An Improved Affine Equivalence Algorithm for Random Permutations

Itai Dinur*

Department of Computer Science, Ben-Gurion University, Israel

Abstract. In this paper we study the affine equivalence problem, where given two functions $\boldsymbol{F}, \boldsymbol{G} : \{0,1\}^n \rightarrow \{0,1\}^n$, the goal is to determine whether there exist invertible affine transformations A_1, A_2 over $GF(2)^n$ such that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Algorithms for this problem have several well-known applications in the design and analysis of Sboxes, cryptanalysis of white-box ciphers and breaking a generalized Even-Mansour scheme.
We describe a new algorithm for the affine equivalence problem and focus on the variant where $\boldsymbol{F}, \boldsymbol{G}$ are permutations over n-bit words, as it has the widest applicability. The complexity of our algorithm is about $n^3 2^n$ bit operations with very high probability whenever $\boldsymbol{F}$ (or $\boldsymbol{G}$) is a random permutation. This improves upon the best known algorithms for this problem (published by Biryukov et al. at EUROCRYPT 2003), where the first algorithm has time complexity of $n^3 2^{2n}$ and the second has time complexity of about $n^3 2^{3n/2}$ and roughly the same memory complexity.
Our algorithm is based on a new structure (called a *rank table*) which is used to analyze particular algebraic properties of a function that remain invariant under invertible affine transformations. Besides its standard application in our new algorithm, the rank table is of independent interest and we discuss several of its additional potential applications.
Keywords: Affine equivalence problem, block cipher, Even-Mansour cipher, cryptanalysis, rank table.

1 Introduction

In the affine equivalence problem, the input consists of two functions $\boldsymbol{F}, \boldsymbol{G}$ and the goal is to determine whether they are affine equivalent, and if so, output the equivalence relations. More precisely, if there exist invertible affine transformations (over some field) A_1, A_2 such that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$, output A_1, A_2. Otherwise, assert that $\boldsymbol{F}, \boldsymbol{G}$ are not affine equivalent.

Variants of the affine equivalence problem have been studied in several branches of mathematics and are relevant to both asymmetric and symmetric cryptography. In the context of asymmetric cryptography, the problem was first formalized by Patarin [17] and referred to as isomorphism of polynomials. In this setting $\boldsymbol{F}, \boldsymbol{G}$ are typically of low algebraic degree (mainly quadratic) over some field.

* The author was supported in part by the Israeli Science Foundation through grant No. 573/16.

The focus of this work is on the affine equivalence variant in which $\boldsymbol{F}, \boldsymbol{G}$ map between n-bit words and the affine transformations A_1, A_2 are over $GF(2)^n$. This variant is mostly relevant in several contexts of symmetric-key cryptography. In particular, it is relevant to the classification and analysis of Sboxes (see [6, 14]) as affine equivalent Sboxes share differential, linear and several algebraic properties (refer to [7] for recent results on this subject). Moreover, algorithms for the affine equivalence problem were applied in [3] to generate equivalent representations of AES and other block ciphers. These algorithms also have cryptanalytic applications and were used to break white-box ciphers (e.g., in [15]). Additionally, solving the affine equivalence problem can be viewed as breaking a generalization of the Even-Mansour scheme [11], which has received substantial attention from the cryptographic community in recent years. The original scheme builds a block cipher from a public permutation $\boldsymbol{F}$ using two n-bit keys k_1, k_2 and its encryption function is defined as $\boldsymbol{E}(p) = \boldsymbol{F}(p + k_1) + k_2$ (where addition is over $GF(2)^n$). The generalized Even-Mansour scheme replaces the key additions with secret affine mappings and breaking it reduces to solving the affine equivalence problem, as originally described in [3].

The best known algorithms for the affine equivalence problem were presented by Biryukov et al. at EUROCRYPT 2003 [3]. The main algorithm described in [3] has complexity of about $n^3 2^{2n}$ bit operations, while a secondary algorithm has time complexity of about $n^3 2^{3n/2}$, but also uses about the same amount of memory.[1] Besides its high memory consumption, another disadvantage of the secondary algorithm of [3] is that it cannot be used to prove that $\boldsymbol{F}$ and $\boldsymbol{G}$ are not affine equivalent.

In this paper we devise a new algorithm for the affine equivalence problem whose complexity is about $n^3 2^n$ bit operations with very high probability whenever $\boldsymbol{F}$ (or $\boldsymbol{G}$) is chosen uniformly at random from the set of all permutations on n-bit words. Our algorithm is also applicable without any modification to arbitrary functions (rather than permutations) and seems to perform similarly on random functions. However we focus on permutations as almost all applications actually require solving the affine equivalence problem for permutations. Since our algorithm can be used to prove that $\boldsymbol{F}$ and $\boldsymbol{G}$ are not affine equivalent, it does not share the disadvantage of the secondary algorithm of [3].

