Tuple cryptanalysis of ARX with application to BLAKE and Skein

Jean-Philippe Aumasson[1], Gaëtan Leurent[2], Willi Meier[3,*], Florian Mendel[4], Nicky Mouha[5,6,†], Raphael C.-W. Phan[7], Yu Sasaki[8], and Petr Susil[9]

[1] Nagravision, Switzerland
[2] University of Luxembourg
[3] FHNW Windisch, Switzerland
[4] Graz University of Technology, Austria
[5] ESAT/COSIC, Katholieke Universiteit Leuven, Belgium
[6] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium
[7] Loughborough University, UK
[8] NTT, Japan
[9] EPFL, Switzerland

Abstract. We introduce tuple cryptanalysis, a variant of structural cryptanalysis techniques as square, saturation, integral, internal collision, or multiset cryptanalysis, the main difference being that tuple cryptanalysis considers ordered rather than unordered multisets. This allows cryptanalysts to better trace structural properties within a cipher's internal state. Unlike previous works that focus on S-box based algorithms, structural analysis is applied to ARX constructions, with preliminary results on reduced versions of Skein's and BLAKE's ARX cores. Due to its simplicity and efficient verification, tuple cryptanalyis can be used as a security benchmark for ARX schemes.

Keywords: hash functions, cryptanalysis, BLAKE, Skein

1 Introduction

We introduce *tuple cryptanalysis*, a technique inspired by and which extends the previous notions of square [1], saturation [2], integral [3], internal collision [4], and multiset cryptanalysis [5], notably the latter proposed in the Biryukov/Shamir attacks on the SASAS construction.

The core object that we utilize is a *tuple*, i.e. an ordered list of possibly repeating elements. Thus, a tuple differs from an ordered set because of element repetition, and differs from a multiset in the sense that the elements have a prescribed internal ordering within the tuple. Tuple cryptanalysis can be seen as a generalization of multiset cryptanalysis—if one disregards the ordering, a tuple reduces to a multiset.

2 Tuple properties

2.1 Definitions and notations

We consider tuples of 2^w elements where each element is a w-bit word. A tuple T can be viewed, by abuse of convention, as an ordered multiset, thus elements may share the same value from an underlying set $\{0,1\}^w$. Therefore, to each unique value in $\{0,1\}^w$ one can associate a multiplicity corresponding to how many tuple elements have that value. A tuple is said to have the following properties[10]:

[*] Supported by GEBERT RÜF STIFTUNG under project number GRS-069/07.

[†] Funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)

[10] $\mathsf{C}, \mathsf{P}, \mathsf{B}, \mathsf{X}$ have respectively been named "passive"/"constant", "active"/"saturated"/"permutation", "balanced", and "garbled" in the literature.

– C: elements equal a constant value $c \in \{0,1\}^w$, thus the element value c has multiplicity 2^w and other values have null multiplicities.
– P: elements are all distinct, i.e. the tuple defines a permutation, so each element has multiplicity 1.
– E: all element values have even multiplicity (possibly zero).
– A: the tuple is ADD-balanced, i.e. the elements' ADD sum[11] is zero (when counting their multiplicities).
– B: the tuple is XOR-balanced, i.e. the elements' XOR sum is zero (when counting their multiplicities).
– F: the elements' ADD sum is 2^{w-1}.

Furthermore, we write X when the tuple satisfies none of the above properties, *or when it hasn't been identified as having any of them*; for example, analysis of a specific ARX chain as Threefish-256's may determine a tuple to be X as the result of $\mathsf{P} \oplus \mathsf{P}$, while experiments reveal that the said P tuples are in fact ordered in such a way that their XOR is actually P as well (a concrete example is found in Fig. 1).

Note that one can split the properties in two categories:

– Those characterized by the multiplicity of the elements (C, P, E).
– Those characterized by relations between the elements (A, B, F).

If $T = (T_0, T_1, \ldots, T_{2^w-1})$ and $S = (S_0, S_1, \ldots, S_{2^w-1})$ are two tuples, then $T + S$ is the tuple defined as

$$T + S := (T_0 + S_0, T_1 + S_1, \ldots, T_{2^w-1} + S_{2^w-1}) \ .$$

If the tuple T satisfies the C property, we write $\mathsf{C}(T)$. Similar notations are used for other properties. We use the notation (say) $\mathsf{C}^5\mathsf{PC}^2$ to denote a 8-word tuple decomposed into word tuples with property CCCCCPCC, in this order. More precisely, $\mathsf{C}^5\mathsf{PC}^2$ is the class of all 8-word tuples $(T^0, \ldots, T^7)$ that satisfy $\mathsf{C}(T^0)$, $\mathsf{C}(T^1)$, $\ldots$, $\mathsf{C}(T^4)$, $\mathsf{P}(T^5)$, $\mathsf{C}(T^6)$, $\mathsf{C}(T^7)$.

2.2 Properties relations and transformations

We give a non-exhaustive list of relations between the aforementioned properties, along with rules on how properties are transformed through various operators.

General rules. Before describing ARX-specific results, we recall general rules as given in [5]. The results below consider an additional property D, defined as the "dual" property of a tuple that is either P or E. The results are stated for multisets, but apply as well to tuples.

Lemma 1 (Biryukov/Shamir).

1. *Any multiset with either property* E *or property* P *(when $w > 1$) also has property* B.
2. *The* E *and* C *properties are preserved by arbitrary functions over w-bit values.*
3. *The* P *property is preserved by arbitrary bijective functions over w-bit values.*
4. *The* B *property is preserved by an arbitrary linear mapping from w bits to n bits when $w > 1$. It is preserved by arbitrary affine mappings when the size of the multiset is even.*

Lemma 2 (Biryukov/Shamir).

