Meet-in-the-Middle Attacks on Full ChiLow

Eran Lambooij[1(✉)], Patrick Neumann[1(✉)], Michiel Verbauwhede[2(✉)], Shichang Wang[3(✉)], and Tianyu Zhang[3(✉)]

[1] Inria, Paris, France
{eran.lambooij,patrick.neumann}@inria.fr
[2] COSIC, KU Leuven, Leuven, Belgium
michiel.verbauwhede@esat.kuleuven.be
[3] Nanyang Technological University, Singapore
{shichang.wang@ntu.edu.sg,tianyu005@e.ntu.edu.sg}

Abstract. This work presents the first *full-round* attacks on ChiLow-32 and ChiLow-40, two tweakable low-latency block ciphers presented at Eurocrypt 2025.
We first describe a straightforward Meet-in-the-Middle attack on full ChiLow-32 with multiple known plaintext-ciphertext pairs. To improve this attack, we carefully reduce the number of guesses required by (1) tracing *differences* in order to remove linear key dependencies and (2) moving from *key* guesses to *state* guesses. Using a novel method that is based on the propagation of differences and linear masks, we are able to map out the state dependencies for computing the difference at the matching point. This results in an attack on ChiLow-32 with time complexity $2^{120.34}$ using 160 known plaintext-ciphertext pairs, and an attack with time complexity $2^{102.09}$ using 64 chosen ciphertexts.
Using these techniques, and an additional trick to better balance the complexities of the meet-in-the-middle branches, we propose an attack on ChiLow-40 with time complexity $2^{122.32}$ and 2^8 chosen plaintexts. All of our attacks are within ChiLow's security model, and are currently the best and only known key recovery attacks on full-round ChiLow-32 and ChiLow-40.

Keywords: Meet-in-the-Middle · Key-Recovery Attack · Symmetric Cryptanalysis · ChiLow· Key Dependencies · State Dependencies

1 Introduction

The past decade has seen a growing demand in lightweight cryptographic algorithms, driven by the proliferation of resource-constrained devices in the Internet of Things (IoT) and embedded systems. In 2018, NIST initiated a standardization process for lightweight cryptographic algorithms, and announced Ascon [17] as the general-purpose standard. However, such a general-purpose cipher must find a good balance between different cost metrics, such as area, energy consumption, and latency, even though the importance of each of these metrics depends on the specific use case. Consequently, use cases with more extreme constraints

can benefit from, or even require, special-purpose designs. This has led to the development of context-specific algorithms tailored to a specific use case. Two notable examples are Bipbip [6] and SCARF [10], which were recently shown to be insecure at FSE 2025 [29], and Eurocrypt 2025 [19] respectively. These results highlight the difficulty of balancing extreme efficiency requirements with robust security.

ChiLow [7] is a tweakable block cipher, proposed at Eurocrypt 2025, and designed for *embedded code encryption*. In this setting, instructions are stored encrypted on an unprotected device and need to be decrypted *on-the-fly* before execution. Hence, the decryption latency of such a cipher needs to be extremely low, leading to strongly optimized ciphers with small security margins. To achieve this, the designers propose the use of a novel non-linear layer χ, which is later generalized in [1]. Together with a lightweight linear layer and the addition of a round (tweak-)key, this forms one round of ChiLow. Based on preliminary cryptanalysis, the designers conclude that 8 such rounds should lead to a secure cipher. However, combined with the small block size of 32 or 40 bits, such a low number of rounds points towards a potential vulnerability to meet-in-the-middle (MitM) attacks recovering the individual round (tweak-)keys. The security model of ChiLow restricts the number of plaintext-ciphertext pairs per tweak to 2^8 due to the specific use case, and while using a different tweak means changing the round (tweak-)key, we show that full-round MitM attacks are still feasible under this restriction. The complexities of our best attacks, together with a comparison to prior work, are given in Table 1.

Related Work The concept of MitM attacks is one of the most influential cryptanalytic techniques. It dates back to Diffie and Hellman's attack on Double-DES [16], in which they partition the primitive into two separate sub-ciphers, each explored independently. This concept has since been applied to various key recovery attacks, for instance on KTANTAN [8,30], extended to preimage attacks [27], and to collision attacks [24]. In addition, a wide range of generic improvements, such as splice-and-cut [4], initial structures [3,28], indirect and partial matching [2], probabilistic techniques [20], bicliques [21], precomputations [12], superposition states [5], distributed initial structures [13] and single-color initial structures [14] have been proposed.

For ChiLow, there are already several notable works despite its recency: Peng *et al.* give the first independent security analysis [26], providing key recovery attacks for up to 7 (out of 8) rounds of ChiLow-32. In addition, Andreoli *et al.* [1] generalize and study χ in greater detail with respect to its cryptographic properties.

Contribution & Technical Overview In this work, we present the first full-round attacks on the ChiLow family of tweakable block ciphers. We significantly improve upon the previous best attacks by the designers of ChiLow [7], and by

Peng *et al.* [26], see Table 1. The core of our attacks has been experimentally verified, and the source code is provided as supplementary material [22].[4]

First, we describe the ChiLow cipher family and summarize the designers' security claims in Section 2. In Section 3, we first present a basic meet-in-the-middle attack on full-round ChiLow-32 by partially guessing the round keys while matching on one bit per known plaintext-ciphertext pair. We then discuss further improvements in Section 4. More specifically, we first move from tracing *values* through the cipher to tracing *differences* in an effort to remove linear key dependencies. In addition, instead of guessing the *key material* that is required to propagate the input difference to the matching point, we instead guess parts of the *state* that correspond to the encryption of a fixed reference value. The latter approach is similar to the *state-test technique* by Boura *et al.* [9]. We show how these *state dependencies* can be traced using the propagation of differences and linear masks. By restricting the choice of input differences, we can keep part of the internal differences constant, allowing us to further reduce the required number of state guesses. Based on these techniques, we present highly effective attacks on full-round ChiLow-32 and ChiLow-40 in Sections 5 and 6. The full-round attack on ChiLow-32 can even be increased by one round, which shows that to meet the designers' security claims ChiLow-32 needs to have at least 10 rounds.

Variant	Rounds	Time	Memory	Data	Type	Ref.
ChiLow-32	3	2^{106}	2^{103}	–	MitM	[7]
	4	2^{126}	–	–	MitM	[7]
	5	2^{32}	$2^{28.65}$	$2^{18.58}$ CC	Cube	[26]
	6	2^{34}	$2^{49.28}$	$2^{33.58}$ CC	Cube	[26]
	7	$2^{121.75}$	–	80 CC	Integral	[26]
	8	$2^{120.34}$	$2^{98.32}$	160 KP	MitM	Sec. 5
	8	$2^{102.09}$	$2^{94.56}$	64 CC	MitM	Sec. 5
	9	2^{118}	$2^{106.68}$	256 CC	MitM	Sec. 5
ChiLow-40	3	2^{106}	–	–	MitM	[7]
	4	2^{126}	–	–	MitM	[7]
	8	$2^{122.32}$	$2^{114.68}$	256 CP	MitM	Sec. 6

Table 1. Overview of the attacks on ChiLow. The *memory* complexity is measured in number of ciphertexts (i.e., 32-, or 40-bit blocks). The unit of *data* complexity indicates the attack scenario: *KP* refers to a *known-plaintext*, *CP* to a *chosen-plaintext* and *CC* to a *chosen-ciphertext* attack. All attacks adhere to the security model of the designers.