As a consequence of our improved algorithm, we solve within several minutes affine equivalence problem instances of size up to $n = 28$ on a single core. Optimizing our implementation and exploiting parallelism would most likely allow solving instances of size at least $n = 40$ using an academic budget. Such instances are out of reach of all previous algorithms for the problem.

Technically, the main algorithm devised in [3] for the affine equivalence problem is a guess-and-determine algorithm (which is related to the "to and fro" algorithm of [18], devised to solve the problem of isomorphism of polynomials) whereas the secondary algorithm is based on collision search (it generalizes

[1] Biryukov et al. also described a more efficient algorithm of complexity $n^3 2^n$ for the linear equivalence problem, which is a restricted variant of the affine equivalence problem.

Daemen's attack on the original Even-Mansour cipher [8]). On the other hand, algorithms that use algebraic techniques (such as [5]) are mainly known for the asymmetric variant, in which $\boldsymbol{F}, \boldsymbol{G}$ are functions of low degree, and it is not clear how to adapt them to arbitrary functions.

In contrast to previous algorithms, our approach involves analyzing algebraic properties of $\boldsymbol{F}, \boldsymbol{G}$ which are of high algebraic degree. More specifically, we are interested in the polynomial representation (algebraic normal form or ANF) of each of the n output bits of $\boldsymbol{F}$ (and $\boldsymbol{G}$) as a Boolean function in the n input bits. In fact, we are mainly interested in "truncated" polynomials that include only monomials of degree at least d (in particular, we choose $d = n - 2$). Each such polynomial can be viewed a vector in a vector space (with the standard basis of all monomials of degree at least $d = n - 2$). Therefore we can define the rank of the set of n truncated polynomials for each $\boldsymbol{F}, \boldsymbol{G}$ as the rank of the matrix formed by arranging these polynomials as row vectors. In other words, we associate a rank value (which is an integer between 0 and n) to $\boldsymbol{F}$ (and to $\boldsymbol{G}$) by computing the rank of its n truncated polynomials (derived from its n output bits) as vectors. We first show that if $\boldsymbol{F}, \boldsymbol{G}$ are affine equivalent, their associated ranks are equal.[2]

To proceed, we analyze $\boldsymbol{F}, \boldsymbol{G}$ independently. We derive from $\boldsymbol{F}$ several functions, each one defined by restricting its 2^n inputs to an affine subspace of dimension $n-1$. Since each such derived function (restricted to an affine subspace) has an associated rank, we assign to each possible $(n-1)$-dimensional subspace a corresponding rank. As there are 2^{n+1} possible affine subspaces (such a subspace can be characterized using its orthogonal subspace by a single linear expression over n variables and a free coefficient), we obtain 2^{n+1} rank values for $\boldsymbol{F}$. These values are collected in the *rank table* of $\boldsymbol{F}$, where a rank table entry r stores the set of all affine subspaces (more precisely, their compact representations as linear expressions) assigned to rank r.[3]

The main idea of the algorithm is to compute the rank tables of both $\boldsymbol{F}$ and $\boldsymbol{G}$ and then use these tables (and additional more complex structures derived from them) to recover the (unknown) affine transformation A_1, assuming that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. In essence, the rank tables allow us to recover *matchings* between $(n-1)$-dimensional affine subspaces that are defined by A_1: an affine subspace S in matched with S' if A_1 transforms S to the affine subspace S'. Each such matching between S and S' reveals information about A_1 in the form of linear equations. Hence we aim to use the rank tables to recover sufficiently many such matchings and compute A_1 using linear algebra. Once A_1 is derived, computing A_2 is trivial. The main property of the rank table that we prove and exploit to recover the matchings is that if S in matched with S', then S appears in the rank table of $\boldsymbol{G}$ in the same entry r as S' appears in the rank table of $\boldsymbol{F}$.

[2] The choice of $d = n - 2$ seems arbitrary at this stage. We only note that choosing a larger or smaller value for d typically results in a rank value which is constant for almost all functions, providing no information about them.

[3] The formal definition of a rank table is slightly different.