1. *Property $\mathsf{C}^{i-1}\mathsf{PC}^{k-i}$ is preserved by a layer of arbitrary S-boxes provided that the ith S-box is bijective.*
2. *Property D^k is preserved by a layer of bijective S-boxes.*
3. *Property D^k is transformed into B^k by an arbitrary linear mapping on n bits and by an arbitrary affine mapping when the size of the multiset is even.*
4. *Property $\mathsf{C}^{i-1}\mathsf{PC}^{k-i}$ is transformed into property D^k by an arbitrary affine mapping when the size of the multiset is even.*

In the above statements, "bijective function" covers e.g. addition or XOR with a constant, and "linear mapping" and "affine mapping" cover rotation by a fixed constant.

[11] As unsigned integers modulo 2^w.

Transformations through ARX. The following logical relations are straightforward, for all tuples T and S:

$$\mathsf{C}(T) \Rightarrow \mathsf{B}(T)$$
$$\mathsf{C}(T) \wedge \mathsf{P}(S) \Rightarrow \mathsf{P}(T+S) \ ;$$

for succinctness we just write them as

$$\mathsf{C} \Rightarrow \mathsf{B}$$
$$\mathsf{C} + \mathsf{P} = \mathsf{P}$$

General truth tables for ADD, XOR, and ROT are given in Table 1. All properties but F and A are invariant through rotation, i.e. $\mathsf{C} \ggg n = \mathsf{C}$, $\mathsf{P} \ggg n = \mathsf{P}$, $\mathsf{E} \ggg n = \mathsf{E}$, $\mathsf{B} \ggg n = \mathsf{B}$. The A property is not ROT-invariant but only MSB-shift-invariant, i.e. $\mathsf{A} \ll n = \mathsf{A}$, because MSB's then do not affect LSB's.

Most results in the ADD and XOR truth tables are trivial. Let us show why the relation $\mathsf{P}+\mathsf{P} = \mathsf{A}$ holds: it is deduced from the observation that $\mathsf{P} \Rightarrow \mathsf{F}$, i.e. the sum of all elements from two P tuples is

$$\sum_{i=0}^{2^w-1} i + \sum_{i=0}^{2^w-1} i = 2^{w-1} + 2^{w-1} \equiv 0 \bmod 2^w \ ,$$

which defines the A property. Observe that we must then have

$$\mathsf{P} + \mathsf{P} \neq \mathsf{P} \ ,$$

that is, the ADD-sum of two P tuples is *never* a permutation. Note however that

$$\mathsf{P} \oplus \mathsf{P} \not\Rightarrow \neg\mathsf{P} \ ,$$

i.e. the XOR of two permutations can be a permutation. A counter example for $w = 2$ is given by $\{0,1,2,3\} \oplus \{3,0,2,1\} = \{3,1,0,2\}$. It is easy to see that a sufficient condition for satisfying $\mathsf{P} \oplus \mathsf{P}$ is that there exists a bijective function f such that the two P tuples T and S satisfy $T_i = S_i \oplus f(S_i)$.

Table 1. Truth tables of ADD, XOR, and ROT over tuples.

$+$	A	B	C	E	F	P
A	A	X	A	X	F	F
B	X	X	X	X	X	X
C	A	X	X	E	F	P
E	X	X	E	X	X	X
F	F	X	F	X	A	A
P	F	X	P	X	A	A

$\oplus$	A	B	C	E	F	P
A	X	X	X	X	X	X
B	X	B	B	B	X	B
C	X	B	C	E	X	P
E	X	B	B	B	X	B
F	X	X	X	X	X	X
P	X	B	P	B	X	B

$\ggg$	A	B	C	E	F	P
n	X	B	C	E	X	P

Ordering-dependent transformations. More specific properties relying on the ordering of elements within a tuple (relatively to another tuple), not captured by truth tables, can be exploited to analyse ARX constructions. For example:

- If T is a P tuple and S is such that $S_i = -T_i$, then $\mathsf{C}(T + S)$; if T and S are such that $S_i = T_i \oplus c$ for some constant c, then $\mathsf{C}(T \oplus S)$.
- If T is a P tuple and S is such that $S_i = -T_i$ for all i's, then $\mathsf{E}(T \oplus S)$. This property follows from the fact that for each $i = 0, \ldots, 2^w - 1$, there exists a unique j for which $T_i = -T_j$, i.e. we have

$$T_i \oplus S_i = T_i \oplus (-T_i) = T_j \oplus (-T_j) \ .$$

  Therefore, each ith of T element is paired up with a jth element such that $T_i \oplus S_i = T_j \oplus S_j$. In particular, zero and $2^w - 1$ satisfy $T_i = -T_i$, thus zero has multiplicity 2 in $T \oplus C$. Note that writing $\mathsf{P} \oplus (-\mathsf{P}) = \mathsf{E}$ is incorrect, for the two permutations need have properly related orderings, which that notation does not express.
- If T and S are P tuples such that for all i's, $T_i = S_i \ggg r$, for some fixed r, then $T \oplus S$ does not satisfy property P. This can be seen by observing that the zero element will appear at least twice, due to the elements 0 and $2^w - 1$ that are both rotation-invariant and XOR-doubling to zero. However, observe that, given any solution x to the equation $x \oplus (x \ggg r) = v$, a second solution can be obtained with $x \oplus (2^w - 1)$. Thus, that equation has an even number of solutions, i.e. elements of $T \oplus S$ will all have even multiplicity, i.e. we have $\mathsf{E}(T \oplus S)$.
- If T, S, and R are two P tuples sharing a same ordering, then $\mathsf{P}(T + S + R)$. More generally, we have $\mathsf{P}(T + 2nT)$ where n is an integer, and $2nT$ denotes the sum of $2n$ copies of T. This property follows from the fact that the tuple $T + 2nT$ has elements $((2n+1)T_0, (2n+1)T_1, \ldots, (2n+1)T_{2^w-1})$, which by definition are all distinct (because $2n + 1$ and 2^w are coprime, which implies that $(2n + 1)x = (2n + 1)y$ if and only if $x = y$).