[4] Available at `https://doi.org/10.5281/zenodo.20542083`

2 Preliminaries

Let $\mathbb{F}_2$ denote the field with two elements and $\mathbb{F}_2^n$ the n-dimensional vector space over this field. For a linear map $L\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ we denote the transpose of L by $L^\intercal$. For two vectors $x, y \in \mathbb{F}_2^n$, $x^\intercal y = \sum_i x_i y_i$ denotes the dot product between $x = (x_0, \ldots, x_{n-1})^\intercal$ and $y = (y_0, \ldots, y_{n-1})^\intercal$. Further, for a set $U \subseteq \mathbb{F}_2^n$ we denote by $\mathrm{span}(U)$ the linear span of U (*i.e.* all linear combinations of elements in U). Finally, $\chi_n : \mathbb{F}_2^n \to \mathbb{F}_2^n$, $x \mapsto y$ is defined as $y_i = x_i + (x_{i+1} + 1)x_{i+2}$, where n is odd, and the indices are taken modulo n [15].

2.1 Specification of ChiLow

ChiLow [7] is a family of tweakable block ciphers designed for embedded code encryption. This specific setting requires extremely low-latency decryption to support the on-the-fly execution of encrypted instructions from untrusted memory. ChiLow utilizes a 128-bit key and a 64-bit tweak. The designers propose two primary variants, ChiLow-32 and ChiLow-40, which operate on 32-bit and 40-bit states, respectively. Specifically, ChiLow-32 employs a split-path architecture with separate paths for data processing (32-bit) and tag generation (8-bit), while ChiLow-40 integrates both into a unified 40-bit state.

Initially, a whitening key is added to the state, followed by the application of eight round functions. Each round function consists of a non-linear layer χ, a linear layer L and a round key addition. The tweak and master key are processed via similar structure to derive the round keys. For a schematic representation of the ChiLow architecture, see Fig. 1.

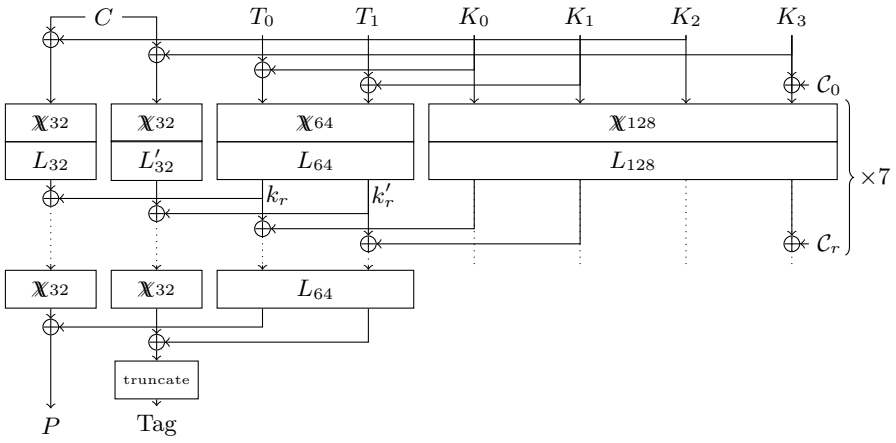

Fig. 1. Full ChiLow-32 decryption and tag computation. C denotes the ciphertext, P the plaintext, (T_0, T_1) the tweak, and (K_0, K_1, K_2, K_3) the key. The round keys are denoted by k_r and k'_r for the data and tag path respectively. We denote the round constants by $\mathcal{C}_r$.

The Non-Linear Layer The non-linear layer $\chi\!\!\chi_n \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$, with $n = 2m$ and m even, is defined as the parallel application of two χ functions on dimension $m-1$ and $m+1$, plus a linear feedforward, where $n = 32$ for ChiLow-32 and $n = 40$ for ChiLow-40 (with m being 16 and 20 respectively). More precisely,

$$\chi\!\!\chi_n(x) = \begin{pmatrix} \chi_{m-1}(x^l) \\ \chi_{m+1}(x^h) \end{pmatrix} + \lambda(x),$$

where x^l denotes the first $m-1$ bits of x, and x^h the remaining $m+1$ bits. The feed forward λ is defined as:

$$\lambda_i(x) = \begin{cases} x_m + x_{m-3} & \text{if } i = m-3 \\ x_{m-1} + x_{m-2} & \text{if } i = m-2 \\ x_{m-3} + x_m + x_{m-1} & \text{if } i = m-1 \\ x_m + x_{m-2} & \text{if } i = m \\ 0 & \text{otherwise} \end{cases}$$

The Linear Layer The linear layer $L_n \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$, where n corresponds to the state size of the specific variant, is defined as:

$$L(x)_i = x_{(\alpha \cdot i + \beta_0) \pmod n} + x_{(\alpha \cdot i + \beta_1) \pmod n} + x_{(\alpha \cdot i + \beta_2) \pmod n},$$

where $(\alpha, \beta_0, \beta_1, \beta_2)$ is $(11, 5, 9, 12)$ for ChiLow-32 ($n = 32$) and $(17, 1, 9, 30)$ for ChiLow-40 ($n = 40$). For the specific linear maps used in the tag path and the tweak- and key-schedules, we refer to the original design paper [7].

Key and Tweak Schedules ChiLow employs a 128-bit key state $K^{(i)}$ and a 64-bit tweak state $T^{(i)}$. The key schedule initializes $K^{(0)} = K$ and updates the state as

$$K^{(i+1)} = L_{128}(\chi\!\!\chi_{128}(K^{(i)} \oplus c^{(i)})),$$

where $c^{(i)}$ is a round constant, see the design paper [7] for details.

The tweak schedule initializes $T^{(0)} = T \oplus K[0:63]$ and updates

$$T^{(i+1)} = L_{64}(\chi\!\!\chi_{64}(T^{(i)})) \oplus K^{(i+1)}[0:63],$$

where $K^{(i+1)}[0:63]$ denotes the first 64 bits of the key schedule state. The last tweak round omits the non-linear layer and consists only of L_{64}.

Our attack relies on the first-round key–tweak interaction, i.e., the relation between T_0, T_1 and K_0, K_1 induced by the first tweak-schedule update.

Security Claim The designers provide multiple security claims, asserting that both the ChiLow-32 data path and ChiLow-40 are indistinguishable from a tweakable random permutation for up to 2^{40} total queries and 2^8 queries per tweak. The advantage of such a distinguisher is upper bounded by $t \cdot 2^{-128}$, where t denotes the time complexity [7, Claim 1-2].