Since the number of $(n-1)$-dimensional affine subspaces is 2^{n+1}, each containing 2^{n-1} elements, a naive approach to computing the rank table (which works independently on each subspace) has complexity of at least $2^{n+1} \cdot 2^{n-1} = 2^{2n}$. However, using symbolic computation of polynomials, we show how to reduce this complexity to about $n^3 2^n$ bit operations. While this computational step is easy to analyze, this is not the case for the overall algorithm's performance. Indeed, its success probability and complexity depend on the monomials of degree at least $n-2$ of $\boldsymbol{F}$ and $\boldsymbol{G}$. In particular, if all n output bits of $\boldsymbol{F}$ and $\boldsymbol{G}$ are functions of degree $n-3$ or lower, they do not contain any such monomials. As a result, all affine subspaces for $\boldsymbol{F}$ and $\boldsymbol{G}$ are assigned rank zero and the rank tables of these functions contain no useful information, leading to failure of the algorithm.[4]

When $\boldsymbol{F}$ (or $\boldsymbol{G}$) is chosen uniformly at random from the set of all possible n-bit permutations (or n-bit functions in general), the case that its algebraic degree is less than $n-2$ is extremely unlikely for $n \geq 8$. Nevertheless, rigorous analysis of the algorithm seems challenging as its performance depends on subtle algebraic properties of random permutations. To deal with this situation, we make a heuristic assumption about the distribution of high degree monomials in random permutations which enables us to use well-known results regarding the rank distribution of random Boolean matrices. Consequently, we derive the distribution of the sizes of the rank table entries for a random permutation. This distribution and additional properties enable us to show that asymptotically the algorithm succeeds with probability close to 1 in complexity of about $n^3 2^n$ bit operations. This heuristic analysis is backed up by thousands of experiments on various problem instances of different sizes. Rigorously analyzing the algorithm and extending it to succeed on all functions (or permutations) with probability 1 in the same complexity remain open problems.

The properties of the rank table and the algorithm for computing it are of independent interest. In particular, we propose methods to build experimental distinguishers for block ciphers based on the rank table and a method to efficiently detect high-order differential distinguishers based on the algorithm for its computation. Furthermore, our techniques are relevant to decomposition attacks on the white-box ASASA block cipher instances proposed by Biryukov et al. [2]. In this application, we adapt the algorithm for computing the rank table in order to improve the complexity of the integral attack on ASASA published in [10] from $2^{3n/2}$ to about 2^n (where n is the block size of the instance).

The rest of the paper is organized as follows. In Section 2 we describe some preliminaries and give an overview of the new affine equivalence algorithm in Section 3. In Section 4 we prove the basic property of rank equality for affine equivalent functions, while in Section 5 we define and analyze the matching between $(n-1)$-dimensional affine subspaces that we use to recover A_1. In Section 6 we define the rank table and additional objects used in our algorithm, and describe the relation between these objects for affine equivalent functions.

[4] In case the algorithm fails, one can try to apply it to $\boldsymbol{F^{-1}}$ and $\boldsymbol{G^{-1}}$ which may be of higher algebraic degree.

In Section 7 we analyze properties of rank tables for random permutations under our heuristic assumption. Then, we describe and analyze the new affine equivalence algorithm in Section 8. Next, in Section 9, we describe applications of the new algorithm and the rank table structure. Finally, we conclude the paper in Section 10.

2 Preliminaries

For a finite set R, denote by $|R|$ its size. Given a vector $u = (u[1], \ldots, u[n]) \in GF(2)^n$, let $wt(u)$ denote its Hamming weight. Throughout this paper, addition between vectors $u_1, u_2 \in GF(2)^n$ is performed bit-wise over $GF(2)^n$.

Multivariate Polynomials Any Boolean function $F : \{0,1\}^n \to \{0,1\}$ can be represented as a multivariate polynomial whose algebraic normal form (ANF) is unique and given as $F(x[1], \ldots, x[n]) = \sum_{u=(u[1],\ldots,u[n])\in\{0,1\}^n} \alpha_u M_u$, where $\alpha_u \in \{0,1\}$ is the coefficient of the monomial $M_u = \prod_{i=1}^{n} x[i]^{u[i]}$, and the sum is over $GF(2)$. The algebraic degree of the function F is defined as $deg(F) = max\{wt(u) \mid \alpha_u \neq 0\}$.

In several cases it will be more convenient to directly manipulate the representation of F as a multivariate polynomial $P(x[1], \ldots, x[n]) = \sum_{u\in\{0,1\}^n} \alpha_u M_u$. Note that unlike F, the polynomial P is not treated as a function but rather as a symbolic object. $P(x[1], \ldots, x[n])$ can be viewed as a vector in the vector space spanned by the set of all monomials $\{M_u \mid u \in \{0,1\}^n\}$.