These properties will prove useful when searching for optimal choices of tuples within a cipher's structure.

2.3 Characterizing tuples at the bit level

The ordering of a tuple's elements can be characterized by the structure of its bit representation. Here, we adapt the notations of Z'aba et al. [6] to describe different kinds of bit tuples obtained by bit slicing a w-bit tuple.

- a_i: this bit tuple has its bit elements ordered as the alternation of contiguous segments of 1s (resp. 0s), where each segment length is 2^i; e.g. $(0, 1, 0, 1, 0, 1, 0, 1)$ and $(1, 0, 1, 0, 1, 0, 1, 0)$ are the two a_0 bit tuples that may be obtained by bit slicing a 3-bit word tuple P where elements are canonically ordered in unit increments from 0 to $2^3 - 1$; and extracting only the least significant bit of all its 8 elements. Note that as a by product of its definition, an a_0 bit tuple is E and therefore B. In fact, it is worth noting here that for a bit tuple, the properties E and B are equivalent, so henceforth for bit tuples we will use them interchangibly.
- b_i: this bit tuple has its bit elements ordered as either once-repeating or alternating segments of contiguous 1s (resp. 0s), where each segment length is 2^i; e.g. $(1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0)$ is a b_1 bit tuple. By definition, b_i for $i > 0$ is E; while b_0 is not interesting as any bitstring can be represented by b_0. Instead, we are interested only in some structured subsets for which b_0 may be E; e.g. the b_0 obtained as a sum of two or more a_i is E, in fact it has an equal number of 1s and 0s.

2.4 Implementation and verification

Given a chain of ARX operations over w-bit words, the transformation of tuple properties is independent of the word size w. This allows one to verify tuple properties of the 64-bit word Threefish algorithm using reduced versions with 8-bit words. Rather than hardcoding the transformation rules identified by analysis, a computer program can individually verify the tuple property of each word of the internal state. This makes empirical observations independent from analytical results, making the former more reliable, as they are immune to analytical errors. Our C programs tracing tuple properties in the internal states of Skein and BLAKE are available on demand.

Divergence between results on reduced-word versions and concrete 32- or 64-bit word algorithm may be due to artefacts coming from the choice of rotation values. Also, one should experiment with random rather than fixed constants to avoid errors due to false positives. Any observations should thus be thoroughly investigated before deducing general rules from it.

3 Application to Threefish

Threefish is the block cipher at the core of the SHA3 candidate Skein. The three instances Threefish-256, -512, and -1024 all work on 64-bit words and rely on the 2-word permutation **MIX** defined as

$$\mathbf{MIX}(x, y) := (x + y, (x + y) \oplus (y \ggg r))$$

where r is some context-dependent rotation constant.

3.1 Transformations through MIX

MIX transforms a tuple CP to at least PB:

$$\mathbf{MIX}(\mathsf{C}, \mathsf{P}) = (\mathsf{C} + \mathsf{P}, (\mathsf{C} + \mathsf{P}) \oplus (\mathsf{P} \ggg r)) = (\mathsf{P}, \mathsf{P} \oplus \mathsf{P}) = (\mathsf{P}, \mathsf{B}) \ ,$$

i.e. one may get a more structured tuple property than B, as discussed later. Another example is given with

$$\mathbf{MIX}(\mathsf{P}, \mathsf{P}) = (\mathsf{P} + \mathsf{P}, (\mathsf{P} + \mathsf{P}) \oplus (\mathsf{P} \ggg r)) = (\mathsf{A}, \mathsf{X}) \ .$$

By making assumptions on the P tuples, one can get more specific results. For example, if the tuples are chosen such that each of the two i's elements are the additive inverse of each other, then one obtains (C, P) as output . The choice of the constant in C can also affect the results obtained: for example, if C is zero then CB is transformed to BB, instead of XX in the general case. This is because ADD then behaves as XOR, allowing the B property to propagate through the **MIX** addition.

Table 2 reports a (non-exhaustive) list of worst case tuple transformations through **MIX** and $\mathbf{MIX}^{-1}$. Unlike in Table 1, the truth table's diagonal is not a symmetry axis.

Table 2. Transformations of tuple properties through **MIX** and $\mathbf{MIX}^{-1}$. Lines list first arguments to **MIX**, for example CP maps to PB while PC maps to PP .

MIX	A	B	C	E	F	P
A	AX	XX	AX	XX	FX	FX
B	XX	XX	XX	XX	XX	XX
C	AX	XX	CC	EB	FX	PB
E	XX	XX	EE	XX	XX	XX
F	FX	XX	FX	XX	AX	AX
P	FX	XX	PP	XX	AX	AX

$\mathbf{MIX}^{-1}$	A	B	C	E	F	P
A	XX	XX	XX	XX	XX	XX
B	XX	XB	XB	XB	XX	XB
C	XX	XB	CC	EE	XX	PP
E	XX	XB	XE	XX	XX	XB
F	XX	XX	XX	XX	XX	XX
P	XX	XB	AP	XB	XX	XB

3.2 Transformation through Threefish rounds

We considered the Threefish ARX chain and compare the properties determined by mere analysis with those empirically obtained for a specific choice of a constant. Empirical results are more precise—i.e. stronger as far as cryptanalysis is concerned—than theoretical predictions relying only general ARX rules, because of dependencies between the tuples of internal words, difficult to track analytically. Indeed, better results are achieved by tracking the actual tuples rather than only their properties, for the latter hide structural properties exploitable in the construction of distinguishers.

To refine the analysis, one should thus track the evolution of tuples rather than of their properties. Manually tracking the evolution of tuples allowed us to (partially) reconstruct the properties observed.