3 A Simple Meet-in-the-Middle Attack on Full ChiLow-32

In this section, we describe a straightforward Meet-in-the-Middle (MitM) attack on the full eight rounds of ChiLow-32, demonstrating the overall attack structure, which is used in other attacks in the remainder of the paper. This attack is in the known plaintext setting. In other words, we passively observe a few plaintext-ciphertext pairs and attempt to recover the master key.

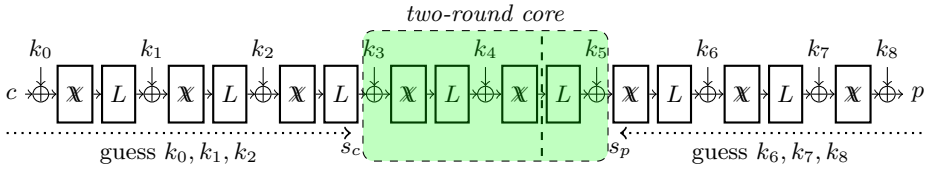

Fig. 2. Overview of the MitM attack.

MitM Stage. By guessing three full round keys from both the ciphertext and plaintext sides, we are left with two round functions and three round key additions. We refer to this structure as the *two-round core*. For a known pair (p, c), we can compute, from both sides, up to this two-round core. In other words, we can generate in- and output pairs of the core. We denote these as s_c (from the ciphertext side) and s_p (from the plaintext side), and we denote the values at the matching point as m_c and m_p. We place the matching point after 1.5 rounds of the two-round core when propagating from the ciphertext side towards the plaintext side. Here, one round consists of $\mathcal{X}$ followed by L, and a half round refers to $\mathcal{X}$ only. This configuration is depicted in Fig. 2.

To reduce the number of guessed key bits, we match using the *difference* between two plaintext-ciphertext pairs encrypted under the same tweak and key. For two pairs (p^0, c^0), and (p^1, c^1), we want to match on $\Delta m_c = m_c^0 \oplus m_c^1$ and $\Delta m_p = m_p^0 \oplus m_p^1$. Note that from the plaintext side, the round key k_5 is only linearly involved. We can directly compute $\Delta m_p = L^{-1}(s_p^0 \oplus k_5) \oplus L^{-1}(s_p^1 \oplus k_5) = L^{-1}(s_p^0 \oplus s_p^1)$ from the plaintext side for matching.

From the ciphertext side, there are two non-linear layers involved. Hence, the round keys cannot be trivially canceled. To extend the above intuition of canceling linear key bits, we compute the ANF expression of m_c in terms of s_c, k_3 and k_4. Similar to k_5, if key bit k_3, or k_4 is only linearly involved in the expression, it can be canceled by computing Δm_c. Hence, we only need to guess the key bits that are non-linearly involved in the expression. By running a quick experiment on all bits of m_c, we found that the minimum number of key bits that are non-linearly involved is 20, which happens for $m_c[15]$ and $m_c[16]$. We

choose $m_c[16]$ for the attack, the non-linear key bits are:

$$k_3 : \{0, 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17, 18, 19, 20\},\ k_4 : \{17, 18\}.$$

By guessing these 20 key bits, we can linearize $m_c[16]$ and compute $\Delta m_c[16]$ from the ciphertext side for matching.

Thus, we can meet on the single bit $\Delta m_c[16] = \Delta m_p[16]$ from both sides and obtain a 1-bit filter. We can do this for d known plaintext-ciphertext pairs that are encrypted under the same tweak and key to get a filter of $d-1$ bits. In total, we guess $g_c = 3 \cdot 32 + 20 = 116$ bits from the ciphertext side and $g_p = 3 \cdot 32 = 96$ bits from the plaintext side. This leaves us with $2^{g_c+g_p-(d-1)} = 2^{213-d}$ round key candidates.

Key Testing Stage. Each round key candidate only corresponds to part of the master key. We now focus our attention on how to recover the *full* master key from a round key candidate. An important observation, as can be seen in Fig. 1, is that the round key k_0 is equal to the master key part K_2. The second observation is that if k_1, and T_0, T_1 are known, we can recover a candidate for K_0, K_1 for each guess of the value of k_1':

$$K_0 \| K_1 = \mathbb{X}_{64}^{-1} \circ L_{64}^{-1}(k_1 \| k_1') + T_0 \| T_1.$$

After this the only part of the master key that we need to guess is K_3. To sum up, we have 2^{64} master key candidates to test for each round key candidate. The total number of master key candidates is thus $2^{g_c+g_p-(d-1)+64} = 2^{277-d}$.

Complexity Analysis. The above attack has a data complexity of d known plaintext-ciphertext pairs encrypted under the same tweak, a memory complexity of $d \cdot 2^{-5} \min(2^{g_c}, 2^{g_p}) = d \cdot 2^{91}$ (measured in the size of ciphertexts), and a time complexity of

$$d \cdot (2^{g_c} + 2^{g_p}) + 2^{g_c+g_p-(d-1)+64} \tag{1}$$

If we plug in the values of g_c and g_p, we see the optimal time complexity is $2^{123.34}$ with $d = 160$, which is less than the 2^8 pairs allowed per tweak in the security model of ChiLow.

3.1 Motivations for Improvements

The above attack is rather straightforward, but still allows us to mount a full-round attack on ChiLow-32. However, a similar attack strategy cannot be applied to 8-round ChiLow-40, as guessing 3 round keys from both sides alone leads to an attack complexity of about $2^{3 \cdot 40} = 2^{120}$. Hence, in the rest of this work, we refine the attack, aiming at both improving the attack on ChiLow-32 and extending it to ChiLow-40. We list a few points for improvements below:

1. For each bit of m_c, counting the number of non-linear key bits involved in its expression only gives an upper bound of the number of key bits needed to be guessed to linearize a bit. However, through our experiment, we noticed there are equivalent keys due to the algebraic structure of $\mathbb{X}$.
2. The current attack requires 160 known (p, c) pairs. We naturally wonder if we can launch an attack with fewer known pairs. Notice that the current attack only meets on a single bit, matching on multiple bits can further reduce the number of known (p, c) pairs to get the required matching filter.

4 Evaluating State Dependencies

In this section we analyse the precise key dependencies of one or more output bits. Since we are interested in evaluating output differences, these key dependencies will be expressed as dependencies on the internal state of a reference value. These are state dependencies. We make our analysis independent of the choice of basis to express the internal states, i.e. we are not only interested in *individual* bits, but *linear combinations* of bits as well, which we shall denote by the dot product with a *linear mask*. Here, we first make use of a well-known fact that is a direct corollary of the inverse of the Walsh-Hadamard transform.