Given a multivariate polynomial $P(x[1], \ldots, x[n]) = \sum_{u\in\{0,1\}^n} \alpha_u M_u$ and a positive integer d, define $P_{(\geq d)}$ by taking all the monomials of P of degree at least d, namely, $P_{(\geq d)}(x[1], \ldots, x[n]) = \sum_{u\in\{0,1\}^n \wedge wt(u)\geq d} \alpha_u M_u$. Note that $P_{(\geq d)}$ can be represented using at most $\sum_{i=d}^{n} \binom{n}{i}$ non-zero coefficients.

Given a function $F : \{0,1\}^n \to \{0,1\}$ represented by a polynomial $P(x[1], \ldots, x[n])$, define $F_{(\geq d)} : \{0,1\}^n \to \{0,1\}$ as the function represented by $P_{(\geq d)}$.

Vectorial Functions and Polynomials Given a vectorial Boolean function $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, let $F^{(i)} : \{0,1\}^n \to \{0,1\}$ denote the Boolean function of its i'th output bit.

We say that a sequence of m polynomials $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^{m}$ represents $\boldsymbol{F}$ if for each $i \in \{1, 2, \ldots, m\}$, the i'th polynomial $P^{(i)}$ represents $F^{(i)}$.

Given a positive integer d, denote $\boldsymbol{P}_{(\geq d)} = \{P^{(i)}_{(\geq d)}(x[1], \ldots, x[n])\}_{i=1}^{m}$. The vectorial function $\boldsymbol{F}_{(\geq d)} : \{0,1\}^n \to \{0,1\}^m$ is defined analogously.

The algebraic degree $deg(\boldsymbol{P})$ of $\boldsymbol{P}$ is defined as the maximal degree of its m polynomials. The algebraic degree $deg(\boldsymbol{F})$ is defined analogously.

As each $P^{(i)}$ can be viewed as a vector in a vector space, we define the *symbolic rank* of $\boldsymbol{P}$ as the rank of the m vectors $\{P^{(i)}\}_{i=1}^{m}$. We denote the symbolic rank of $\boldsymbol{P}$ as $SR(\boldsymbol{P})$. Note that $SR(\boldsymbol{P}) \in \mathbb{Z}_{m+1}$.

Affine Transformations and Affine Equivalence An affine transformation $A : \{0,1\}^m \to \{0,1\}^n$ over $GF(2)^m$ is defined using a Boolean matrix $L_{n\times m}$ and a word $a \in \{0,1\}^n$ as $A(x) = L(x) + a$ ($L(x)$ is simply matrix multiplication). The transformation is invertible if $m = n$ and L is an invertible matrix. If $a = 0$, then the A is called a linear transformation (such functions are a subclass of affine functions).

Two functions $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ are *affine equivalent* is there exist two invertible affine transformations $A_1 : \{0,1\}^n \to \{0,1\}^n$ and $A_2 : \{0,1\}^m \to \{0,1\}^m$ such that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. It is easy to show that the affine equivalence relation partitions the set of all functions into (affine) equivalence classes. We denote $\boldsymbol{F} \equiv \boldsymbol{G}$ if $\boldsymbol{F}$ is affine equivalent to $\boldsymbol{G}$.

Symbolic Composition Given $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^m$, and an affine function $A_1 : \{0,1\}^{n'} \to \{0,1\}^n$, the composition $\boldsymbol{P} \circ A_1$ is a sequence of m polynomials in n' variables. For $i \in \{1, 2, \ldots, m\}$, the i'th polynomial in this composition is $P^{(i)} \circ A_1$. It can be computed by substituting each variable $x[j]$ in $P^{(i)}$ with the (affine) symbolic representation of the j'th output bit of A_1 (and simplifying the outcome to obtain the ANF).

For example, given $P(x[1], x[2], x[3]) = x[1]x[2] + x[1]x[3] + x[2] + 1$ and $A_1 : \{0,1\}^2 \to \{0,1\}^3$ defined by the relations $x[1] = y[1] + y[2] + 1, x[2] = y[2], x[3] = y[1] + y[2]$, then

$$P \circ A_1 = (y[1] + y[2] + 1)(y[2]) + (y[1] + y[2] + 1)(y[1] + y[2]) + y[2] + 1 =$$
$$(y[1]y[2] + y[2] + y[2]) + (y[1] + y[1]y[2] + y[1]y[2] + y[2] + y[1] + y[2]) + y[2] + 1 =$$
$$y[1]y[2] + y[2] + 1.$$

Thus, we compose each monomial M_u with coefficient 1 in P with A_1 to obtain a polynomial expression, add all the expressions and simplify the result. Formally, if we denote P's M_u coefficient by α_u, then

$$P \circ A_1 = \sum_{u \in \{0,1\}^n} \alpha_u \cdot (M_u \circ A_1).$$

Note that composition with an affine function does not increase the algebraic degree of the composed polynomial, namely $deg(\boldsymbol{P} \circ A_1) \leq deg(\boldsymbol{P})$.