1	PC	CC
	PP	CC
2	PC	CP
	PP	PB
3	PB	PP
	AX	AX
4	AX	AX
	XX	XX
5	AX	AX
	XX	XX

1	PC	CC
	PP	CC
2	PC	CP
	PP	PE
3	PB	PP
	BB	EP
4	BP	EB
	BB	FA
5	BA	FB
	FX	BB

Fig. 1. Evolution of tuple properties in Threefish-256's ARX chain: (left) analysis; (right) case of null constants and rounds keys. For each round, we show the input and the output of each **MIX**.

3.3 Distinguishing attacks

We construct distinguishers that consist in the efficient finding of a tuple of 2^w inputs, i.e. Threefish plaintexts, satisfying some tuple properties in such a way that the output, i.e. ciphertext, also satisfies some unexpected tuple properties. Note that a random tuple is A (resp. B, F) with probability 2^{-w}. Unlike most differential distinguishers, tuple properties lead to deterministic distinguishers.

Our "inside-out" strategy is simple and similar to that of zero-sum attacks [7–10]: we choose a tuple of 2^{64} internal states with a given property, e.g. being a CPC^6 tuple, then we compute the round function backwards and forwards until just before some tuple property is no longer observed.

In general, better initial tuple properties are those with the strongest non-trivial local structure (i.e. P) and with only one active word; the property will then diffuse and lose its structure, e.g. progressively downgrading to B and then to X. Optimized initial tuple structures will consist of a set of P's related in such a way that ordering-dependent transformations (see §§2.2) can be exploited to reduce diffusion.

Known-key attack. In this setting the key and the tweak are picked uniformly at random, and remain fixed through all 2^w evaluations of the cipher; i.e. each element of the tuple is processed with the same round keys. The actual value of the key and tweak are not used, but one needs access to the internal state after four rounds (which may be realized by introducing precise faults in a given register).

Fig. 2 presents

1. the evolution of tuple properties as predicted by analysis only relying on the nature of the properties, not on the actual tuples and their dependencies;
2. the properties typically observed with a random key and random C constants;
3. the properties observed with null rounds keys and constants.

In these three cases, tuple properties are traceable on up to 7, 8, and 9 rounds respectively. The analysis holds for any word size, thus directly applies to the (64-bit) Threefish-512; empirical observations may include artefacts of the reduced-word implementation (e.g. due to false alarms, choice of rotation constants). Nevertheless, the difference of results between mere analysis of properties and actual observations reveals that interdependencies between tuples provide additional structure to the output. A more detailed analysis (tracing the structure of tuples) would thus reveal more properties.

Fig. 3 is similar to Fig. 2 but for Threefish-1024: tuple properties are observed on up to 9, 10, and 12 rounds, respectively.

Chosen-keys attack (sketch). The strategy in this scenario is choosing several words in a subkey so that the value after the addition of a internal state and the subkey can be constant. In this section, we explain the attack on Threefish-512 as an example. For example, assume that the internal state and the key just before the key addition has the form CPC^6, such that the P tuple is respectively T and $-T + C$ in the state and in the key, where C is some C tuple. The results of the addition of these values always become a C^8 tuple.

0	XX	XX	PP	AP
1	CC	CC	PP	XX
2	CC	CC	AP	CC
3	CC	PC	CC	CC
4	PC	CC	CC	CP
5	CP	CB	PC	PC
6	XB	PB	PP	PX
7	XX	AX	XX	XX

0	BA	XX	PP	AP
1	CC	CC	PP	XX
2	CC	CC	BP	CC
3	CC	PC	CC	CC
4	PC	CC	CC	CP
5	CP	CB	PC	PC
6	FB	PP	PP	PX
7	EX	EX	XB	AB
8	XX	XX	FX	XX

0	XX	XX	PP	AP
1	CC	CC	PP	XX
2	CC	CC	AP	CC
3	CC	PC	CC	CC
4	PC	CC	CC	CP
5	CP	CE	PC	PC
6	EE	PP	PP	PE
7	EE	EB	BE	EE
8	BX	FX	FA	AB
9	XX	FX	XX	XX

Fig. 2. Evolution of tuple properties in Threefish-512's ARX chain: (left) analysis; (center) example with random key and C constants; (right) case of null constants and round keys.

0	XX	AP	PP	XX	XX	XX	PP	XX
1	CC	PP	XX	CC	CC	CC	AP	XX
2	CC	CC	CC	XX	CC	PP	CC	CC
3	CC	CC	AP	CC	CC	CC	CC	CC
4	CC	CC	CC	PC	CC	CC	CC	CC
5	CC	CC	PC	CC	CP	CC	CC	CC
6	CB	CC	CC	PC	CC	CC	CP	PC
7	XC	CB	PC	CP	CP	PC	PC	CX
8	XB	XP	PP	PX	PB	PX	XP	PX
9	XX	XX	XX	AX	XX	XX	XX	XX

0	XX	AP	PP	XX	XX	XX	PP	XX
1	CC	PP	XX	CC	CC	CC	AP	XX
2	CC	CC	CC	XX	CC	PP	CC	CC
3	CC	CC	AP	CC	CC	CC	CC	CC
4	CC	CC	CC	PC	CC	CC	CC	CC
5	CC	CC	PC	CC	CP	CC	CC	CC
6	CB	CC	CC	PC	CC	CC	CP	PC
7	FC	CB	PC	CP	CP	PC	PC	CX
8	FF	BP	PP	PX	PF	PF	XP	PF
9	AX	BB	XX	EX	AX	BB	BB	BX
10	XX	XX	XX	XX	XX	FX	XX	XX

0	XX	AP	PP	XX	XX	XX	PP	XX
1	CC	PP	XX	CC	CC	CC	AP	XX
2	CC	CC	CC	XX	CC	PP	CC	CC
3	CC	CC	AP	CC	CC	CC	CC	CC
4	CC	CC	CC	PC	CC	CC	CC	CC
5	CC	CC	PC	CC	CP	CC	CC	CC
6	CE	CC	CC	PC	CC	CC	CP	PC
7	EC	CE	PC	CP	CP	PC	PC	CE
8	EE	EP	PP	PE	PE	PE	EP	PE
9	EB	BB	BA	EB	AB	BB	BE	BE
10	FA	BB	FX	AX	FF	BB	FX	FX
11	FX	AX	XX	XX	XX	BA	XX	AX
12	XX	BX	XX	XX	XX	XX	BX	XX

Fig. 3. Evolution of tuple properties in Threefish-1024's ARX chain: (top-left) analysis; (top-right) example with random key and C constants; (bottom) case of null constants and round keys.

