enables us to achieve a virtually identical performance to the noiseless case, see Fig. 3. Note, however, that our target implementation was unprotected, whereas countermeasures against side-channel attacks [12, 20] are bound to decrease the accuracy.

To accurately estimate the running time of Algorithms 1 (Original Prange), 2 (Template Prange), and 3 (Template Dumer) for all dimensions n, we generated random error vectors and measured the runtime *per permutation*. To obtain the overall runtime, we multiply by the expected number of permutations, which is computed as in Theorems 1,2 by averaging over different error vectors. The resulting estimates are presented in Fig. 4.

Example 6 (*Running Example* $n = 2197$) Our implementation of Algorithm 3 (with $p = 2$, $\ell = 16$) on 2x AMD EPYC 7742 CPUs recovered the secret $\mathbf{e}$ of an instance with $n = 2197$ in 10^3 s using $1.1 \cdot 10^6$ iterations (the predicted number of iterations for this instance is 1.6×10^6). The implementation is parallelized over the choice of permutation, and with 200 threads outputs the secret in 10 s using only 334MB of RAM.

Acknowledgements Sebastian Bitzer and Antonia Wachter-Zeh acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the program of "Souverän. Digital. Vernetzt.". Joint project 6 G-life, project identification number: 16KISK002.

Funding Open Access funding enabled and organized by Projekt DEAL.

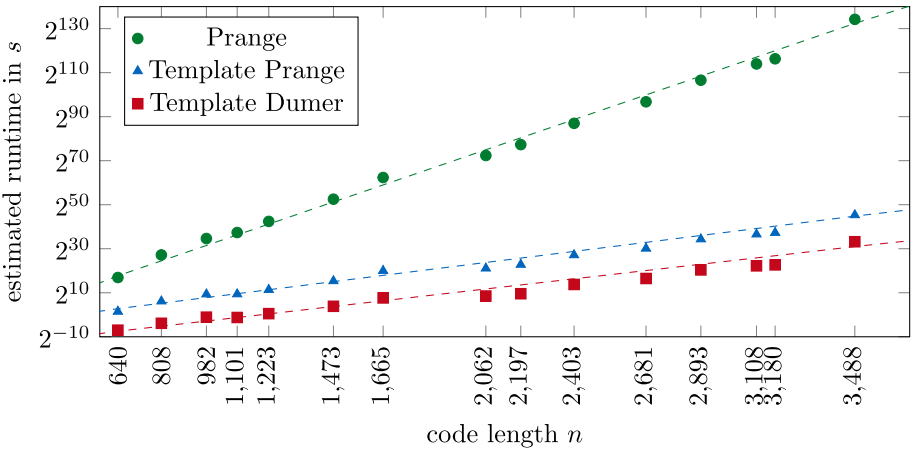

Fig. 4 Single-threaded performance of our implementation on AMD EPYC 7742

References

1. Al Jabri A.: A statistical decoding algorithm for general linear block codes. In: Honary B. (ed.) 8th IMA International Conference on Cryptography and Coding, vol. 2260, pp. 1–8. LNCS. Springer, Hedeilberg (2001).
2. Albrecht M.R., Bernstein D.J., Chou T., Cid C., Gilcher J., Lange T., Maram V., von Maurich I., Misoczki R., Niederhagen R., Paterson K.G., Persichetti E., Peters C., Schwabe P., Sendrier N., Szefer J., Tjhai C.J., Tomlinson M., Wang W.: Classic McEliece: conservative code-based cryptography (2020). https://classic.mceliece.org/nist/mceliece-20201010.pdf.
3. Aragon N., Lavauzelle J., Lequesne M.: decodingchallenge.org (2019). http://decodingchallenge.org.
4. Aragon, N., Barreto, P.L., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Ghosh, S., Gueron, S., Guneysu, T., Melchor, C.A., Misoczki, R., Persichetti, E., Richter-Brockmann, J., Sendrier, N., Tillich, J.-P., Valentin, V., Zémor, G.: BIKE — Bit Flipping Key Encapsulation (2023). https://bikesuite.org/.
5. Augot D., Finiasz M., Sendrier N.: A family of fast syndrome based cryptographic hash functions. In: Progress in Cryptology—Mycrypt 2005, pp. 64–83 (2005).
6. Bitzer S., Delvaux J., Kirshanova E., Maaben S., May A., Bitzer A.W.-Z.S., Delvaux J., Kirshanova E., Maaben, S., May A., Wachter-Zeh A.: Practical Template Syndrome Decoding Attack. GitHub repository. https://github.com/ChuTriel/TemplateISD (2024).
7. Brier E., Clavier C., Olivier F.: Correlation power analysis with a leakage model. In: 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), pp. 16–29. Springer (2004).
8. Carrier K., Debris-Alazard T., Meyer-Hilfiger C., Tillich J.-P.: Statistical decoding 2.0: Reducing decoding to LPN. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 477–507. Springer, Heidelberg (2022).https://doi.org/10.1007/978-3-031-22972-5_17.
9. Chari S., Rao J.R., Rohatgi P.: Template attacks. In: Kaliski B.S. Jr., Koç Ç.K. Paar C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003).https://doi.org/10.1007/3-540-36400-5_3
10. Chen M.-S., Chou T.: Classic McEliece implementation for ARM-Cortex M4 (2022). https://github.com/pqcryptotw/mceliece-arm-m4/blob/main/pqm4-projects/crypto_kem/mceliece348864/ches2021/decrypt_n3488_t64.c. commit f2a699dd480f9f91d566eb4b910fd4e51e3bdc91.