Lemma 1. *Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $v \in \mathbb{F}_2^n$. Further, let U be the set of all linear masks u such that the linear approximation (u, v) over F has non-zero correlation, i.e., $C_{v,u}^F = 2^{-n} \sum_x (-1)^{u^\intercal x + v^\intercal F(x)} \neq 0$. To evaluate $x \mapsto v^\intercal F(x)$ it is sufficient to know $u^\intercal x$ for all $u \in U$.*

Proof. By the inverse of the Walsh-Hadamard transform we know that

$$(-1)^{v^\intercal F(x)} = \sum_{u \in \mathbb{F}_2^n} C_{v,u}^F (-1)^{u^\intercal x} = \sum_{u \in U} C_{v,u}^F (-1)^{u^\intercal x},$$

where the last step follows from $C_{v,u}^F = 0$ if $u \notin U$. □

Because of the linearity of the dot product, it is sufficient to know $u_i^\intercal x$ for a basis $u_0, \ldots, u_{d-1}$ of $\mathrm{span}(U)$ to compute $u^\intercal x$ for all $u \in U$. Hence, $d \leq n$ bits of x are enough to compute $v^\intercal F(x)$. This allows us to evaluate state dependencies by a simple propagation of linear masks.

Propagation of Linear Masks Figure 3 depicts the dependency of a difference over a function on a constant reference value x and its internal state y. The main idea is to guess parts of the state y instead of the round key k. Therefore, we are interested in which parts of y (and of input difference δ) we need to know in order to be able to compute $v^\intercal \gamma$.

From Lemma 1 it follows that these state dependencies can be evaluated by propagating the linear mask v backwards through this construction, as shown in gray in Figure 3. In accordance with Lemma 1, we can compute the set U_v of input masks such that the linear approximation (u, v) has non-zero correlation

for all $u \in U_v$. The set of input masks on (δ, y) then corresponds to $(u, u+u')$ with $u, u' \in U_v$. Therefore, it suffices to know the dot product with a basis of $\mathrm{span}(U_v)$ on δ and $\mathrm{span}(U_v + U_v)$ on y.

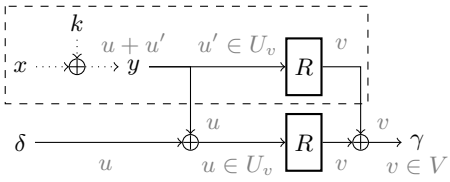

Fig. 3. Propagation of linear masks (depicted in gray) over a single round. The path in the dashed box is kept constant and can therefore be seen as a sort of *key schedule*.

Remark 1. While $\mathrm{span}(U + U) \subseteq \mathrm{span}(U)$ by construction, equality does not always hold. A trivial example is U being a (truly) affine subspace $V + a$. Here, $\mathrm{span}(U+U) = V \subsetneq \mathrm{span}(U) = V + \{0, a\}$. Hence, the dimension of $\mathrm{span}(U+U)$ might be smaller than the dimension of $\mathrm{span}(U)$.

To compute the state dependencies for a set of masks V we can use the linearity of the dot product and apply the approach described above to each basis element $\{v_0, \ldots, v_{d-1}\}$ of $\mathrm{span}(V)$. The set of input masks U with non-zero correlation over R to a mask in V is then

$$U \subseteq \bigcup_i U_{v_i}.$$

This process can trivially be iterated over multiple rounds. Figure 4 shows this process. Note that we do not need to know anything about x, since we guess the relevant part of y already. In other words, we can see the mask for x to be always zero and iterate over an arbitrary number of rounds by using U as the new output masks V. Since we have seen that we can take the linear span of the input and of the output masks, taking the linear span does not affect the state dependencies even when iterating this process over multiple rounds.

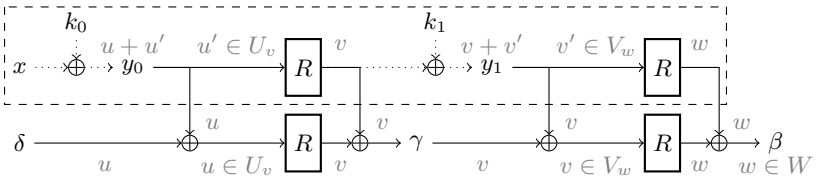

Fig. 4. Propagation of linear masks (depicted in gray) over two rounds. The path in the dashed box is kept constant.

Lowering State Dependencies using Truncated Differences To further lower the number of state dependencies, we can additionally keep part of the input constant by restricting the choice of input difference. The overall idea here is that a restriction on the intermediate differences might lead to less state dependency.

We track these intermediate restriction by means of truncated differences, which, at the highest level of abstraction, can be seen as *sets* of differences. Consider the notation from Fig. 3, and restrict the input difference δ to belong to the truncated difference Δ. We can then aggregate all masks u on δ by their equivalence modulo the masks that are constant on δ. Since we only consider linear approximations, we can assume Δ is an affine subspace of dimension k without loss of information.

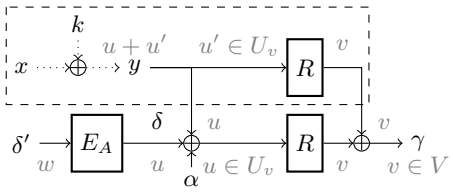

Fig. 5. Propagation of linear masks (depicted in gray) over a single round taking into account an affine input set of differences A of dimension k. E_A expands a k-bit vector δ' to an element $\delta \in A$. The path in the dashed box is kept constant.

Figure 5 shows how the input masks are aggregated by expanding a k-bit vector into the possible δ's. The correlation of the linear approximation of $(w, u+u')$ to v over the construction of Figure 5 can be written as the sum of trails corresponding to all masks u such that $E_A^\intercal u = w$, which corresponds, up to the sign, to the approximation over the construction of Fig. 3. Note here that the choice of E_A is not unique and that the reachable w depend on this choice. However, the trails that are aggregated stay the same and correspond to the equivalence classes of the masks u up to the space of masks that are constant on Δ, which is equal to $\ker(E_A^\intercal)$. Therefore, the choice of E_A does not impact the analysis of the state dependencies.

