Narrow-Bicliques: Cryptanalysis of Full IDEA

Dmitry Khovratovich[1], Gaëtan Leurent[2], and Christian Rechberger[3]

[1] Microsoft Research, USA
[2] University of Luxembourg, Luxembourg
[3] DTU, Denmark

Abstract. The biclique attack framework was recently introduced as a way to add more rounds to a meet-in-the-middle attack while potentially keeping the same time complexity. We apply and extend the recently introduced biclique framework to IDEA and for the first time describe an approach to noticeably speed-up key-recovery for the full 8.5 round IDEA.

We also show that the biclique approach to block cipher cryptanalysis not only obtains results on more rounds, but also improves time and data complexities over existing attacks. We consider the first 7.5 rounds of IDEA and demonstrate a variant of the approach that works with practical data complexity.

The conceptual contribution is the narrow-bicliques technique: the recently introduced independent-biclique approach extended with ways to allow for a significantly reduced data complexity with everything else being equal. For this we use available degrees of freedom as known from hash cryptanalysis to *narrow* the relevant differential trails. Our cryptanalysis is of high computational complexity, and does not threaten the practical use of IDEA in any way, yet the techniques are practically verified to a large extent.

Keywords: block ciphers, bicliques, meet-in-the-middle, IDEA, key recovery

1 Introduction

Since Rijndael has been chosen as a new cipher standard in 2001, block cipher cryptanalysis has been less attractive for the cryptologic community. It may be partly attributed to the eStream and SHA-3 competition, which essentially diverted the attention of cryptanalysts to the design and analysis of new primitives. The most efficient methods — differential and linear cryptanalysis, square attacks, boomerang, meet-in-the-middle and impossible differential attacks — have all been designed in the 90s or earlier, and undergone only a series of evolutionary improvements. Occasional applications of hash function-specific methods like rebound attacks operate mainly in weaker models of security.

The situation seems to change with the recent introduction of biclique attacks on AES [6]. Even considered an extension to meet-in-the-middle attacks, the biclique attack brings new techniques and tools to the world of block ciphers, which were known mainly in the cryptanalysis of hash functions. In contrast to earlier attempts to cryptanalyze AES[5], the new approach does not use any related keys. To understand the reasons behind the new results and to motivate our work, we proceed with a more detailed story of meet-in-the-middle attacks and their evolution.

Meet-in-the-middle attacks on block ciphers. The basic idea of meet-in-the-middle attacks is to split an invertible transformation into two parts and separate parameters that are involved in only one part. Then these parameters can be searched independently with a match in the middle as a certificate of a right combination. One of the first applications is the cryptanalysis of DoubleDES $E_{K_2}(E_{K_1}(\cdot))$, which demonstrated that the total security level is not the sum of key lengths [12]. The reason is that given a plaintext/ciphertext pair, an adversary is able to compute the internal middle state of a cipher trying all possible values of K_1 and K_2 independently.

The same principle applies at the round level as well. If there is a sequence of rounds in a block cipher that does not depend on a particular key bit, the meet-in-the-middle attack might work. However, its application has been limited by the design of block ciphers, the majority of which use the full key in the very first rounds of a cipher. As a result, even as little as a half of a cipher is rarely attacked, with four attacked rounds in AES [8] and seven in DES [9,13]. Compared to 7-round attacks on AES [20], and full 16-round attacks on DES [21], the meet-in-the-middle attacks were clearly inferior to other methods in spite of their impressively low data complexity. The widespread use of meet-in-the-middle attacks against the preimage resistance of hash functions follows this argument, as the message schedule of, e.g., SHA-1, admits as many as 15 rounds being independent of some message bits. The block ciphers KTANTAN [7] and GOST [15], recently attacked within the meet-in-the-middle framework, also do not use the full key for large number of rounds.

In this context the recent meet-in-the-middle attacks on the full AES [6] might look as a counterexample. Nevertheless, they have not disclosed any new key schedule properties. However, they are able to cover as many as 6 rounds with a new construction — a biclique, inherited from hash function cryptanalysis [17]. In addition to the aforementioned length of the biclique, its dimension is another important property, and significantly contributes to the computational advantage compared to brute-force approaches. A biclique does not impose constraints on the key schedule decomposition and can be long enough to add a significant number of rounds to a meet-in-the-middle attack. The latter property is, however, difficult to achieve when aiming for a significant advantage over brute-force. When dealing with the full number of rounds, only AES-192 has faced an improvement in a factor of four or larger. We additionally stress that the non-ideal diffusion of a single AES round is an important factor for these results. If AES had a full MDS matrix as the diffusion layer, like SHARK [22], it would be much more difficult to attack. The data complexity would increase greatly.

All these properties are significant issues when one wants to deal with a cipher that achieves full diffusion in a single round. Hence IDEA, which has this property, is a natural challenge for a biclique attack. As explained further, we have to leap over three well-diffusion rounds to successfully cryptanalyze the full version of the cipher.

1.1 Cryptanalytic attacks on IDEA and our contribution.

The "International Data Encryption Standard" (IDEA) is one of the longest standing and most analyzed ciphers known. It was designed by Lai and Massey in 1991 [18, 19]. Except for negligibly small classes of weak keys only reduced round variants up to 5 out of its 8.5 rounds have been cryptanalyzed in the most relevant single-key setting. If the cryptanalyst were to choose arbitrary sequences of middle-rounds results go up to 6 rounds, and in the less relevant related-key model up to 7.5 rounds [2].

We consider the starting rounds only as a more difficult and more natural challenge. Attacks on middle-rounds weaken the cipher considerably as there would be no equivalent of a whitening key. Moreover, the full key would be used only after two rounds of the cipher, which is evidently not a property of the actual design.

Our main technical results are a first method for key recovery of full IDEA noticeably faster than brute force search, and improved attacks on round-reduced variants. An overview of our new results as well as a comparison with earlier work is given in Table 1. We list here the conceptually new approaches that eventually led to this result:

- The independent-biclique strategy allows for higher dimensions which in turn can lead to faster key recovery. A straightforward application to IDEA would lead to the full codebook requirement even for a one-round biclique due to the diffusion properties. It drastically differs from AES, where 3-round bicliques may still yield reasonable data complexity. In this paper we extend the independent-biclique framework to allow for lower data complexity requirements. We achieve this by using available degrees of freedom for limiting the diffusion in spite of high dimension. Hence we introduce the prefix *"narrow"*.
- In earlier work on AES the independent-biclique was always combined with a key testing phase that loops over all keys. This combination is however not necessary and many of our attacks do not require testing all keys.
- In previous meet-in-the-middle style attacks the Biryukov-Demirci relation was used in a differential way to cancel out key dependencies, it was termed "keyless Biryukov-Demirci relation". Used in our framework, the BD relation can be used directly, and hence avoids overhead computations.