Analogously, given an affine function $A_2 : \{0,1\}^m \to \{0,1\}^{m'}$, the composition $A_2 \circ \boldsymbol{P}$ is a sequence of m' polynomials in n variables. It can be computed by substituting each variable $x[j]$ in the (affine) symbolic representation of A_2 with $\boldsymbol{P}^{(j)}$. Equivalently, if $A_2(x) = L(x) + a$, then $A_2 \circ \boldsymbol{P}$ can be computed by symbolic matrix multiplication (and addition of a) as $L(\boldsymbol{P}) + a$. In particular, if $m' = 1$ and $a = 0$, then A_2 reduces to a vector $v = (v[1], v[2], \ldots, v[m]) \in \{0,1\}^m$ and $v(\boldsymbol{P}) = \sum_{i=1}^m v[i]\boldsymbol{P}^{(i)}$ is a symbolic inner product.

By rules of composition, if $\boldsymbol{F}$ is represented by $\boldsymbol{P}$, then $\boldsymbol{P} \circ A_1$ represents $\boldsymbol{F} \circ A_1$ (which is a standard composition of functions) and $A_2 \circ \boldsymbol{P}$ represents $A_2 \circ \boldsymbol{F}$.

Half-Space Masks and Coefficients Let $A : \{0,1\}^{n-1} \to \{0,1\}^n$ be an affine transformation such that $A(x) = L(x) + a$ for a matrix $L_{n \times n-1}$ with linearly independent columns. Then the (affine) range of A is an $(n-1)$-dimensional affine subspace spanned by the columns of L with the addition of a. The subspace orthogonal to the range of A is of dimension 1 and hence spanned by a single non-zero vector $h \in \{0,1\}^n$. Namely, a vector $v \in \{0,1\}^n$ is in the range of A if and only if $h(v+a) = 0$, i.e., v satisfies the linear equation $h(v) + h(a) = 0$.

Since h partitions the space of $\{0,1\}^n$ into two halves, we call h the *half-space mask* (HSM) of A and call the bit $h(a)$ the *half-space free coefficient* (HSC) of A.

We call the linear subspace spanned by the columns of L the *linear range* of A. A vector $v \in \{0,1\}^n$ is in the linear range of A if and only if $h(v) = 0$.

Canonical Affine Transformations Given non-zero $h \in \{0,1\}^n$ and $c \in \{0,1\}$, there exist many affine transformations whose HSM and HSC are equal to h, c, respectively. We will use the fact (stated formally below) that all affine transformations with an identical affine range are related by composition on the right with an invertible affine transformation.

Fact 1 *The affine transformations $A_1 : \{0,1\}^{n-1} \to \{0,1\}^n$ and $A_2 : \{0,1\}^{n-1} \to \{0,1\}^n$ have the same affine range if and only if there exists an invertible affine transformation $A' : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$ such that $A_1 = A_2 \circ A'$.*

Given A_1, A_2 the matrix A' above can be computed using basic linear algebra.

We now define the *canonical affine transformation* $C_{|h,c} : \{0,1\}^{n-1} \to \{0,1\}^n$ with respect to h, c. Let ℓ denote the index of the first non-zero bit of $h = (h[1], \ldots, h[n])$. Write $C_{|h,c}(x) = L(x) + a$. We define $a = c \cdot e_\ell$ (where e_ℓ is the ℓ'th unit vector) and define $L[i]$ (the i'th column of L) using h and the unit vectors as follows:

$$L[i] = \begin{cases} e_i & \text{if } i < \ell \\ e_{i+1} + h[i+1]e_\ell & \text{otherwise } (\ell \leq i \leq n-1) \end{cases}$$

Thus, on input $(y[1], \ldots, y[n-1])$, the transformation $C_{|h,c}$ is defined by the symbolic form

$$(x[1], x[2], \ldots, x[n]) = (y[1], \ldots, y[\ell-1], \sum_{i=\ell}^{n-1} h[i+1]y[i] + c, y[\ell], \ldots, y[n-1]).$$

The motivation behind the definition of $C_{|h,c}$ is that it allows very simple symbolic composition when applied on the right: its main action is to replace the variable $x[\ell]$ with the affine combination that is specified by the coefficients of h and by c. Other variables are just renamed: variables with index $i < \ell$ remain with the same index, while for each variable with index $i > \ell$, its index it reduced by 1.