The main problem we need to solve here is how to propagate these truncated differences efficiently. Since the earlier discussion is already restricted to affine spaces of differences, we will also discuss the propagation of affine spaces here. Let Δ be such an affine space, meaning that every $\delta \in \Delta$ can be written as the linear combination of a basis $\delta_0, \ldots, \delta_{d-1}$ of its linear part plus an offset α. We

get that

$$R(x) + R(x+\delta) = R(x) + R(x + \sum_{i=0}^{d-1} \lambda_i \delta_i + \alpha)$$
$$= \sum_{j=0}^{d-1} \underbrace{\left(R(x + \sum_{i=0}^{j-1} \lambda_i \delta_i + \alpha) + R(x + \sum_{i=0}^{j} \lambda_i \delta_i + \alpha) \right)}_{\lambda_j \left(R(x+\sum_{i=0}^{j-1} \lambda_i \delta_i + \alpha) + R(x+\sum_{i=0}^{j-1} \lambda_i \delta_i + \delta_j + \alpha) \right)} + R(x) + R(x+\alpha).$$

In other words, it is enough to compute the output differences

$$\Gamma_{\delta_i} = \{R(x) + R(x+\delta_i) \mid x \in \mathbb{F}_2^n\}$$

for a basis $\delta_0, \ldots, \delta_{d-1}$ of input differences and the offset α to know that the set of output differences Γ is contained in the affine space

$$\text{span}\left(\bigcup_i \Gamma_{\delta_i} \right) + \Gamma_\alpha.$$

While this affine space might contain additional differences, these do not matter for the state dependencies of the current round. Further, it is always possible to find a basis $\gamma_0, \ldots, \gamma_e$ of this space that is contained in $\bigcup_i \Gamma_{\delta_i}$. Hence, this also does not increase the number of state dependencies in the following rounds.[5]

Key vs. State Guesses While state guesses (*i.e.*, guessing y from Fig. 3) are equivalent to key guesses (*i.e.*, guessing k) if the input difference is not restricted, such a restriction can lower the number of guesses in rounds close to the input side, and therefore limit our knowledge of the state in the path of the reference value (*i.e.*, x). In this case, guessing the state can be more efficient.

4.1 State Dependencies for Quadratic Round Functions

By the discussion above, finding state dependencies boils down to propagating differences and linear masks through the round function. In fact, it suffices to compute all output differences reachable from a given input difference, and all input masks with non-zero correlation for a given output mask.

Since the round function of ChiLow is quadratic, we will discuss propagation of differences and masks through generic quadratic functions. For the rest of this section, let $Q : \mathbb{F}_2^n \to \mathbb{F}_2^M$ be a quadratic map with component functions $Q(x)_i = x^\intercal M_i x + c_i$. Multiple choices of M_i can give rise to the same quadratic function. While this ultimately doesn't matter in this section, a unique representation can be chosen by taking M_i to be upper-triangular.

[5] In [23] a similar result is proven using subspace trails.

Propagation of Differences The output differences reachable over Q from a given input difference δ corresponds exactly to the range of the function $x \mapsto Q(x) + Q(x+\delta)$. This is because

$$\begin{aligned} Q(x)_i + Q(x+\delta)_i =& x^\intercal M_i x + c_i + (x+\delta)^\intercal M_i (x+\delta) + c_i \\ =& x^\intercal M_i \delta + \delta^\intercal M_i x + \delta^\intercal M_i \delta \\ =& \delta^\intercal M_i \delta + \delta^\intercal (M_i + M_i^\intercal) x \\ =& Q(\delta)_i + Q(0)_i + \delta^\intercal (M_i + M_i^\intercal) x. \end{aligned}$$

Written out using the matrix notation of Q, we get

$$x \mapsto Q(\delta) + Q(0) + \underbrace{\begin{bmatrix} \delta^\intercal (M_0 + M_0^\intercal) \\ \dots \\ \delta^\intercal (M_{n-1} + M_{n-1}^\intercal) \end{bmatrix}}_{N} x\,.$$

Therefore, the output differences are exactly the affine subspace $Q(\delta) + Q(0) + \mathrm{col}(N)$, where $\mathrm{col}(N)$ is the column space of N.

Propagation of Linear Masks To propagate linear masks we compute all $u \in U$ such that $C^Q_{v,u} \neq 0$ for given output mask v. Let $v^\intercal Q(x) = x^\intercal M_v x + c$ with $c = \sum_{i=0}^{n-1} v_i c_i$ and $M_v = \sum_{i=0}^{n-1} v_i M_i$. Further, let $V = \ker(M_v + M_v^\intercal)$ be the kernel of $M_v + M_v^\intercal$. For all $y \in V$ and $x \in \mathbb{F}_2^n$ we get that

$$\begin{aligned} (x+y)^\intercal M_v (x+y) =& x^\intercal M_v x + x^\intercal M_v y + y^\intercal M_v x + y^\intercal M_v y \\ =& x^\intercal M_v x + x^\intercal (M_v + M_v^\intercal) y + y^\intercal M_v y \\ =& x^\intercal M_v x + y^\intercal M_v y. \end{aligned}$$

This has two implications. First, $v^\intercal Q(x)$ is affine on V and we write $x^\intercal M_v x = w^\intercal x$ if $x \in V$. Secondly, the sum in the computation of $C^Q_{v,u}$ can be split into the product of two sums, one over V and another over an algebraic complement V^c:

$$2^n (-1)^c C^Q_{v,u} = \sum_{x \in \mathbb{F}_2^n} (-1)^{x^\intercal M_v x + u^\intercal x} = \sum_{z \in V^c} (-1)^{z^\intercal M_v z + u^\intercal z} \underbrace{\sum_{y \in V} (-1)^{(u+w)^\intercal y}}_{\neq 0 \text{ iff } (u+w) \perp V}.$$

It follows that U is a subset of $w + \mathrm{col}(M_v + M_v^\intercal)$. In fact, this is an equality, because $z^\intercal M_v z$ is bent over V^c. This follows, for example, from [25, Problem 14 and 15 of Ch. 14] and the fact that $B^\intercal (M_v + M_v^\intercal) B$ is of full rank if B is a matrix with the basis elements of V^c as columns.

Computing state dependencies To wrap up, we summarize how everything comes together to compute the state dependencies. Consider the evaluation of

an output difference on a given space of output masks V over the composition of quadratic round function given an affine subspace Δ of possible input differences. The goal is to evaluate which masks on y give a non-zero correlation approximation in the construction of Fig. 5. To do this, we first derive the possible input difference and output masks in each round in the non-constant part of Fig. 4. This can be done using the techniques discussed earlier in this section.

Now we can analyse Fig. 5 for each round separately. However, instead of analysing the trails, we note that this construction is also a quadratic function, so we can apply the earlier analysis for propagation of linear masks directly. The state dependencies then follow from projecting the space of input masks on the state.