To illustrate the flexibility of the narrow-biclique approach to IDEA, we also consider round-reduced versions. As an example, consider IDEA reduced to the

Table 1. New key recovery for IDEA. By brute-force, the computations/success rate ratio is 2^{128}. For simplicity, only round-reduced variants starting from the actual beginning of the cipher are considered.

Rounds	Data	Comp./succ.rate	Memory	Ref.	Biclique rounds	Matching
first 5	2^{17}	$2^{125.5}$	n.a.	[24]	-	key-dep. BD
first 5	10	2^{119}	2^{24}	[2]	-	differential BD
first 5	2^{64}	$2^{115.3}$	n.a.	[24]	-	key-dep. BD
first 5	2^{25}	2^{110}	2^{16}	Sec. 6	1	direct
first 5	2^{25}	$2^{101.5} + 2^{112}$MA	2^{110}	Sec. 6	1	direct
first 6	2^{41}	$2^{118.9}$	2^{12}	Sec. 7	1	direct BD
first 7.5	2^{18}	$2^{126.5}$	2^{3}	Sec. 8	1.5	direct BD
first 7.5	2^{52}	$2^{123.9}$	2^{7}	Sec. 8	1.5	direct BD
8.5 (full)	2^{52}	$2^{126.06}$	2^{3}	Sec. 5	1.5	direct BD
8.5 (full)	2^{59}	$2^{125.97}$	2^{3}	Sec. 5	1.5	direct BD

first 5 rounds, which is the highest number of rounds that allowed results before. For this we simultaneously improve time and memory complexity over other pervious attacks, while at the same time achieving a practical data complexity of only 2^{25} chosen plaintexts. We also describe the first attack on the 6 initial rounds of IDEA, with data complexity 2^{41} and time complexity 2^{119}, which is a significant improvement over brute-force.

Independently and concurrently, Biham et al. [3] use similar techniques and also arrive at improvements over previous work. However, for the same variant of IDEA considered, the data complexities we obtain compare favourably to theirs for the reasons outlined above. Note that in [3] middle rounds are considered, which we exclude for reasons outlined above. Hence for the same number of rounds we attack a stronger cipher. For full IDEA, whereas we focus on minimizing time complexity, Biham et al. consider a setting with extremely little available data and for this have a different approach that is closer to the time complexity of brute force. Overall, even though the approaches are similar at a high level, they differ in their details and might well benefit from each other.

In [4, 10, 14] classes of weak-keys were found for full IDEA. Even though the class is for all practical purposes negligibly small (only up to a fraction of 2^{-64} of all keys are affected), a small change for IDEA was proposed to get rid of these weak-key properties [10]: a constant addition in the key schedule. Our approach to key recovery even works for those strengthened variants of IDEA, in exactly the same way, because it is independent of such constants.

2 Description of IDEA

In here we give a brief description of IDEA and discuss implementation cost considerations that lead to a cost model in which we evaluate our subsequent cryptanalytic results.

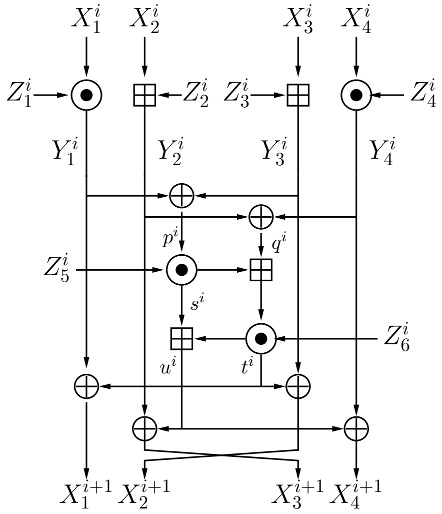


Fig. 1. One round of IDEA.

IDEA is a 8.5-round block cipher with a 64-bit state and a 128-bit key. Internal state and subkeys are treated as 16-bit words. Each round is an invertible transformation and follows the key addition (KA) layer (two multiplications, two modular additions) with the multiplication-addition (MA) function (again, two multiplications, two modular additions). Addition is performed modulo 2^{16}. Multiplication is performed modulo $2^{16} + 1$, where 0 is replaced with 2^{16}.

We denote input variables to round i by X^i, the subkeys of round i by Z^i. Additional input variables are depicted in the outline of a single round in Figure 1. Key bits in subkeys are listed in Table 2, where the leftmost bit is the most significant bit in the 16-bit word.

Compared to all other operations of IDEA, the multiplication modulo $2^{16}+1$ is the most expensive. It can either be realized as a 17-bit multiplier, using a table of size 2^{16}, or with the help of two or three lookup tables of size 2^8. This naturally motivates the model we use to estimate time complexities in this paper: counting multiplications and/or table lookups, and relating them to the number of multiplications needed for IDEA. Each round of IDEA needs four of these multiplications, and the additional key addition layer at the end (often counted as 0.5 round) needs another two. Hence in total 34 multiplications are needed for a single computation. Some of our attacks require the computation of a single output bit of the multiplication which we model with a cost of 0.5 multiplications (see references to complexity estimates in various models in [25]).

3 Biclique attack

Biclique attacks were introduced for hash function cryptanalysis [17] as an extension to the initial structure technique [23], and later applied to block ciphers [6]. In the biclique attack on block ciphers the full key space is partitioned into groups of keys, so that keys in a group can be efficiently tested in the meet-in-the-middle framework.

The keyspace partition can be described in various ways. For permutation-based key schedules as in IDEA we simply introduce three sets of key bits: K^b, K^f, and K^g. In a key group the value K^g is fixed (and hence enumerates the groups), and K^b and K^f take all possible values.

Biclique. A *biclique* is a set of internal states, which are constructed either in the first or in the last rounds of a cipher and mapped to each other by specifically chosen keys. We consider the former option only in the paper. Let f be the mapping describing the first cipher rounds, then a biclique for a group K^g is a set of states $\{P_i\}, \{S_j\}$ such that

$$P_i \xrightarrow[f]{K^b=i\,||\,K^f=j} S_j.$$

Keys in a group are tested as follows. A cryptanalyst asks for the encryption of plaintexts P_i and gets ciphertexts C_i. Then he checks if

$$\exists\, i, j:\ S_j \xrightarrow[g]{K^b=i\,||\,K^f=j} C_i, \tag{1}$$

where g maps states S_j to ciphertexts. A biclique is said to have dimension d, if both K^b and K^f have d bits.