Remark 1. Note that we have to show that the definition of $C_{|h,c}$ is valid. First, the $n-1$ columns of L are clearly linearly independent. It remains to prove

that h, c are indeed the HSM and HSC of $C_{|h,c}$. For this purpose, it suffices to show that for each column $L[i]$, the vector $L[i] + a$ satisfies the equation $h(L[i] + a) + c = 0$. Since $h(a) = h(c \cdot e_\ell) = c \cdot h(e_\ell) = c$ (as $h_\ell = 1$), it remains to show that $h(L[i]) = 0$. Indeed, if $0 \leq i < \ell$, then $h(L[i]) = h_i = 0$ (as ℓ is the index of the first non-zero bit of h). Otherwise, $\ell \leq i \leq n - 1$, then $h(L[i]) = h[i + 1] + h[i + 1]h[\ell] = 0$ (as $h[\ell] = 1$).

3 Overview of the New Affine Equivalence Algorithm

We demonstrate the new algorithm using an example. Although it is oversimplified, this example is sufficient to convey the main ideas of our algorithm.

Definition of Functions We define the function $\boldsymbol{F} : \{0,1\}^3 \to \{0,1\}^3$ using its symbolic representation $\boldsymbol{P} = \{P^{(i)}(x[1], x[2], x[3])\}_{i=1}^3$,

$$\begin{aligned} P^{(1)}(x[1], x[2], x[3]) =& x[1]x[2] + x[1]x[3] + x[2] + 1 \\ P^{(2)}(x[1], x[2], x[3]) =& x[1]x[2] + x[1] + x[2] \\ P^{(3)}(x[1], x[2], x[3]) =& x[1]x[3] + x[3]. \end{aligned}$$

We define $\boldsymbol{G} : \{0,1\}^3 \to \{0,1\}^3$ using 2 affine transformations as $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$, where A_2 is simply the identity and A_1 is defined using the relations

$$x[1] = y[1] + y[3] + 1,\ x[2] = y[1] + y[2],\ x[3] = y[2].$$

Composing $A_2 \circ \boldsymbol{P} \circ A_1$ and simplifying the resultant ANFs, gives the symbolic representation of $\boldsymbol{G}$ as $\boldsymbol{Q} = \{Q^{(i)}(y[1], y[2], y[3])\}_{i=1}^3$, where

$$\begin{aligned} Q^{(1)}(y[1], y[2], y[3]) =& y[1]y[3] + y[1] + y[2] + 1 \\ Q^{(2)}(y[1], y[2], y[3]) =& y[1]y[2] + y[1]y[3] + y[2]y[3] + y[3] + 1 \\ Q^{(3)}(y[1], y[2], y[3]) =& y[1]y[2] + y[2]y[3]. \end{aligned}$$

The input to our affine equivalence algorithm is $\boldsymbol{F}, \boldsymbol{G}$ defined above and its goal is to recover the (presumably) unknown affine transformation A_1. The first step of the algorithm is to interpolate $\boldsymbol{F}, \boldsymbol{G}$ and obtain $\boldsymbol{P}, \boldsymbol{Q}$, respectively.

Rank Tables and Histograms The most basic property that we prove in Theorem 1 is that since $\boldsymbol{F}$ and $\boldsymbol{G}$ are affine equivalent, the symbolic ranks of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (as vectors) are equal. Indeed, it is easy to verify that both $\boldsymbol{P}$ and $\boldsymbol{Q}$ have symbolic rank of 3. More significantly, Theorem 1 is stronger and asserts that $SR(\boldsymbol{P}_{(\geq d)}) = SR(\boldsymbol{Q}_{(\geq d)})$ for every $d \geq 1$. Indeed, if we take $d = 2$, we get $\boldsymbol{P}_{(\geq 2)} = \{x[1]x[2]+x[1]x[3], x[1]x[2], x[1]x[3]\}$, which has symbolic rank 2. This is also the symbolic rank of $\boldsymbol{Q}_{(\geq 2)} = \{y[1]y[3], y[1]y[2]+y[1]y[3]+y[2]y[3], y[1]y[2]+y[2]y[3]\}$.