Obviously, the constant state continues until the next subkey with the form $-T + C$ appears and thus the attack can be extended by at least 4 rounds because Threefish only inserts a subkey in every 4 rounds.

The analysis in [11] proposed the key difference that achieves the null difference in one subkey k_2. Details of the key difference is shown in [11, Table 2], which is also described in Table 3. Based on this difference for a pair of (key, tweak)s, 2^{64} (key, tweak)s forming $\mathsf{C}^4\mathsf{P}\mathsf{C}^3$ state for sub-key k_3 can be constructed. Let us assume that such 2^{64} (key, tweak)s exist. Then, zero-difference and Δ-difference in Table 3 are replaced with C and P for the tuple analysis, respectively. Namely, the property of 2^{64}-set for each subkey can be described as follows;

$$
\begin{aligned}
k_0 &= \mathsf{C}^5\mathsf{P}\mathsf{C}\mathsf{P},\\
k_1 &= \mathsf{C}^7\mathsf{P},\\
k_2 &= \mathsf{C}^8,\\
k_3 &= \mathsf{C}^4\mathsf{P}\mathsf{C}^3,\\
k_4 &= \mathsf{C}^3\mathsf{P}^2\mathsf{C}\mathsf{P}\mathsf{C},\\
k_5 &= \mathsf{C}^2\mathsf{P}^2\mathsf{C}\mathsf{P}^2\mathsf{C}.
\end{aligned}
$$

k_2 never breaks the property of C^8 which are generated by the addition of the internal state and k_3. Hence, C^8 state continues for eight rounds and the attack becomes more efficient.

Table 3. Details of the subkeys and of their differences, given a difference Δ in k_7 and t_0 (leading to Δ differences in k_8 and t_2).

s	d	$k_{s,0}$	$k_{s,1}$	$k_{s,2}$	$k_{s,3}$	$k_{s,4}$	$k_{s,5}$	$k_{s,6}$	$k_{s,7}$
		Differences							
0	0	k_0 0	k_1 0	k_2 0	k_3 0	k_4 0	$k_5 + t_0$ Δ	$k_6 + t_1$ 0	k_7 Δ
1	4	k_1 0	k_2 0	k_3 0	k_4 0	k_5 0	$k_6 + t_1$ 0	$k_7 + t_2$ 0	$k_8 + 1$ Δ
2	8	k_2 0	k_3 0	k_4 0	k_5 0	k_6 0	$k_7 + t_2$ 0	$k_8 + t_0$ 0	$k_0 + 2$ 0
3	12	k_3 0	k_4 0	k_5 0	k_6 0	k_7 Δ	$k_8 + t_0$ 0	$k_0 + t_1$ 0	$k_1 + 3$ 0
4	16	k_4 0	k_5 0	k_6 0	k_7 Δ	k_8 Δ	$k_0 + t_1$ 0	$k_1 + t_2$ Δ	$k_2 + 4$ 0
5	20	k_5 0	k_6 0	k_7 Δ	k_8 Δ	k_0 0	$k_1 + t_2$ Δ	$k_2 + t_0$ Δ	$k_3 + 5$ 0
6	24	k_6 0	k_7 Δ	k_8 Δ	k_0 0	k_1 0	$k_2 + t_0$ Δ	$k_3 + t_1$ 0	$k_4 + 6$ 0

We now explain how to compute 2^{64} (key, tweak)s with considering all the details. In Table 3, we need to make sure that the differences in k_7 and t_2 cancel each other by $k_7 + t_2$ (modular addition), and at the same time, the differences in k_8 and t_2 cancel each other by $k_8 + t_0$ (modular addition), while k_8 is computed by $C_{240} \oplus \bigoplus_{i=0}^{7} k_i$ (exclusive OR) and t_2 is computed by $t_0 \oplus t_1$ (exclusive OR).

Previous work [11] only considers the difference in the most significant bit, and thus, the use of two different operations does not cause any problem. However, the tuple attack needs to make sure that the differences always cancel each other for any Δ, and we thus need careful analysis.

The generating procedure is as follows. Fix $k_0 \cdots k_6$ to a constant value which satisfies $C_{240} \oplus \bigoplus_{i=0}^{6} k_i = 0$. Let k_7 take all the 2^{64} values in $\{0 \cdots 2^{64} - 1\}$. This makes $k_8 = k_7$ for each k_7. We then set $t_0 = -k_7$ and $t_1 = 0$. Then, for each k_7, we have $t_2 = t_0 = -t_2$. This is sufficient for $k_7 + t_2 = k_8 + t_0$ to be constant (actually 0), which now ensures the C^8 property for sub-key k_2.

With exploiting the above property, we can construct 12-round tuple property in the inverse computation. The construction of the property is described in Fig. 4:

1. In the inverse direction, we set the subkey and the state value between rounds 11 and 12 to tuples $\mathsf{C}^4\mathsf{P}\mathsf{C}^3$, such that the P tuples of the state and the key are the additive inverses of each other. This makes the constant round up to round 4. Finally, after the inverse computation for round 4, the subkey tuple is $\mathsf{C}^7\mathsf{P}$, which leaves a balanced word in the plaintext.

2. In the forward direction, we simply search for the tuple in the state value between rounds 11 and 12. Our code detected that after 5 rounds, some word in the internal state still keeps a detectable property. Finally, by combining the tuple properties in both directions, we can detect a non-ideal property for 17 rounds of Threefish-512.