Key testing. Each key group is tested separately. There are two approaches to test keys within a group. In the first approach a cryptanalyst uses an intermediate variable v that can be computed in both directions:

$$S_j \xrightarrow[g_1]{K^f=j} v \overset{?}{=} v \xleftarrow[g_2]{K^b=i} C_i.$$

The functions g_1 and g_2 are called *chunks*. This approach is illustrated in Figure 2. The computational complexity of testing a single group is

$$C_{biclique} + 2^{|K^f|}C_{g_1} + 2^{|K^b|}C_{g_2} + C_{recheck},$$

where C_{g_1} and C_{g_2} are the costs of computing v, $C_{biclique}$ is the biclique construction cost, and $C_{recheck}$ is the cost of rechecking key candidates on other state bits or another plaintext/ciphertext pair. The full complexity is derived by the multiplication on the total number of groups.

In the second case a cryptanalyst is unable to find a variable with these properties. Then he tests each key individually

$$S_j \xrightarrow[g_1]{K^b=i\,||\,K^f=j} v \overset{?}{=} v \xleftarrow[g_2]{K^b=i\,||\,K^f=j} C_i,$$

but reuses the computations of chunks, which are defined as parts of g_1 and g_2 that are independent of K^b and K^f, respectively.

This technique was called an *independent-biclique* approach [6] due to use of independent differential trails in the biclique construction.

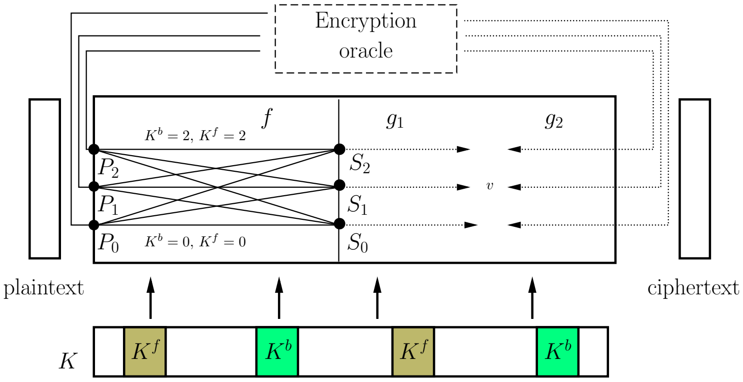


Fig. 2. Key testing with a biclique of three plaintexts and three internal states.

Bicliques based on independent differentials. An easy way to construct biclique is to use related-key differentials that do not share active nonlinear components. Let (P, S, K) be a tuple of a plaintext, an internal state, and a key. Let also K^f and K^b be tuples of key bits.

Proposition 1 ([6]). *Suppose that the tuple* (P_0, S_0, K_0) *conforms to the two sets of related-key differential trails:*

$$0 \xmapsto[f]{\Delta K^f=\Delta_j^K} \Delta_j; \qquad\qquad \nabla_i \xmapsto[f]{\Delta K^b=\nabla_i^K} 0,$$

that share no active non-linear transformations. Then the following states

$$P_i = P_0 \oplus \nabla_i, \qquad\qquad S_j = S_0 \oplus \Delta_j.$$

form a biclique for a group of keys defined by K_0.

Narrow-bicliques. A straightforward application of the independent differentials technique limits the length of a biclique to the number of rounds needed for the full diffusion. In AES one may use truncated differentials with probability 1, and they would still allow for bicliques over the last three rounds. This is virtually impossible for IDEA, as any one-round truncated differential with probability 1 covers the full state and necessitate the full codebook. Therefore, the biclique differentials must be sparse and hence probabilistic.

We propose to amplify the biclique differentials with guess-and-determine and message modification-like techniques, so that even high-dimensional bicliques would not require the full codebook. Similar techniques have been used for the hash function SHA-2 [17], but aimed only for the independency of trails, but not sparsity. In contrast, for block ciphers the sparsity of a trail is a crucial parameter for the data complexity, as uncontrolled difference results into an uncontrolled plaintexts. It gets much worse in a cipher, whose key is much larger than a plaintext, since the number of bicliques to be constructed greatly exceed the codebook size. As a result, with high probability every possible plaintext becomes involved in a biclique, and this situation we want to avoid.

In our attacks on IDEA we employ various techniques and tools to control the biclique differentials and reduce the data complexity. As a validity certificate, we have implemented a large portion of our biclique construction algorithms on a PC. We experimentally verified the amount of freedom we have in the algorithms and our ability to spend that freedom on setting specific plaintext bits to predefined constants.

A large-memory variant. One of the appealing properties of biclique cryptanalysis is the fact that memory requirements are naturally low, only exponential in the dimension of the biclique, which is usually a small constant. In here we show a rather generic way to speed-up biclique key recovery if a very large, but only sequentially accessible memory is available.

The basic idea is that all those computations that do not depend on replies of the plaintext or ciphertext oracle can be stored and reused for multiple key recoveries. When doing this, the computational complexity to recover the first key remains the same, for subsequent key recoveries however, less computations are needed. In Section 6, for 5-round IDEA we give an example where this gives a noticable speed-up.

4 Biryukov-Demirci Relation

The Biryukov-Demirci relation was introduced in [16] as a combination of two observations by Biryukov (unpublished) and Demirci [11]. Consider two consecutive rounds of IDEA and two lines of computations:

$$X_2^i \to X_3^{i+1} \to X_2^{i+2} \text{ and } X_3^i \to X_2^{i+1} \to X_3^{i+2}$$

For these lines we have:

$$\big((X_2^i \boxplus Z_2^i) \oplus (s^i \boxplus t^i)\big) \boxplus Z_3^{i+1} = X_2^{i+2} \oplus t^{i+1}; \tag{A}$$

$$\big((X_3^i \boxplus Z_3^i) \oplus t^i\big) \boxplus Z_2^{i+1} = X_3^{i+2} \oplus (s^{i+1} \boxplus t^{i+1}). \tag{B}$$

For the least significant bit the modular addition resolves into XOR:

$$LSB(X_2^i \oplus Z_2^i \oplus s^i \oplus t^i \oplus Z_3^{i+1} \oplus t^{i+1}) = LSB(X_2^{i+2});$$

$$LSB(X_3^i \oplus Z_3^i \oplus t^i \oplus Z_2^{i+1} \oplus s^{i+1} \oplus t^{i+1}) = LSB(X_3^{i+2}).$$

Let us sum the equations and redistribute the summands:

$$LSB(X_2^i \oplus X_3^i \oplus Z_2^i \oplus Z_3^i \oplus s^i) = LSB(X_2^{i+2} \oplus X_3^{i+2} \oplus s^{i+1} \oplus Z_2^{i+1} \oplus Z_3^{i+1}). \tag{2}$$

Therefore, for the matching in the MITM attack it is enough to compute X_2^i, X_3^i, s^i in the forward direction, and $X_2^{i+2}, X_3^{i+2}, s^{i+1}$ in the backward direction. To compute s^{i+1} it is enough to compute X_1^{i+2}, X_2^{i+2} and know an appropriate subkey Z_5^{i+1}. If some bits of subkeys Z_2^{i+1} and Z_3^{i+1} belong to K^b or K^f, they are distributed to corresponding sides of the equation (this technique named indirect partial matching [1] was applied to hash functions).