We would like to use this property to recover A_1. Let us examine the 2-dimensional affine subspace defined by the 3-bit HSM $h' = 100$ (whose bits are $h[1] = 1, h[2] = 0, h[3] = 0$) and the single bit HSC $c = 0$. We calculate the symbolic form of $(\boldsymbol{F} \circ C_{|h',0})_{(\geq 1)}$ by evaluating $(\boldsymbol{P} \circ C_{|h',0})_{(\geq 1)}$ (i.e., plugging $x[1] = 0$ into $\boldsymbol{P}_{(\geq 1)}$) and obtain $\{x[2], x[2], x[3]\}$ which has symbolic rank 2. Similarly, for $c = 1$ we calculate $(\boldsymbol{P} \circ C_{|h',1})_{(\geq 1)}$ (i.e., plug $x[1] = 1$ into $\boldsymbol{P}_{(\geq 1)}$) and obtain $\{x[3], 0, 0\}$ which has symbolic rank 1. Hence, we attach the symbolic rank pair $(2, 1)$ to $h' = 100$. We do the same for all 7 non-zero $h' \in \{0, 1\}^3$. The result is a table whose entries are pairs of ranks of the form $(maxR, minR) \in \mathbb{Z}_4 \cdot \mathbb{Z}_4$ (where $maxR \geq minR$), such that entry $(maxR, minR)$ stores the set of HSMs that are associated with this pair of ranks.

$$(3, 2) : \{010, 011, 111, 110\}$$
$$(2, 2) : \{001\}$$
$$(2, 1) : \{100, 101\}$$

This table is called the *rank table* of $\boldsymbol{F}$ (with respect to the degree $d = 1$ as we only considered monomials of degree at least 1). The set of HSMs in an entry $(maxR, minR)$ of the rank table is called a *rank group* (e.g., the rank group with index $(2, 1)$ is $\{100, 101\}$). Similarly, we compute the rank table of $\boldsymbol{G}$ with respect to $d = 1$.

$$(3, 2) : \{100, 001, 110, 011\}$$
$$(2, 2) : \{010\}$$
$$(2, 1) : \{101, 111\}$$

Although the rank tables are different, the size of each rank group $(maxR, minR)$ of $\boldsymbol{F}, \boldsymbol{G}$ is identical. We define the *rank histogram* of $\boldsymbol{F}$ (with respect to d) as a mapping from each $(maxR, minR)$ value to the corresponding rank group size (e.g., the histogram entry for $\boldsymbol{F}$ with index $(2, 1)$ has value $|\{100, 101\}| = 2$). As we show in Lemma 9, that rank histograms of affine equivalent functions (such as $\boldsymbol{F}, \boldsymbol{G}$) are identical.

To explain this, we look at the HSM $h = 101$ in the rank group of $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$ with index $(2, 1)$ and note that it partitions the space into halves $\{000, 101, 010, 111\}$ and $\{001, 011, 100, 110\}$ (according to whether $x[1] + x[3] = 0$ or $x[1] + x[3] = 1$). After applying A_1, these half-spaces are mapped into $\{100, 101, 110, 111\}$ and $\{000, 001, 010, 011\}$. This is exactly the partition defined by $h' = 100$, which is in the rank group of $\boldsymbol{F}$ with the same index $(2, 1)$. In terms of canonical affine transformations, $C_{|h',c}$ and $A_1 \circ C_{|h,0}$ have the same half-space range of $\{100, 101, 110, 111\}$ (for $c = 1$ in this case) and we define a mapping $h \mapsto_{A_1} h'$ to capture this. In Lemma 3 we show that this mapping is a bijection as A_1 is invertible.

Exploiting Matchings The central property of the mapping $\mapsto_{A_1}$ is proved in Lemma 6 which asserts that it preserves affine equivalence. Namely, if $\boldsymbol{F} \equiv \boldsymbol{G}$ and $h \mapsto_{A_1} h'$, then $\boldsymbol{F} \circ C_{|h',c} \equiv \boldsymbol{G} \circ C_{|h,0}$ (for some $c \in \{0,1\}$). By flipping the half-space ranges and applying the same argument, we also obtain $\boldsymbol{F} \circ C_{|h',c+1} \equiv \boldsymbol{G} \circ C_{|h,1}$. Combined with Theorem 1 (which states that symbolic rank is an invariant of affine equivalent functions) we obtain that for any $d \geq 1$, $r_0 \triangleq SR((\boldsymbol{F} \circ C_{|h',c})_{(\geq d)}) = SR((\boldsymbol{G} \circ C_{|h,0})_{(\geq d)})$ and $r_1 \triangleq SR((\boldsymbol{F} \circ C_{|h',c+1})_{(\geq d)}) = SR((\boldsymbol{G} \circ C_{|h,1})_{(\geq d)})$. Since the ordered rank pairs for h', h in $\boldsymbol{F}, \boldsymbol{G}$ are equal to $(maxR, minR)$ (for $maxR = max\{r_0, r_1\}, minR = min\{r_0, r_1\}$), they belong to rank groups with the same index $(maxR, minR)$ in the rank tables of $\boldsymbol{F}, \boldsymbol{G}$, respectively. The fact that $\mapsto_{A_1}$ is a bijection leads to Lemma 9 (which assets that the rank histograms of $\boldsymbol{F}, \boldsymbol{G}$ are identical).