4.2 Application to ChiLow

Let us look at the state dependencies for ChiLow when either the output masks or the input differences are the full space $\mathbb{F}_2^n$. As χ is the only non-linear component in ChiLow, propagation through a single round can be trivially reduced to propagation through χ. We can write χ as

$$x \mapsto x^{\mathsf{T}} \underbrace{\begin{bmatrix} e_0+e_{-2} & e_{-1} & 0 & 0 & \dots & e_{-2} \\ & e_1+e_{-1} & e_0 & 0 & \dots & 0 \\ & & \ddots & \ddots & \dots & \vdots \\ & & & \ddots & \ddots & 0 \\ & & & & e_{-2}+e_{-4} & e_{-3} \\ & & & & & e_{-1}+e_{-3} \end{bmatrix}}_{M} x,$$

where the entries of M are vectors from $\mathbb{F}_2^n$,[6] and e_i denotes the i-th unit vector, with indices interpreted modulo (i.e., $i = i \pmod n$). Hence, $M + M^{\mathsf{T}}$

$$M + M^{\mathsf{T}} = \begin{bmatrix} 0 & e_{-1} & 0 & 0 & \dots & e_{-2} \\ e_{-1} & 0 & e_0 & 0 & \dots & 0 \\ 0 & e_0 & \ddots & \ddots & \dots & 0 \\ 0 & 0 & \ddots & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & e_{-3} \\ e_{-2} & 0 & 0 & 0 & e_{-3} & 0 \end{bmatrix}$$

Propagation of Differences For the propagation of a single difference δ through χ the discussion above boils down to

$$N = \delta^{\mathsf{T}}(M + M^{\mathsf{T}}) = (\delta_1 e_{-1} + \delta_{-1} e_{-2}, \delta_0 e_{-1} + \delta_2 e_0, \dots, \delta_0 e_{-2} + \delta_{-2} e_{-3}),$$

[6] This makes M a 2-dimensional representation of a 3-dimensional *hypermatrix*.

where we can interpret the entries of this vector as columns of a matrix over $\mathbb{F}_2$. Expanding the vector in this way yields

$$N = \begin{bmatrix} 0 & \delta_2 & \delta_1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \delta_3 & \delta_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \ddots & 0 \\ \delta_{-1} & 0 & 0 & 0 & 0 & \dots & \delta_0 \\ \delta_1 & \delta_0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix},$$

which is similar to the representation of the derivative of χ originally observed by Daemen [15, Section 6.9.1]. Since the number of possible output differences is equal to the size of the column space of N, the number of output differences is minimized by using unit vectors as input differences to χ and therefore to ChiLow. Additionally, to increase the overlap of the state dependency between multiple differences, the subspace is best spanned by consecutive unit vectors. Here, consecutive is with respect to a single χ function in the decomposition of $\mathcal{K}$. We, heuristically, assume that the best choices for one round also work well for two rounds.

Table 2 contains the best subspaces we found. Note that going below dimension 6 would not leave enough ciphertexts to match on, and above dimension 8 is unnecessary due to the data limit per tweak. We only give one example per dimension, but there are many equivalent choices (due to the rotational self-equivalence of the χ function).

Block Size	Dimension	Non-constant Bits	State Dependency (/Bits)
32	6	2 – 7	33
	7	2 – 8	35
	8	2 – 9	37
40	6	10 – 15	40
	7	10 – 16	43
	8	10 – 17	45

Table 2. Required number of state bits to be known to evaluate the full output difference after two rounds of ChiLow when the input difference is restricted to the given non-constant bits.

Propagation of Linear Masks By the discussion above, the output mask v propagates through χ to the affine space with linear part equal to the column

space of

$$M_v + M_v^\intercal = \begin{bmatrix} 0 & v^\intercal e_{-1} & 0 & 0 & \dots & v^\intercal e_{-2} \\ v^\intercal e_{-1} & 0 & v^\intercal e_0 & 0 & \dots & 0 \\ 0 & v^\intercal e_0 & \ddots & \ddots & \dots & 0 \\ 0 & 0 & \ddots & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & v^\intercal e_{-3} \\ v^\intercal e_{-2} & 0 & 0 & 0 & v^\intercal e_{-3} & 0 \end{bmatrix} = \begin{bmatrix} 0 & v_{-1} & 0 & 0 & \dots & v_{-2} \\ v_{-1} & 0 & v_0 & 0 & \dots & 0 \\ 0 & v_0 & \ddots & \ddots & \dots & 0 \\ 0 & 0 & \ddots & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & v_{-3} \\ v_{-2} & 0 & 0 & 0 & v_{-3} & 0 \end{bmatrix}.$$

As can be seen, the dimension of this space for non-zero v is at least 2. For a full round, the masks leading to this minimal dimension of 2 are

1. unit vectors (Hamming weight one), or
2. vectors with exactly two consecutive nonzero bits that both lie within the domain of a single χ function (i.e., both within the first or both within the second χ of $\mathcal{X}$).

We now consider the state dependency over 1.5 rounds (i.e. $\mathcal{X}$ followed by one full round). For the masks listed above, we find that the state dependency in both ChiLow versions ranges from 17 to 20 bits for the first type and from 22 to 29 bits for the second type. Since the second type is strictly worse, we will restrict our search for larger affine subspaces of masks to those spanned by unit vectors. Table 3 gives the 0-, 1-, and 2-dimensional affine mask spaces with the smallest state dependency within our search space.

Block Size	Matching Bits	State Dependency (/Bits)
32	$\{15\}$ or $\{16\}$	17
	$\{3,4\}$, $\{4,5\}$, $\{5,6\}$ or $\{26,27\}$	23
	$\{0,1,14\}$, $\{3,4,5\}$, ... (7 more instances)	27
40	$\{19\}$ or $\{20\}$	17
	$\{4,5\}$ or $\{20,21\}$	24
	$\{3,4,5\}$, $\{4,5,6\}$, $\{20,21,22\}$, $\{30,31,32\}$	28

Table 3. Required number of state bits to be known for matching on the given bits of the output x of 1.5 rounds of ChiLow. The other instances for 3 matching bits can be found in the supplementary material.

5 Optimizing Attacks on Full-Round ChiLow-32

In this section we propose two optimizations of the base attack in Section 3 by using the techniques from the last section.

We first look at an optimization in the known plaintext setting, which is solely based on the propagation of linear masks. After that we use chosen ciphertexts and the propagation of (truncated) differences to reduce the time complexity further by fixing part of every ciphertext to a chosen constant. This complicates the master key recovery as we do no longer guess the first two round keys in their entirety, however, this can be solved by adding a round key derivation step to the attack.

5.1 An Optimized Known Plaintext Attack

Our results show that the attack of Section 3 can be performed with 3 fewer bit guesses. However, the attack must be modified slightly to accommodate internal state guesses instead of key guesses. We encrypt the ciphertexts over the first 3 rounds and then compute the internal state difference with respect to one reference ciphertext. These differences can then be propagated using internal state guesses of this reference ciphertext. The state bits that must be guessed are bits 17 and 18 of the state before the fifth χ evaluation, and the following bits of the state before the fourth χ evaluation:

$$\{1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20\}.$$

In fact, these state guesses are equivalent to guessing the corresponding bits of the round keys.

Optimizing Equation (1) results in an attack with time complexity $2^{120.34}$ and 160 known plaintexts. We can also match on more bits, however, since the ciphertext side already guesses more bits than the plaintext side, this only improves the data complexity of the attack, not the time complexity.

5.2 Chosen Ciphertext Attack

To reduce the number of bits guessed on the ciphertext side of the attack, we further lower the state dependencies by fixing part of the ciphertexts to a constant, as described in Section 4. Hence, we use *chosen* instead of *known* ciphertexts.

Since the ciphertext side covers 4.5 rounds (involving 5 χ functions), and since propagating differences forwards (and masks backwards) lasts at most two χ functions until all differences (and masks) appear, differences and masks never interact with each other when propagating as outlined in Section 4. This means that we can optimize the propagation of differences and the propagation of masks independently and make direct use of the results presented in Tables 2 and 3.