The BD-relation essentially excludes six multiplication operations, or about 1.5 rounds, from the matching part.

Improved filtering. We can improve the filtering provided by the BD relation by considering more than one bit in equations (A) and (B). More precisely, we consider X_2^i, X_3^i, s^i, X_2^{i+2}, X_3^{i+2}, s^{i+1}, Z_2^i, Z_3^i, Z_2^{i+1} and Z_3^{i+1} as known parameters, and we denote the left hand side and the right hand of (A) and (B) as $L_A(t^i)$, $R_A(t^{i+1})$, $L_B(t^i)$ and $R_B(t^{i+1})$, respectively.

If we know k bits of t^i, we can compute k bits of $L_A(t^i)$ and $L_B(t^i)$, and $k+1$ bits of $L_A(t^i) \oplus L_B(t^i)$. Similarly, if we know k bits of t^{i+1}, we can compute k bits of $R_A(t^{i+1})$ and $R_B(t^{i+1})$, and $k+1$ bits of $R_A(t^{i+1}) \oplus R_B(t^{i+1})$. As seen in the previous section, some values of the parameters are incompatible with any choice of t^i or t^{i+1}. In order to improve the filtering, we will guess some bits of t^i and t^{i+1} and exclude more parameter choices.

For instance, if we guess one bit of t^i and t^{i+1}, we can compute 1 bit of $L_{A,B}$ and 2 bits of $L_A \oplus L_B$ in the forward direction, and 1 bit of $R_{A,B}$ and 2 bits of $R_A \oplus R_B$ in the backward direction. We put those values a hash table for *every* value of t^i and t^{i+1}, and we look for a match between the forward values and the backward values for *some* value of t^i and t^{i+1}. We can show that there is a match with probability 3/8 which means that we have a filtering of 1.41 bits.

More precisely, for given value of X_2^i, X_3^i, s^i, Z_2^i, Z_3^i, Z_2^{i+1}, Z_3^{i+1} in the forward direction, and X_2^{i+2}, X_3^{i+2}, s^{i+1} in the backward direction, there exists a choice for the first bit of t^i, t^{i+1} that result in a match iff:

$$(L_A \oplus L_B)^{[0]} = (R_A \oplus R_B)^{[0]} \ \textbf{and} \begin{cases} (L_A(0) \oplus L_B(0))^{[1]} = (R_A(\alpha) \oplus R_B(\alpha))^{[1]} \\ \textbf{or} \\ (L_A(1) \oplus L_B(1))^{[1]} = (R_A(\bar{\alpha}) \oplus R_B(\bar{\alpha}))^{[1]} \\ \text{where } \alpha = (L_A \oplus R_A)^{[0]} \end{cases}$$

This shows that the parameters will be compatible with probability $1/2 \times 3/4 = 3/8$, and this has been verified experimentally. We can show in the same way that we have a filtering of roughly 2 bits when guessing 3 bits of t^i and t^{i+1}; in this case we have to evaluate $L_{A,B}$ and $R_{A,B}$ for 8 values of t^i and t^{i+1}, which still costs less than one evaluation of the block cipher (we can evaluate four 4-bit values of $L_{A,B}$ or $R_{A,B}$ in parallel using 16-bit operations). We can have 5 bits of filtering using the full 16 bits of t, but we don't see how to use that efficiently.

5 Key recovery for the full IDEA

Our approach to cryptanalyze the full IDEA is to construct a short biclique of high dimension, and cover the remaining rounds with the independent-biclique approach. To find an optimal configuration of K^f and K^b bits, and also of the matching position, we have run a short search program. First, we figured out that the longest biclique that is still efficient covers 1.5 rounds. Then we computed the maximum chunk length and hence the minimum matching cost. Then we selected for K^f the bits that form long chunks after round 1, and for K^b the bits that form long chunks ending with the ciphertext.

According to the search results, we have chosen the following key partitioning, which results in a biclique of dimension 3:

- K^g (guess): bits $K_{0\ldots40,\,42\ldots47,\,50\ldots124}$.
- K^f (forward): bits $K_{125\ldots127}$.
- K^b (backward): bits $K_{41,48,49}$.

We have also chosen the partition of the full IDEA into a biclique, chunks, and the matching part according to Table 2. It is also illustrated in Figure 3. By the attack algorithm, each chunk is computed 2^3 times per key group, and the operations in the matching part are computed for each key. The Biryukov-Demirci relation (2) serves as internal variable for the matching in rounds 4–6:

$$\underbrace{LSB(X_2^4 \oplus X_3^4 \oplus Z_2^4 \oplus Z_3^4 \oplus s^4)}_{\text{computed forwards}} \overset{?}{=} \underbrace{LSB(X_2^6 \oplus X_3^6 \oplus s^5) \oplus LSB(Z_2^5 \oplus Z_3^5)}_{\text{computed backwards}}.$$

Biclique. A straightforward way to construct a biclique with our key partition would be as follows. Fix K^g and choose arbitrarily a plaintext P_0. For $K^b = 0$ and each value of K^f compute internal states $S_0, S_1, \ldots, S_7$ that are tuples of variables $(Y_1^2, Y_2^2, Y_3^2, Y_4^2)$. Consider S_0 and for $K^f = 0$ and each value of K^b compute plaintexts $P_0, P_1, \ldots, P_7$. Since differentials resulting from the key differences in K^b and K^f do not interleave, these plaintexts and states form a biclique:

$$P_i \xrightarrow[f]{K^b=i\,||\,K^f=j} S_j.$$

However, we do not control the plaintexts $P_1, \ldots, P_7$. Since we construct 2^{122} bicliques, we are likely to cover the full codebook. To reduce the data complexity, we implement a more complicated biclique construction algorithm, which enforces particular plaintext bits to zero in every biclique.

Table 2. Round partition for 8.5-round attack. The cipher is splitted into four parts, whose subkey bits are listed. The parts are a biclique, two chunks (where either K^b or K^f) are not used, and matching (where both K^b and K^f are used). The latter part dominates the complexity.