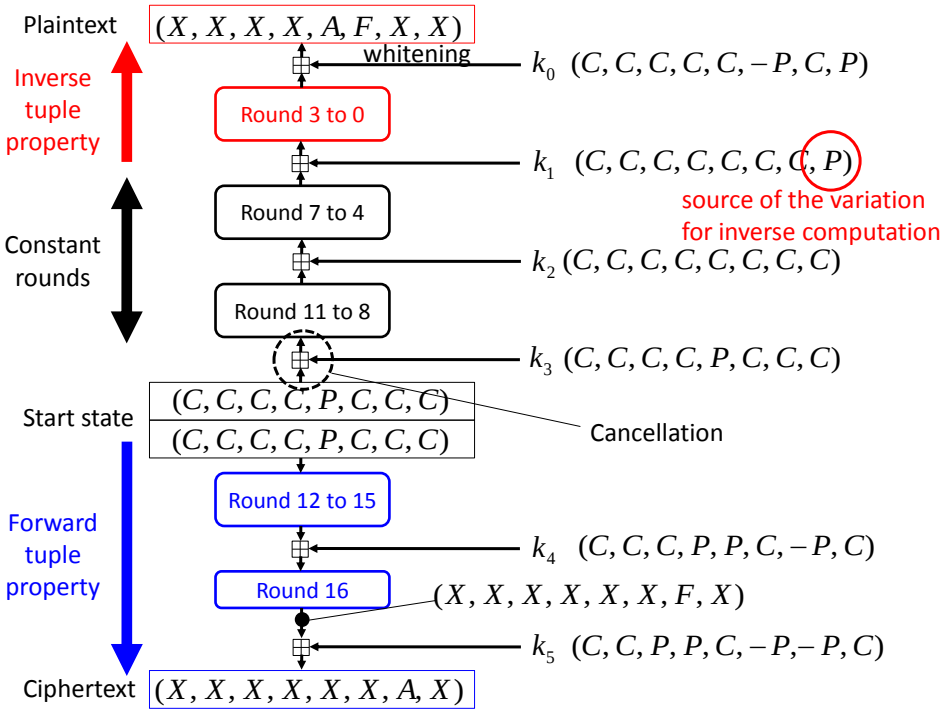

Fig. 4. Sketch of a chosen-keys attack for 17 rounds of Threefish-512. Notation $-\mathsf{P}$ represents that elements in this tuple are ordered so that addition of P and $-\mathsf{P}$ tuples are 0.

This attack, with a complexity of 2^{64}, finds a 2^{64} plaintexts whose additive-sum is zero and 2^w for two words in the specified positions, and the additive-sum of 2^{64} corresponding cihpertexts also has zero for a word in the specified position. On the other hand, for a random permutation, the probability for achieving the same property with just 2^{64} queries is 2^{-64}, and thus our distinguisher has a significant advantage.

4 Application to BLAKE

4.1 Transformations through G

BLAKE's ARX core is the G function, which computes:

$$\begin{aligned}
a &:= a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\
d &:= (d \oplus a) \ggg 16 \\
c &:= c + d \\
b &:= (b \oplus c) \ggg 12 \\
a &:= a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\
d &:= (d \oplus a) \ggg 8 \\
c &:= c + d \\
b &:= (b \oplus c) \ggg 7
\end{aligned}$$

where c_i's are constants and m_i's are message words. In BLAKE, this 4-word keyed permutation is successively applied to the four columns and to the four diagonals of a 4×4 array of words (32-bit in BLAKE-256, 64-bit in BLAKE-512).

Unlike with **MIX**, we won't enumerate all the (1296) entries of the truth table of G. We shall instead give the most interesting and useful transformations. We focus on the half-round function (i.e. the first four lines of G), as it is sufficient to determine tuple-transformation properties of the full G and of a round of BLAKE.

We first examine the one-P transformations by each half G, as they will appear in our distinguishers, though not necessarily as starting points. Our notations follow the G algorithm and are self-explanatory:

- PCCC ↦ PPPP ↦ AXXX:

$$\begin{aligned}
a &:= \mathsf{P} + \mathsf{C} + \mathsf{C} = \mathsf{P} & a &:= \mathsf{P} + \mathsf{P} + \mathsf{C} = \mathsf{A} \\
d &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 16 = \mathsf{P} & d &:= (\mathsf{P} \oplus \mathsf{A}) \ggg 16 = \mathsf{X} \\
c &:= \mathsf{C} + \mathsf{P} = \mathsf{P} & c &:= \mathsf{P} + \mathsf{X} = \mathsf{X} \\
b &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 12 = \mathsf{P} & b &:= (\mathsf{P} \oplus \mathsf{X}) \ggg 12 = \mathsf{X}
\end{aligned}$$

  When the C are all zero and P is some tuple T, then the output tuple after the first half is $(T, T \ggg 28, T \ggg 16, T \ggg 16)$. After the second half of G, one obtains a tuple AXXX.
- CPCC ↦ PPPP ↦ AXXX:

$$\begin{aligned}
a &:= \mathsf{C} + \mathsf{P} + \mathsf{C} = \mathsf{P} & a &:= \mathsf{P} + \mathsf{P} + \mathsf{C} = \mathsf{A} \\
d &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 16 = \mathsf{P} & d &:= (\mathsf{P} \oplus \mathsf{A}) \ggg 16 = \mathsf{X} \\
c &:= \mathsf{C} + \mathsf{P} = \mathsf{P} & c &:= \mathsf{P} + \mathsf{X} = \mathsf{X} \\
b &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 12 = \mathsf{P} & b &:= (\mathsf{P} \oplus \mathsf{X}) \ggg 12 = \mathsf{X}
\end{aligned}$$