But since this means that we recover less of the state in the first two rounds from the ciphertext side, we need to augment the master key recovery part of the attack. We recall that, previously, we used the fact that we fully recover these round keys to easily recover the master key. To overcome this limitation, we check each of the 2^{64} possible round keys (k_0, k_1) against each internal state candidate in the following manner. Let g_2 be the number of internal state bits guessed before each of the first two χ functions. Then, the number of round

key candidates per internal state candidate is 2^{64-g_2}. This can be derived from the fact that the internal state bits are balanced Boolean functions of the round keys. To compute the total time complexity of the modified attack we take ℓ to be the number of internal state difference bits that we match on. In total, this gives a time complexity of

$$d \cdot (2^{g_c} + 2^{g_p}) + 2^{g_c+g_p-\ell(d-1)+64-g_2+64}. \tag{2}$$

An Optimized Chosen Ciphertext Attack We now give an optimized attack after evaluating the different options for the dimension of the difference space, and the number of matching bits and the amount of data. The best attack on ChiLow-32, based on the techniques presented in this work, has a time complexity of $2^{102.09}$, uses 64 chosen ciphertexts, and has a memory complexity of $2^{94.56}$ state sizes. This result is achieved using the 6-dimensional subspace of Table 2 and matching on three bits with the following indices $\{3, 4, 5\}$ from Table 3. We guess a total of 92 state bits on the ciphertext side, which means that the plaintext side now dominates the attack. Table 4 gives us the state bits that are guessed before each $\mathbb{X}$ function to evaluate the property.

State	Guessed bits
1	$\{1, 2, 3, 4, 5, 6, 7, 8\}$
2	$\{0, \ldots, 30\} \setminus \{8, 11, 17, 20, 23, 26, 31\}$
3	$\{0, \ldots, 31\}$
4	$\{0, \ldots, 30\} \setminus \{3, 6, 10, 15, 16, 17, 21, 28\}$
5	$\{4, 5, 6, 7\}$

Table 4. State bits guessed in the ciphertext direction of our best Meet-in-the-Middle attack on ChiLow-32.

An Extension to 9 Rounds We can extend this attack to 9 rounds by adding one additional round at the ciphertext side. We use the 8-dimensional subspace of input differences from Table 2, which we denote by Δ, and match on a single bit (Table 3). Note that Δ corresponds to the only non-constant bits being bits 2 to 9, and that the state dependencies in the first round contain these bits. Since we fully saturate these bits by effectively propagating the ciphertexts $c + \Delta$, we can move the 8 state guesses of bits 2 to 9 before the first round to the plaintext side by reducing the state guesses to an algebraic complement of Δ and fixing an order of the states at the input of the first round in advance. On the plaintext side, we first guess these 8 bits of the state at the ciphertext side to ensure the same order, allowing us to continue as before.

Hence, we guess $g_c = 37 + 2 \cdot 32 + 17 - 8 = 110$ bits from the ciphertext side, while guessing $g_p = 3 \cdot 32 + 8 = 104$ bits from the plaintext side. This leads to

time and memory complexities of 2^{118} and $2^{106.68}$, respectively. This shows that even a 9-round variant of ChiLow-32 is vulnerable to MitM attacks.

6 A Chosen Plaintext Attack on Full-Round ChiLow-40

The main obstacle to applying the approach from the last section to ChiLow-40 is that the two MitM parts become too imbalanced. As guessing on the plaintext side already has a time complexity $d \cdot 2^{120}$, this makes such an attack prohibitively expensive. We can overcome this by moving the difference propagation to the plaintext side. To make sure we stay in the security model of the designers, we use the 6-dimensional space Δ of differences from Table 2, which has 8 state dependencies over a single round and propagates to differences fully contained in an 8-dimensional space Γ. Further, we fix the path of our reference value by fixing the plaintext p instead of the ciphertext c.

The overall structure of the attack is shown in Fig. 6. Starting at the left of the addition of k_7, we can simply guess the reference value y, as well as k_5 and k_6 to compute the difference at the matching point m_p. As before, we only need to guess y in an algebraic complement Δ^c of Δ, since we effectively propagate the values $y + \Delta$ and we can simply fix an order for these values in advance. Hence, $d = 2^6$ and $g_p = 3 \cdot 40 - 6 = 114$.

Of course, we need to make sure that we use the same order when computing in the other direction. But the state dependencies for computing the plaintext difference already contain the part of y that belongs to Δ, allowing us to ensure the same order. Since we fix the reference plaintext p and since the output differences are contained in Γ, which is of size 2^8, we can ask for the encryption of all plaintexts in $p+\Gamma$ to get a set of ciphertexts $\mathcal{C}$. Aside from the aforementioned 8 state dependencies to compute the plaintext differences, we fully guess k_0 and k_1, and guess 28 bits of k_2 and k_3 to compute 3 bits (see Table 3) of the difference at the matching point m_c. Hence, $g_c = 8+2\cdot 40+28 = 116$. Plugging these values into the ChiLow-40 equivalent version of Eq. (2) yields a time complexity of

$$d \cdot (2^{g_c} + 2^{g_p}) + 2^{g_c+g_p-\ell(d-1)+48} \approx 2^{122.32},$$

where we leave out the additional key recovery term as we are back to fully guessing the first two round keys. Memory complexity becomes $d/40 \min(2^{g_p}, 2^{g_c}) \approx 2^{114.68}$. Hence, this attack breaks the security claim for ChiLow-40.

7 Countermeasures

While the security of a cipher with block size n and security level κ against *trivial* MitM attacks can be ensured by using $R > 2\lceil \frac{\kappa}{n} \rceil$ rounds, we have seen in Section 3 that even slightly more sophisticated MitM attacks might still be applicable. Using the ideas from Section 4 enables us to push MitM attacks even further. But the insights gained in Section 4 also enable us to propose potential modifications of ChiLow that would allow to prevent the attacks presented in this work.

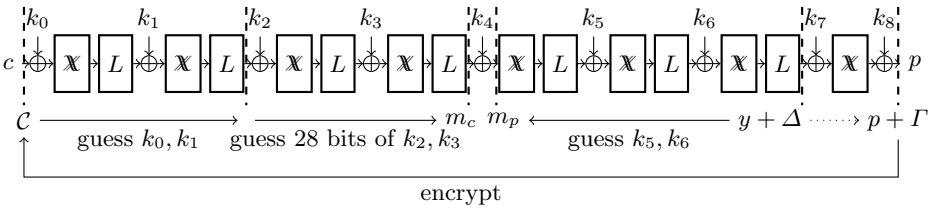

Fig. 6. A chosen plaintext attack on full-round ChiLow-40.