Round	Z_1 $\odot$	Z_2 $\boxplus$	Z_3 $\boxplus$	Z_4 $\odot$	Z_5 $\odot$	Z_6 $\odot$
Biclique						
1	0–15	16–31	32–47	48–63	64–79	80–95
2	96–111	112–127↓	25–40	41–56↑		
Chunk 1 (K^b not used)						
2					57–72	73–88
3	89–104	105–120	121-8↓	9–24	50–65	66–81
4	82–97	98–113	114–1	2–17	18–33	
Matching						
4						34–49↑
5	75–90	91–106	107–122	123–10	11–26	27–42
6	43–58	59–74	100–115	116–3	4–19	20–35
7	36–51	52–67	68–83	84–99	125–12↓	13–28
Chunk 2 (K^f not used)						
8	29–44	45–60	61–76	77–92	93–108	109–124
9	22–37	38–53↑	54–69	70–85		

The improved algorithm works as follows:

1. Fix K^g, $K^f = K^b = 000$.
2. Choose arbitrarily Y_3^2, Y_4^2, p^1.
   - For each K^b (eight options) compute the output of the MA function;
   - For each K^b (eight options) compute X_2^1 — second word of the plaintext.
   - Check if 5 least significant bits of each X_2^1 are zero (many other bit sets would work as well). If not, choose other Y_3^2, Y_4^2, p^1.
   - Note that this implies that the 5 least significant bits of s^1 and t^1 are the same for all K^b. Therefore the 5 least significant bits of X_3^1 will also be the same for all K^b.
3. Choose a value t with the 5 least significant bits set to zero
   - Use t as X_3^1 with $K^b = 000$, and compute X_1^2 and X_2^2.
   - For each K^b (eight options) compute X_1^1 — first word of the plaintext, from X_1^2 and X_2^2.
   - Check if the least significant bit of each X_1^1 — first word of the plaintext — is zero. If not, choose another t. If all the t's have been tried, choose another Y_3^2, Y_4^2, p^1.
4. Compute other plaintext words for each K^b. Derive plaintexts $P_0, P_1, \ldots, P_7$.
5. Vary K^f and derive internal states $S_0, S_1, \ldots, S_7$.

Therefore, a single biclique can be constructed in 2^{40} time, and 11 plaintext bits (15, 27–31 and 43–47) are set to zero. We notice that key bits 16–25, 32–40, 57–63, 96–124 are neutral for the biclique and can be flipped without violating

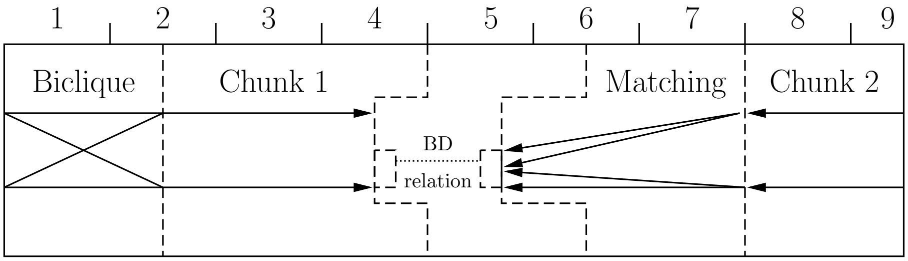


Fig. 3. Attack on the full IDEA (rounds 1–9). Having constructed a biclique, we partially encrypt output states (chunk 1), ask for ciphertexts, partially decrypt them (chunk 2), and match with the help of the BD-relation. The relation allows to ignore about 1.5 rounds of computation.

its plaintext property. Therefore, we can reuse the biclique for 2^{55} key groups and hence make the amortized cost negligible.

The first part of the construction require that there exist a choice of Y_3^2, Y_4^2, p^1 so that the 5 least significant bits of each X_2^1 are zero; this is expected to be the case for a proportion $1-e^{-8} > 99.9$ of the keys. For the full construction, we have a 58-bit condition, which will be satisfied by a proportion $1-e^{-11} \approx 99.99\%$ of the keys. We can also control one more bit of X_1^1, but this leads to a 61-bit conditions, which is satisfied by 95% of the key. Alternatively, we can look for a six bit match in X_2^1 and X_3^1, and a one bit match in X_1^1. This gives control over 13 bits of the plaintext, but it only works for less than $(1-1/e) \approx 63\%$ of the keys. Finally, we can achieve various tradeoffs by changing the dimension of the biclique; we can control:

- 23 bits of a dim.-2 biclique with succ. rate $(1-e^{-4})(1-e^{-5}) > 97.5\%$
- 24 bits of a dim.-2 biclique with succ. rate $(1-e^{-4})(1-e^{-2}) > 84\%$
- 11 bits of a dim.-3 biclique with succ. rate $(1-e^{-8})(1-e^{-11}) > 99.9\%$
- 12 bits of a dim.-3 biclique with succ. rate $(1-e^{-8})(1-e^{-3}) > 94\%$
- 5 bits of a dim.-4 biclique with succ. rate $(1-e^{-16})(1-e^{-14}) > 99.9999\%$

Complexity. Each biclique tests 2^6 keys. The first chunk employs 9 multiplications, the second chunk — 6 multiplications, the matching part — 13 multiplications (and hence 7 when we use the relation, of which two compute only a single output bit and are hence counted as a half multiplication is discussed in Section 2). We recheck on Y_1^5, for which we need 2 multiplications: on Z_6^4 and Z_1^5. A negligible proportion of keys is rechecked on the full state and on another plaintext/ciphertext pair. Therefore, 2^6 keys are tested with

$$9 \cdot 8 + 6 \cdot 8 + (5 + \frac{1}{2} + \frac{1}{2}) \cdot 64 + 2 \cdot 32 = 632 \text{ multiplications } = 2^{4.06}$$

calls of IDEA. The total time complexity is hence $2^{126.06}$.

We can expand K^b to 4 bits for the cost of increased data complexity. As we would be able to spend only 4 degrees of freedom per plaintext, the data complexity becomes 2^{59} and the number of multiplications for 2^7 keys is 1064 which yields a total complexity of about $2^{125.97}$ calls of IDEA, as an easily be computed:

$$9 \cdot 8 + 6 \cdot 16 + (5 + \frac{1}{2} + \frac{1}{2}) \cdot 128 + 2 \cdot 64 = 1064 \text{ multiplications.}$$

6 New 5 round attack

The only 5-round attacks that start at the beginning of the cipher are the following. Biham et al. mention the possibility of an attack with memory complexity 2^{24} and time complexity of 2^{119}. The fastest attack so far needs $2^{115.5}$ time, but also the full codebook. In here we provide the fastest attack which additionally requires only 2^{25} chosen plaintexts.