11. Chen M.-S., Chou T.: Classic McEliece on the ARM cortex-M4. IACR TCHES **2021**(3), 125–148 (2021). https://doi.org/10.46586/tches.v2021.i3.125-148 . https://tches.iacr.org/index.php/TCHES/article/view/8970.
12. Chen C., Eisenbarth T., Von Maurich I., Steinwandt R.: Masking large keys in hardware: a masked implementation of McEliece. In: Dunkelman O., Keliher L. (eds.) Selected Areas in Cryptography—SAC 2015, pp. 293–309. Springer, Cham (2016).
13. Ducas L., Esser A., Etinski S., Kirshanova E.: Asymptotics and improvements of sieving for codes. In: Advances in Cryptology—EUROCRYPT 2024, pp. 151–180. Springer, Berlin (2024).
14. Dumer I.: On minimum distance decoding of linear codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory, pp. 50–52, Moscow (1991).
15. Esser A., May A., Zweydinger F.: McEliece needs a break—solving McEliece-1284 and quasi-cyclic-2918 with modern ISD. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 433–457. Springer (2022).
16. Esser A., May A., Zweydinger F.: McEliece needs a break—solving McEliece-1284 and quasi-cyclic-2918 with modern ISD. In: Dunkelman O., Dziembowski,S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 433–457. Springer, Heidelberg (2022).https://doi.org/10.1007/978-3-031-07082-2_16.
17. Esser A., Santini P.: Not just regular decoding: asymptotics and improvements of regular syndrome decoding attacks. In: Advances in Cryptology—CRYPTO 2024, pp. 183–217. Springer, Berlin (2024)
18. Esser A., Verbel J., Zweydinger F., Bellini E.: CryptographicEstimators—a software library for cryptographic hardness estimation. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, pp. 560–574 (2024).
19. Finiasz M., Sendrier N.: Security bounds for the design of code-based cryptosystems. In: Advances in Cryptology—ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, 6–10 December 2009. Proceedings 15, pp. 88–105. Springer, Berlin (2009).
20. Gan P., Ravi P., Raj K., Baksi A., Chattopadhyay A.: Classic McEliece hardware implementation with enhanced side-channel and fault resistance. Cryptology ePrint Archive (2024).
21. Grosso V., Cayrel P.-L., Colombier B., Drăgoi V.-F.: Punctured syndrome decoding problem: Efficient side-channel attacks against classic McEliece. In: International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 170–192. Springer, Cham (2023).
22. Guo Q., Johansson T., Nguyen V.: A new sieving-style information-set decoding algorithm. IEEE Trans. Inf. Theory **70**(11), 8303–8319 (2024).
23. Horlemann A.-L., Puchinger S., Renner J., Schamberger T., Wachter-Zeh A.: Information-Set Decoding with Hints. In: Code-Based Cryptography, pp. 60–83. Springer, Cham (2022).
24. Lahr N., Niederhagen R., Petri R., Samardjiska S.: Side channel information set decoding using iterative chunking - plaintext recovery from the "classic McEliece" hardware reference implementation. In: Moriai S., Wang H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 881–910. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-64837-4_29.
25. May A., Meurer A., Thomae E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Lee D.H., Wang X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_6.
26. Melchor C.A., Aragon N., Bettaieb S., Bidoux L., Blazy O., Bos J., Deneuville J.-C., Dion A., Gaborit P., Lacan J., Persichetti E., Robert J.-M., Véron P., Zémor G.: Hamming quasi-cyclic (HQC) (2023). https://pqc-hqc.org/.
27. Narisada S., Fukushima K., Kiyomoto S.: Multiparallel MMT: faster ISD algorithm solving high-dimensional syndrome decoding problem. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **106**(3), 241–252 (2023).
28. Prange E.: The use of information sets in decoding cyclic codes. IRE Trans. Inf. Theory **8**(5), 5–9 (1962).
29. Standaert F.: How (not) to use Welch's t-test in side-channel security evaluations. In: Bilgin B., Fischer J. (eds.) 17th Conference on Smart Card Research and Advanced Applications (CARDIS 2018), vol. 11389, pp. 65–79. Lecture Notes in Computer Science. Springer, Cham (2018).
30. Stern J.: A method for finding codewords of small weight. In: Coding Theory and Applications, pp. 106–113. Springer, New York (1989).