- CCPC ↦ CPPC ↦ PXAP:

$$\begin{aligned}
a &:= \mathsf{C} + \mathsf{C} + \mathsf{C} = \mathsf{C} & a &:= \mathsf{C} + \mathsf{P} + \mathsf{C} = \mathsf{P} \\
d &:= (\mathsf{C} \oplus \mathsf{C}) \ggg 16 = \mathsf{C} & d &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 16 = \mathsf{P} \\
c &:= \mathsf{P} + \mathsf{C} = \mathsf{P} & c &:= \mathsf{P} + \mathsf{P} = \mathsf{A} \\
b &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 12 = \mathsf{P} & b &:= (\mathsf{P} \oplus \mathsf{A}) \ggg 12 = \mathsf{X}
\end{aligned}$$

- CCCP ↦ CPPP ↦ PXXB:

$$\begin{aligned}
a &:= \mathsf{C} + \mathsf{C} + \mathsf{C} = \mathsf{C} & a &:= \mathsf{C} + \mathsf{P} + \mathsf{C} = \mathsf{P} \\
d &:= (\mathsf{P} \oplus \mathsf{C}) \ggg 16 = \mathsf{P} & d &:= (\mathsf{P} \oplus \mathsf{P}) \ggg 16 = \mathsf{B} \\
c &:= \mathsf{C} + \mathsf{P} = \mathsf{P} & c &:= \mathsf{P} + \mathsf{B} = \mathsf{X} \\
b &:= (\mathsf{C} \oplus \mathsf{P}) \ggg 12 = \mathsf{P} & b &:= (\mathsf{P} \oplus \mathsf{X}) \ggg 12 = \mathsf{X}
\end{aligned}$$

Among the four choices above, the best option seems to start with a P tuple in the c position, for it leads to the most structured output. Note that the first choice (P in a) leads through the inverse half G to PCCP, and to PCPB after a full G. As with Threefish, one can start with P tuples T and $-T$ in a and b to reach CPCC after one half G.

4.2 Transformations through BLAKE rounds

We consider the BLAKE permutation independently of the redundancy introduced by the compression function. BLAKE's round function processes the four rows of a 4×4 word array in parallel, thus three columns with only C tuples remain only-C after the column step. For example:

$$\begin{pmatrix} \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix} \mapsto \begin{pmatrix} \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{X} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{A} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix}$$

At the next step the four diagonals are transformed in parallel, hence each of the G functions will process at most one non-C tuple. Following our previous example, the diagonal step transforms the tuples as follows:

$$\begin{pmatrix} \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{X} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{A} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix} \mapsto \begin{pmatrix} \mathsf{A} & \mathsf{P} & \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} & \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} & \mathsf{X} & \mathsf{X} \\ \mathsf{B} & \mathsf{X} & \mathsf{X} & \mathsf{X} \end{pmatrix}$$

At the subsequent (column) step, the state becomes all-X.

4.3 Distinguishing attacks

Known-key attack. Like for Skein, one can construct an inside-out distinguisher for some random key (i.e. message), as depicted on Fig. 5: 2.5 rounds of the core permutation of BLAKE can be attacked with this simple strategy, which has complexity 2^{32} for BLAKE-256 and 2^{64} for BLAKE-512.

$$\begin{pmatrix} \mathsf{X} & \mathsf{E} & \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} & \mathsf{P} & \mathsf{X} \\ \mathsf{A} & \mathsf{X} & \mathsf{X} & \mathsf{A} \\ \mathsf{B} & \mathsf{X} & \mathsf{X} & \mathsf{X} \end{pmatrix} \leftarrow \begin{pmatrix} \mathsf{A} & \mathsf{P} & \mathsf{A} & \mathsf{A} \\ \mathsf{B} & \mathsf{P} & \mathsf{C} & \mathsf{B} \\ \mathsf{P} & \mathsf{B} & \mathsf{F} & \mathsf{C} \\ \mathsf{P} & \mathsf{P} & \mathsf{X} & \mathsf{F} \end{pmatrix} \leftarrow \begin{pmatrix} \mathsf{C} & \mathsf{C} & \mathsf{A} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{B} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{P} & \mathsf{C} & \mathsf{C} \end{pmatrix} \leftarrow \begin{pmatrix} \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix} \rightarrow \begin{pmatrix} \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{X} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{A} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix} \rightarrow \begin{pmatrix} \mathsf{A} & \mathsf{P} & \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} & \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} & \mathsf{X} & \mathsf{X} \\ \mathsf{B} & \mathsf{X} & \mathsf{X} & \mathsf{X} \end{pmatrix}$$

Fig. 5. Tuple properties of BLAKE's ARX chain, for random message and constants. The first three arrows denote the tuple property for the backward computation and the last two arrows denote the tuple property for the forward computation.

Chosen-keys attack (sketch). In the same spirit as the chosen-keys attack on Threefish, we sketch a chosen-keys attack on BLAKE's core permutation, as depicted on Fig. 6. The proposed attack targets four rounds of BLAKE's permutation, using the word permutations corresponding to round 3.5 to 7.5.

The attack collects 2^{64} values of the following form for the state between round 5 and 6 and the message, which is an XOR of the start state for the backward and forward directions.

$$\begin{pmatrix} \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{P} & \mathsf{C} & \mathsf{C} & \mathsf{C} \\ \mathsf{C} & \mathsf{C} & \mathsf{C} & \mathsf{C} \end{pmatrix}, (m_0, m_1, m_2, m_3, \ldots, m_{15}) = (\mathsf{C}, \mathsf{C}, -\mathsf{P}, \mathsf{C}, \ldots, \mathsf{C}).$$

Note that the property in the backward direction can be held for any constant value for a pair of upper-left word denoted by P and m_2 denoted by $-\mathsf{P}$. Therefore, for all 2^{32} values of a pair of upper-left word and m_2, the property in the backward direction is iterated, which preserves the property of B and P for the XOR-sum of 2^{32} iterated results. Similarly, the same thing will occur for the forward direction. In the end, this attack, with a complexity of 2^{64}, finds a 2^{64} inputs whose XOR-sum is 0 for two words in the specified positions, and the XOR-sum of 2^{64} corresponding outputs also has 0 for four words in the specified positions, A detailed analysis is necessary to verify or improve this attack strategy.