- K^g (guess): bits $K_{0\ldots8,\,25\ldots49,\,75\ldots127}$.
- K^f (forward): bits $K_{50\ldots74}$.
- K^b (backward): bits $K_{9\ldots24}$.

We can easily construct a biclique of size $2^{25} \times 2^{16}$ in round 1, since the paths are clearly independent: the backward key only affects X_1^1 and X_2^1, while the forward key affects Y_3^1, p^1, q^1 and everything after the MA function. The matching point is p^3 (16 bits), which can be computed in both chunks.

We can control 39 bits of the plaintext in the following way:

- We start with $X_3^1 = 0$, $X_4^1 = 0$, some arbitrary value for Y_1^1, and $Y_2^1 = 2^9 K_{25\ldots31} + \texttt{0x1ff}$
- We can then compute X_1^1 and X_2^1 for each choice of K^b. Note that the high 7 bits of $X_2^1 = Y_2^1 \boxminus K_{16\ldots31}$ will be zeros because there will be no carry in the subtraction.
- Finally we compute $X_1^2, X_2^2, X_3^2, X_4^2$ for each choice of K^f.

The biclique construction has negligible cost, as we can use most of the key bits as neutral.

Alternatively, we can see this attack as a basic MITM if we start with state Y_1^1, Y_2^1, X_3^1, X_4^1. In the forward part, we can compute p^3 independently of K^b. For the backward part we compute X_1^1 and X_2^1, then we query the oracle on that state, and continue the computation from the ciphertext $X_1^1, X_2^1, X_3^1, X_4^1$, up to p^3.

If we flip key bits 112–127, only 2 multiplications in round 2 are affected. Therefore, in the first chunk we need only 3 multiplications to recompute in total. In the second chunk we recompute 7 multiplications. The matching comes for free, but the 2^{25} key candidates must be rechecked on 1 multiplication. Hence the total complexity of testing $2^{128-25-16} = 2^{87}$ key groups is computed as follows:

$$C = 2^{87}\left(\frac{4}{20}2^{25} + \frac{7}{20}2^{16}\right) = 2^{110}.$$

When trying to recover multiple independent keys, and following the large-memory variant outlined in Section 3, the results of 2^{25} forward computations for all 2^{87} bicliques could be precomputed and stored in the form of a table of $2^{25} \cdot 16$ bits (for p^3) for each keygroup together with the plaintext. Hence at the cost of a memory of size equivalent to 2^{110} blocks (of 64 bits each) that needs to be accessed $2^{25} \cdot 2^{87} = 2^{112}$ times in a sequential way for every individual key recovery, the computational cost would drop to about an equivalent of $2^{87}\left(\frac{7}{20}2^{16}\right) = 2^{101.5}$ IDEA calls, as only the backwards computations need to be performed for every keygroup with the respective oracle responses.

Table 3. Round partition for the 5-round attack

Round	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
Biclique						
1	0–15	16–31	32–47	48–63	64–79	80–95
Chunk 1						
2	96–111	112–127	25–40	41–56	57–72	73–88
3	89–104		121-8			
Chunk 2						
4	82–97	98–113			18–33	34–49
5	75–90	91–106	107–122	123–10	11–26	27–42

7 New 6 round attack

The best currently known 6-round attack, the very recent and still unpublished MITM approach of [2] works for 6-rounds only with a single starting position that does not coincide with the actual start of the cipher. In here we give the first 6-round attack.

The key partition is as follows:

- K^g (guess): bits $K_{0\ldots65,\,75\ldots111}$.
- K^f (forward): bits $K_{66\ldots74}$.
- K^b (backward): bits $K_{112\ldots127}$.

We use a one-round biclique at the end of the cipher, in round 6. Then we compute s^2 in the forward direction and s^3 in the backward direction, and use the BD relation.

For the construction of the biclique, we start with $Y_1^6 = 0$, $X_2^6 = -K_{59\ldots74}$, $Y_3^6 = 0$ and $Y_4^6 = 0$. For each K_f, we compute the final state at the end of round 6. We note that we have $X_1^7 = X_2^7$ and the 7 most significant digits of X_3^7 and X_4^7 are also equal. Thus, we control 23 bits of the ciphertext, and the data complexity is 2^{41}.

In order to filter out enough bad guesses, we will use 9 bicliques for each K_g. The most expensive part of the attack is the backward computation of s^3. For each K_g guess, this costs:

$$9 \cdot \left(2^5 + 2^{16} + 2^{16} + \frac{1}{2} \cdot 2^{16}\right) = 2^{20.5} \text{ multiplications} = 2^{15.9} \text{ IDEA calls}$$

The total complexity is therefore $2^{103} \cdot 2^{15.9} = 2^{118.9}$.

8 New 7.5 round attack

In the 7.5-round attack we construct a biclique in the first 1.5 rounds. The key partitioning is defined as follows:

- K^g (guess): bits $K_{0\ldots 24,\,25\ldots 40,\,42\ldots 99,\,125\ldots 127}$.
- K^f (forward): bits $K_{100\ldots 124}$.
- K^b (backward): bits $K_{25,41}$.

The differentials based on K^f and K^b do not interleave in the first 1.5 rounds. Therefore, we can construct a biclique in a straightforward way similar to the full-round attack. However, the differential generated by K^b affects the full plaintext. To reduce the data complexity we construct two bicliques for a key group so that the differential generated by K^b vanishes at the input of the MA-function.

We proceed as follows. For the first biclique we fix $K_{25} \oplus K_{41} = 0$, and for the second one $K_{25} \oplus K_{41} = 1$. As a result, a difference in K^b generates simultaneous differences in Z_3^2 and Z_4^2. Denote the difference in Z_3^2 by ∇ (generated by bit K_{25}), and in Z_4^2 by ∇' (generated by K_{41}). We want the difference in X_4^2 to be equal to ∇ so that the MA-structure have zero input difference (Figure 4). We fulfill this condition by random trials. Bicliques are hence constructed as follows:

1. Fix $X_1^1 = X_2^1 = X_3^1 = 0$;
2. Choose arbitrarily values for X_4^1:
   - Generate internal states for the biclique;
   - Check whether the MA function has zero input difference. If not, try another value of X_4^1.

Computational and data complexity. A single pair of biclique is generated in less than 2^{16} calls of IDEA. This value is amortized since we can derive 2^{19} more bicliques by changing key bits 96–104, (and recomputing Y_1^2), 125–127 (recompute Y_2^2) and bits 57–63 of Z_4^1 (and recomputing the plaintext). Therefore, an amortized cost to construct a biclique is negligible.