Increasing the Number of Rounds A simple approach to increase the security is to increase the number of rounds. We were able to attack up to 9 rounds of ChiLow-32 in Section 5 and 8 rounds of ChiLow-40 in Section 6. Since these attacks are relatively close to the security claim, it should therefore suffice to increase the number of rounds to 10 for ChiLow-32 and 9 for ChiLow-40 . This approach does however naturally increase the circuit depth, and therefore the latency of the cipher.

Higher Diffusion Another approach could be to increase the diffusion to ultimately increase the number of state dependencies. However, any increase in circuit depth of the round function implies an amplified increase of overall latency. In addition, the gains from small changes to the round function are expected to be low, especially for state dependencies over a single round. Hence, without significant changes to the design, increasing the number of rounds appears to be more promising.

Higher State Size Another counter measure could be to increase the state size. While we understand the plaintext size to be fixed by ChiLow's intended use case, the tag size is not. Hence, at least in the case of ChiLow-40, it would be feasible to increase the state size this way. Moreover, an increased tag size would have obvious security benefits. But it also comes at the cost of a higher ciphertext size, ultimately requiring more memory to store the code that ChiLow is intended to protect.

8 Conclusion

We presented full-round Meet-in-the-Middle attacks on the ChiLow family. These attacks exploit the relatively weak diffusion of the round function, and the small state size of the data path of ChiLow to construct efficient Meet-in-the-Middle attacks. Our best attack on ChiLow-32 has a time complexity of $2^{102.09}$, which breaks its security claim. Similarly, on ChiLow-40, using techniques in the chosen-plaintext setting, we present an attack with a time complexity of $2^{122.32}$, which also invalidates its claimed security.

However, these attacks are far from practical, due to their large time and memory complexity. Furthermore, our attacks should not be mistaken as arguments against the general design of ChiLow, but rather show that the designers were slightly too optimistic in their pursuit of low latency. As discussed in the last section, the best countermeasure against our attacks might just be to increase the number of rounds to 10 for ChiLow-32, and 9 for ChiLow-40.

Acknowledgements This work was independently conceived during Michiel's visit to Inria, Paris and at Nanyang Technological University, Singapore. The authors would like to thank Gaëtan Leurent for the visit and Jian Guo and Yiran Yao for early stage discussions in Singapore. This work was supported by the project *Cryptanalyse* from PEPR Cybersécurité (22-PECY-0010). This work was supported by CyberSecurity Research Flanders with reference number VR20192203. Additionally, this research is funded by the European Union (ERC-2023-COG, SoBaSyC, 101125450). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. Besides, this research is partially supported by the Ministry of Education in Singapore under Grant RG102/24.

References

1. Samuele Andreoli, Gregor Leander, Enrico Piccione, and Lukas Stennes. Generalizations of ChiChi: Families of Low-Latency Permutations in Any Even Dimension. *IACR Transactions on Symmetric Cryptology*, 2025(3):800–826, September 2025.
2. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 578–597. Springer, Berlin, Heidelberg, December 2009.
3. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 70–89. Springer, Berlin, Heidelberg, August 2009.
4. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 103–119. Springer, Berlin, Heidelberg, August 2009.
5. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. Superposition meet-in-the-middle attacks: Updates on fundamental security of AES-like hashing. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 64–93. Springer, Cham, August 2022.
6. Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. BipBip: A low-latency tweakable block cipher with small dimensions. *IACR TCHES*, 2023(1):326–368, 2023.
7. Yanis Belkheyar, Patrick Derbez, Shibam Ghosh, Gregor Leander, Silvia Mella, Léo Perrin, Shahram Rasoolzadeh, Lukas Stennes, Siwei Sun, Gilles Van Assche, and Damian Vizár. ChiLow and ChiChi: New constructions for code encryption. In EUROCRYPT 2025, Part I [18], pages 212–243.

8. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 229–240. Springer, Berlin, Heidelberg, August 2011.
9. Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 179–199. Springer, Berlin, Heidelberg, December 2014.
10. Federico Canale, Tim Güneysu, Gregor Leander, Jan Philipp Thoma, Yosuke Todo, and Rei Ueno. SCARF - A low-latency block cipher for secure cache-randomization. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 1937–1954. USENIX Association, August 2023.
11. Anne Canteaut, editor. *FSE 2012*, volume 7549 of *LNCS*. Springer, Berlin, Heidelberg, March 2012.
12. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 222–240. Springer, Berlin, Heidelberg, August 2013.
13. Shiyao Chen, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang. Diving deep into the preimage security of AES-like hashing. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 398–426. Springer, Cham, May 2024.
14. Shiyao Chen, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang. Scrutinizing the security of AES-based hashing and one-way functions. Cryptology ePrint Archive, Paper 2025/792, 2025.
15. Joan Daemen. *Cipher and Hash Function Design. Strategies based on linear and differential cryptanalysis.* PhD thesis, Katholieke Universiteit Leuven, 1995. Joos Vandewalle and René Govaerts (promotor).
16. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.
17. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, 34(3):33, July 2021.
18. *EUROCRYPT 2025, Part I*, LNCS. Springer, Cham, June 2025.
19. Antonio Flórez-Gutiérrez, Eran Lambooij, Gaëtan Leurent, Håvard Raddum, Tyge Tiessen, and Michiel Verbauwhede. Cryptanalysis of full SCARF. In EUROCRYPT 2025, Part I [18], pages 397–426.
20. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 56–75. Springer, Berlin, Heidelberg, December 2010.
21. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family. In Canteaut [11], pages 244–263.
22. Eran Lambooij, Patrick Neumann, Michiel Verbauwhede, Shichang Wang, and Tianyu Zhang. Meet-in-the-middle attacks on full chilow (supplementary material), June 2026.
23. Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. Searching for subspace trails and truncated differentials. *IACR Trans. Symm. Cryptol.*, 2018(1):74–100, 2018.

24. Ji Li, Takanori Isobe, and Kyoji Shibutani. Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to SHA-2. In Canteaut [11], pages 264–286.
25. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North Holland, 1977.
26. Shuo Peng, Akram Khalesi, Zahra Ahmadian, Hosein Hadipour, Jiahui He, Kai Hu, Zhongfeng Niu, Shahram Rasoolzadeh, and Meiqin Wang. Cube and integral attacks on chilow-32. *IACR Transactions on Symmetric Cryptology*, 2026(1):253–292, Mar. 2026.
27. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 253–271. Springer, Berlin, Heidelberg, December 2008.
28. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 134–152. Springer, Berlin, Heidelberg, April 2009.
29. Jinliang Wang, Christina Boura, Patrick Derbez, Kai Hu, Muzhou Li, and Meiqin Wang. Cryptanalysis of full-round BipBip. *IACR Trans. Symm. Cryptol.*, 2024(2):68–84, 2024.
30. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved meet-in-the-middle cryptanalysis of KTANTAN (poster). In Udaya Parampalli and Philip Hawkes, editors, *ACISP 11*, volume 6812 of *LNCS*, pages 433–438. Springer, Berlin, Heidelberg, July 2011.