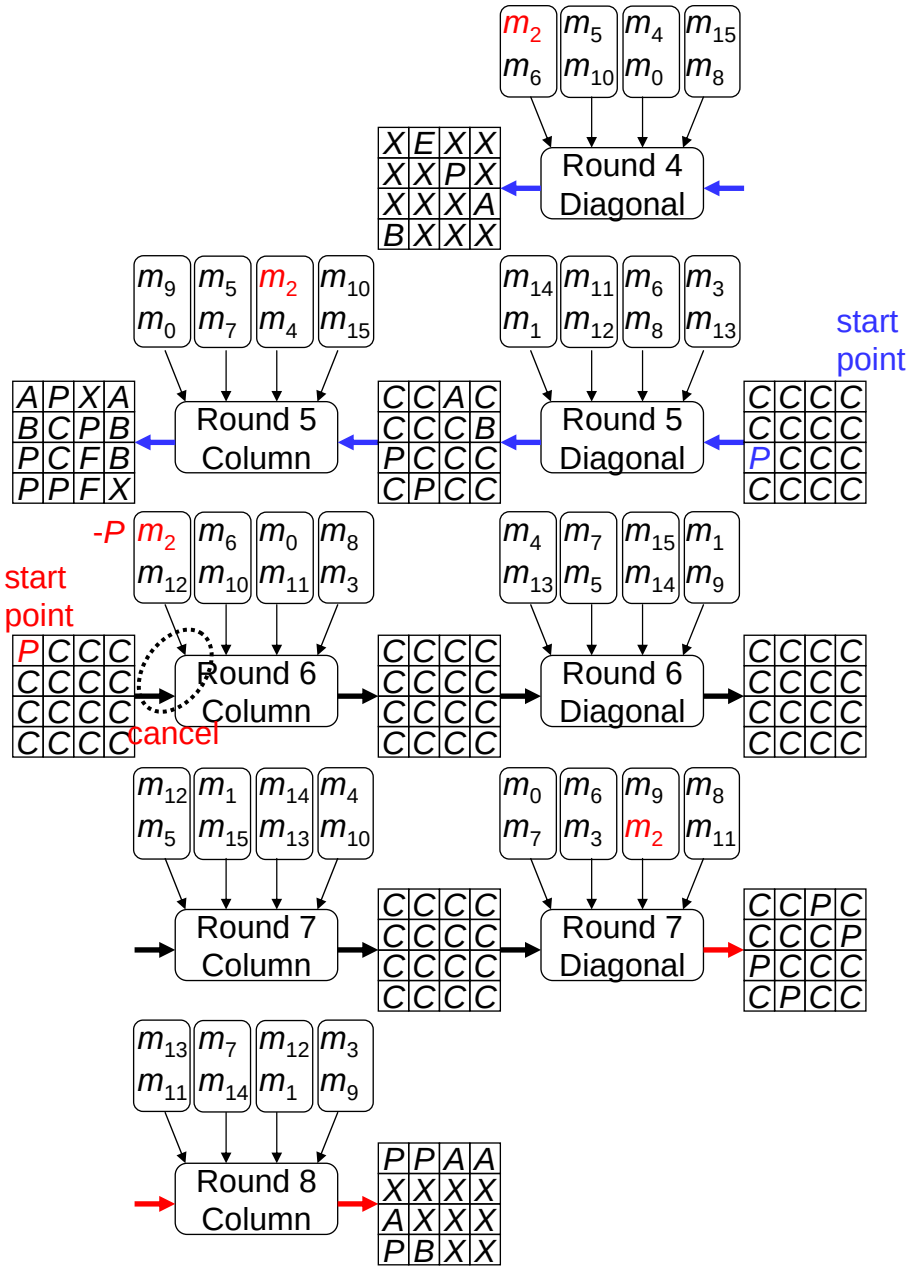

Fig. 6. Sketch of a chosen-keys attack for four rounds of BLAKE's core permutation.

5 Conclusion

We introduced tuple attacks, a refinement of multiset attacks, and showed how to apply it to ARX algorithms. Although our preliminary application to Skein and BLAKE does not improve on previous attacks, tuple attacks have the advantage of being efficiently verifiable, because their power—in terms of rounds attacked—is mostly independent of the word size. We thus propose tuple attacks as a new benchmark tool to evaluate the security level of ARX chains.

Future work is expected to improve our attacks on Skein and BLAKE's ARX cores, and to better understand how the bit-level structure of tuples affects the evolution of tuples' properties. For example, a question of great interest is the existence of fixed-points of the round transformation of tuple properties.

References

1. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: FSE. (1997)
2. Lucks, S.: The saturation attack - a bait for Twofish. In: FSE. (2001)
3. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: FSE. (2002)
4. Gilbert, H., Minier, M.: A collision attack on seven rounds of Rijndael. In: Third AES Conference. (2000)
5. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. J. Cryptology **23**(4) (2010)
6. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-pattern based integral attack. In: FSE. (2008)
7. Aumasson, J.P.: Zero-sum distinguishers. Rump session of CHES 2009 http://131002.net/data/talks/zerosum_rump.pdf.
8. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. Public comment on the NIST Hash Competition (2009) http://131002.net/data/papers/AM09.pdf.
9. Boura, C., Canteaut, A.: Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256. In: SAC. (2010)
10. Boura, C., Canteaut, A.: A zero-sum property for the Keccak-f permutation with 18 rounds. In: ISIT. (2010)
11. Aumasson, J.P., Çalık, C., Meier, W., Özen, O., Phan, R.C.W., Varıcı, K.: Improved cryptanalysis of Skein. In: ASIACRYPT. (2009)