Since the ∇-differential affects the most significant bit of X_2^1 only, the plaintexts generated in bicliques have 47 bits fixed to zero. Therefore, the data complexity does not exceed 2^{17}. The full computational complexity of the attack is computed as follows:

$$C = 2^{107}\left(C_{bicl} + 2^{13}C_{chunk1} + 2C_{chunk2} + 2^{13}C_{recheck}\right),$$

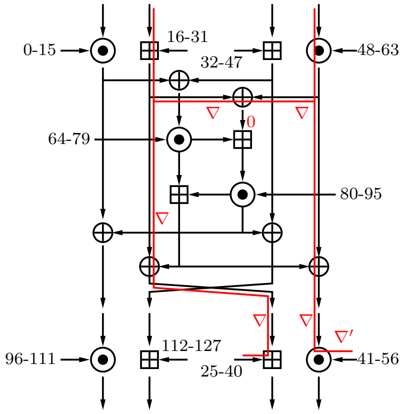


Fig. 4. Biclique ∇-differential in 7.5-round attack.

where we test 2^{106} key groups with two bicliques each. The amortized biclique construction cost is negligible. We note that the multiplication by Z_5^2 in round 2 is also amortized as the change in key bits 57–63 does not affect it. Therefore, the total number of multiplications in the first chunk is $1+4+2 = 7$ multiplications, in the second chunk — $1 + 3 + 4 + 2 = 10$ multiplications, to recheck — 3 multiplications (to compute the full p^5 in both directions). The total complexity is hence $2^{127}\frac{10}{30} = 2^{126.5}$.

We can decrease the time complexity for the cost of the increase in the data complexity. Let us assign one more bit to K^b so that there are 8 values of K^b. We spend 64 bits of freedom in the internal state to fix 13 bits of each biclique plaintext, as shown for the attack on the full IDEA. Then the complexity is estimated as follows:

$$C = 2^{100}\left(2^{25}\frac{8}{30} + 2^{3}\frac{10}{30} + 2^{28}\frac{1}{2}\cdot\frac{2}{30}\right) = 2^{124.1}.$$

We can further reduce the complexity using the improved BD filtering on two bits described in Section 4, which filters out 5/8 of the candidates. First we consider bits 112-113, 105-106, 121-122, 114-115 as part of K^g instead of K^f, so that all the keys involved in the BD relation are part of K^g. We also use some precomputations. For each K^b, we compute s^5, and we evaluate $R_A(t^5)$ and $R_B(t^5)$ for 2 guesses of t^5. Then, we consider the potential candidates from the forward chunk: for each possible 1-bit value of X_2^4, X_3^4 and s^4, plus the second bit of $X_2^4 \oplus X_3^4 \oplus s^4$ we guess 1 bit of t^4 in order to compute $L_A(t^4)$ and $L_B(t^4)$; then we can filter the corresponding candidates for K^b (we expect 3 candidates on average). Then for each K^f, we just use this table to recover the candidates. For the complexity evaluation, we assume that finding a match in the hash table costs the same as one multiplication. This yields a complexity of:

$$C = 2^{108}\left(2^{17}\frac{8}{30} + 2^{3}\frac{10+2+2^{5}}{30} + 2^{20}\frac{3}{8}\cdot\frac{2}{30}\right) = 2^{123.9}.$$

9 On Practical Verification

Especially for the type of cryptanalysis described in this paper where carrying out an attack in full is computationally infeasible, practical verification of attack details and steps is important in order to get confidence in it. To address this, we explicitly state the following:

- We have implemented the dimension-3 biclique construction of Section 5, which works as expected, and takes a few hours on a desktop PC. An example is given in Appendix A.
- We have implemented the improved matching procedure as described in Section 4. We have verified that we have the expected number of remaining candidates.

10 Concluding discussion

We showed that a number of extensions to the recently introduced biclique framework and specific properties of IDEA eventually lead to the cryptanalysis of the full version of the cipher. Though IDEA withstood all cryptanalysis attempts in the last 20 years, it is now vulnerable to key recovery methods that are about 4 times faster than brute force search. We also show attacks on the first 7.5 rounds where the attack algorithm does not have to consider each key separately, resulting in a larger complexity advantage. For smaller number of rounds we surpass the best attacks so far, hence refuting the view that biclique attacks lead only to a small advantage over the brute-force.

We emphasize the use of several techniques from hash function cryptanalysis, which are usually associated with the start-in-the-middle framework. Following the recent work on AES, we demonstrate that these techniques are important also in secret-key cryptanalysis. We foresee widespread application of tools aimed for data complexity reduction, which could be based on our concept of narrow-bicliques.

As a natural application of a new concept we would again name AES. Being able to construct high-dimensional bicliques rather deep in the cipher with reasonable data complexity, an adversary might be able to get a significant advantage over brute-force for the full number of rounds, and possibly even present the best attacks on already broken number of rounds. As meet-in-the-middle attacks might potentially work in $2^{n/2}$ time, it would be extremely interesting to figure out the number of AES rounds that could be broken with this almost practical complexity.

This line of work opens up more questions that we feel are important:

1. Bounds for biclique attacks. With a biclique attack in this paper that is 2^{18} times faster than brute force (or $2^{26.5}$ times faster if large sequentially accessable memory is available) an earlier intuition that this class of attacks only allows for rather small speed-ups over brute force search is dismissed. Nevertheless it may be possible to give meaningful bounds on classes of biclique attacks.
2. How to best defend against biclique cryptanalysis? In this paper we see that even a more conservative key schedule design for IDEA is almost as vulnerable. It seems as if only very expensive key schedule designs, i.e. those where the key can not easily be deduced from subkey material, would provide resistance. This remains a topic of future work, though.

Acknowledgements

We thank Orr Dunkelman and Adi Shamir for bringing to our attention their new attack on 6 middle rounds of IDEA during the MSR Symmetric Cryptanalysis Workshop 2011. This was the starting point for our investigations and new results. We also thank Florian Mendel and Andrey Bogdanov for the discussions on possible research directions. Finally, we thank reviewers of Eurocrypt 2012 for their helpful comments.

Part of this work was done while Christian Rechberger was with ENS Ulm and Foundation Chaire France Telecom, and visiting MSR Redmond. Gaëtan Leurent is supported by the AFR grant PDR-10-022 of the FNR Luxembourg. This work was supported in part by the European Commission under contract ICT-2007-216646 ECRYPT NoE phase II.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for step-reduced SHA-2. In: ASIACRYPT'09. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 578–597
2. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New data-efficient attacks on 6-round IDEA. Cryptology ePrint Archive, Report 2011/417 (2011) http://eprint.iacr.org/.
3. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New data-efficient attacks on reduced-round idea. Cryptology ePrint Archive, Report 2011/417 (2011) http://eprint.iacr.org/.
4. Biryukov, A., Jr., J.N., Preneel, B., Vandewalle, J.: New Weak-Key Classes of IDEA. In Deng, R.H., Qing, S., Bao, F., Zhou, J., eds.: ICICS'02. Volume 2513 of Lecture Notes in Computer Science., Springer (2002) 315–326
5. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: ASIACRYPT'09. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 1–18
6. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 344–371

7. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In: Selected Areas in Cryptography'10. Volume 6544 of Lecture Notes in Computer Science., Springer (2010) 229–240
8. Bouillaguet, C., Derbez, P., Fouque, P.A.: Automatic search of attacks on round-reduced AES and applications. In: CRYPTO'11. Volume 6841 of Lecture Notes in Computer Science., Springer (2011) 169–187
9. Chaum, D., Evertse, J.H.: Crytanalysis of DES with a reduced number of rounds: Sequences of linear factors in block ciphers. In: CRYPTO'85. Volume 218 of Lecture Notes in Computer Science., Springer (1985) 192–211
10. Daemen, J., Govaerts, R., Vandewalle, J.: Weak Keys for IDEA. In Stinson, D.R., ed.: CRYPTO'93. Volume 773 of Lecture Notes in Computer Science., Springer (1993) 224–231
11. Demirci, H.: Square-like attacks on reduced rounds of IDEA. In: Selected Areas in Cryptography'02. Volume 2595 of Lecture Notes in Computer Science., Springer (2002) 147–159
12. Diffie, W., Hellman, M.: Special feature exhaustive cryptanalysis of the NBS Data Encryption Standard. Computer **10** (1977) 74–84
13. Dunkelman, O., Sekar, G., Preneel, B.: Improved meet-in-the-middle attacks on reduced-round DES. In: INDOCRYPT'07. Volume 4859 of Lecture Notes in Computer Science., Springer (2007) 86–100
14. Hawkes, P.: Differential-Linear Weak Key Classes of IDEA. In: EUROCRYPT'98. (1998) 112–126
15. Isobe, T.: A single-key attack on the full GOST block cipher. In: FSE'11. Volume 6733 of Lecture Notes in Computer Science., Springer (2011) 290–305
16. Jr., J.N., Preneel, B., Vandewalle, J.: The Biryukov-Demirci attack on reduced-round versions of IDEA and MESH ciphers. In: ACISP'04. Volume 3108 of Lecture Notes in Computer Science., Springer (2004) 98–109
17. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family. Available online at http://eprint.iacr.org/2011/286.pdf (2011)
18. Lai, X., Massey, J.L.: Markov ciphers and differential cryptanalysis. In: EUROCRYPT. (1991) 17–38
19. Lai, X., Massey, J.L.: Hash function based on block ciphers. In: EUROCRYPT. (1992) 55–70
20. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: INDOCRYPT'10. Volume 6498 of Lecture Notes in Computer Science., Springer (2010) 282–291
21. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT'93. Volume 765 of Lecture Notes in Computer Science., Springer (1993) 386–397
22. Rijmen, V., Daemen, J., Preneel, B., Bosselaers, A., Win, E.D.: The cipher SHARK. In: FSE'96. Volume 1039 of Lecture Notes in Computer Science., Springer (1996) 99–111
23. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: EUROCRYPT'09. Volume 5479 of Lecture Notes in Computer Science., Springer (2009) 134–152
24. Sun, X., Lai, X.: The Key-Dependent Attack on Block Ciphers. In Matsui, M., ed.: ASIACRYPT'09. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 19–36
25. Wegener, I., Woelfel, P.: New results on the complexity of the middle bit of multiplication. Computational Complexity **16**(3) (2007) 298–323

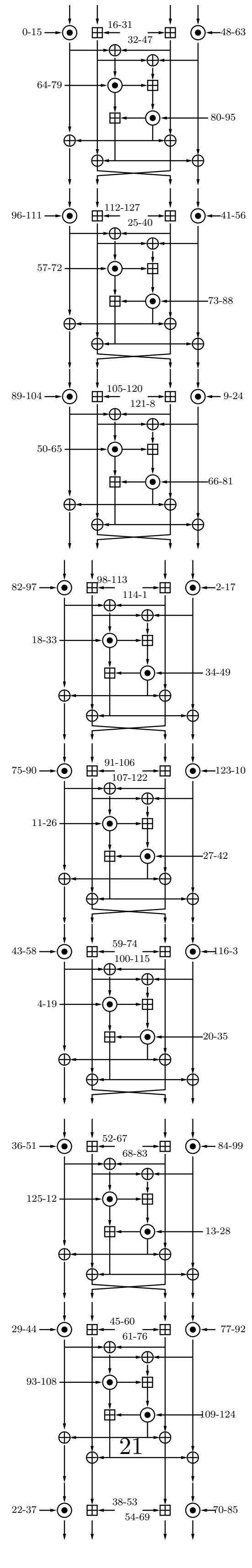
A Biclique example

We give an example of a dimension-3 biclique for the first 1.5 round of IDEA. It is built with the bits of K^g set to the key `0x0102030405060708090a0b0c0d0e0f10`, and the bits used for K^f and K^b are $41, 48, 49$ and $116, 117, 118$, respectively. Each plaintext has 11 bits set to zero as explained in Section 5.

P_0	1754 5580 00c0 d05b	S_0	7092 7352 f5b1 7272
P_1	0f10 ca00 a440 aa79	S_1	7092 7152 f5b1 7272
P_2	bda8 f580 a0a0 c6b7	S_2	7092 6f52 f5b1 7272
P_3	17c4 86a0 6f00 6c69	S_3	7092 6d52 f5b1 7272
P_4	e9fe 6500 5100 143a	S_4	7092 6b52 f5b1 7272
P_5	9252 0200 ec00 230c	S_5	7092 6952 f5b1 7272
P_6	aa8e b5a0 5fc0 16ef	S_6	7092 6752 f5b1 7272
P_7	4a9e c520 b040 ecc0	S_7	7092 6552 f5b1 7272

A schematic aid for verifiying our attacks.

Supplementary material: Full IDEA scheme


0-15
16-31
48-63
32-47
64-79
80-95
96-111
112-127
41-56
25-40
57-72
73-88
89-104
105-120
9-24
121-8
50-65
66-81
82-97
98-113
2-17
114-1
18-33
34-49
75-90
91-106
123-10
107-122
11-26
27-42
43-58
59-74
116-3
100-115
4-19
20-35
36-51
52-67
84-99
68-83
125-12
13-28
29-44
45-60
77-92
61-76
93-108
109-124
22-37
38-53
70-85
54-69