The main goal of our affine equivalence algorithm is to recover *matchings* $h \mapsto_{A_1} h'$ for several pairs h, h'. This is useful, as in Lemma 5 we show that each such matching gives n linear equations on the unknown matrix L of $A_1(x) = L(x) + a$. Furthermore, the constant c associated with $h \mapsto_{A_1} h'$ (which determines whether $\boldsymbol{F} \circ C_{|h',0} \equiv \boldsymbol{G} \circ C_{|h,0}$ or $\boldsymbol{F} \circ C_{|h',1} \equiv \boldsymbol{G} \circ C_{|h,0}$) gives a linear equation on a (once again, by Lemma 5). In total, we need to find about n matchings $h \mapsto_{A_1} h'$ along with their associated constants to completely recover A_1.

Going back to the example, the rank group with index $(2,2)$ for $\boldsymbol{G}$ is $\{010\}$, while the rank group with the same index for $\boldsymbol{F}$ is $\{001\}$. Therefore, after computing the rank tables we know that

$$010 \mapsto_{A_1} 001. \tag{1}$$

Remark 2. We matched $010 \mapsto_{A_1} 001$ in the rank group $(maxR, minR) = (2,2)$ and since $maxR = minR$ we cannot derive the constant c associated with this matching (hence we cannot derive a linear equation on a). Such constants can only be derived for matchings $h \mapsto_{(A_1)} h'$ in rank groups where $maxR > minR$, as in such cases we know whether $\boldsymbol{F} \circ C_{|h',0} \equiv \boldsymbol{G} \circ C_{|h,0}$ or $\boldsymbol{F} \circ C_{|h',1} \equiv \boldsymbol{G} \circ C_{|h,0}$ according to the equality $SR((\boldsymbol{F} \circ C_{|h',c})_{(\geq d)}) = SR((\boldsymbol{G} \circ C_{|h,0})_{(\geq d)})$. More precisely, if $maxR > minR$, then either $SR((\boldsymbol{F} \circ C_{|h',0})_{(\geq d)}) = SR((\boldsymbol{G} \circ C_{|h,0})_{(\geq d)})$ or $SR((\boldsymbol{F} \circ C_{|h',1})_{(\geq d)}) = SR((\boldsymbol{G} \circ C_{|h,0})_{(\geq d)})$, but not both. In this sense, it is more useful to recover matchings for HSMs in rank groups $(maxR, minR)$ such that $maxR > minR$.

By applying similar arguments to the rank group $(2,1)$, we know that either $101 \mapsto_{A_1} 100$ or $101 \mapsto_{A_1} 101$ (and similarly $111 \mapsto_{A_1} 100$ or $111 \mapsto_{A_1} 101$). Since we have very few possibilities, we can guess which matchings hold, derive A_1 and test our guess. Unfortunately, for larger n we expect the rank groups to be much bigger and it would be inefficient to exhaustively match HSMs for $\boldsymbol{F}, \boldsymbol{G}$ only based on their ranks. Thus, to narrow down the number of possibilities and eventually uniquely match sufficiently many pairs h, h' such that $h \mapsto_{A_1} h'$, we need to attach more data to each HSM for $\boldsymbol{F}, \boldsymbol{G}$.

HSM Rank Histograms The main observation that allows attaching more data to each HSM is given in Lemma 4 which shows that the mapping $\mapsto_{A_1}$ is additive.

Consider the two rank groups with index $(3,2)$ for $\boldsymbol{F}, \boldsymbol{G}$. They are of size 4 and their HSMs cannot be uniquely matched. We first focus on $\boldsymbol{G}$ and examine the rank group $(3,2)$ which is $\{100, 001, 110, 011\}$. We take $h_1 = 011$ and compute its *HSM rank histogram* with respect to the rank group $(2,1)$ (which is $\{101, 111\}$). This is done by computing the $(maxR, minR)$ rank pairs for the set defined by adding all elements of rank group $(2,1)$ to 011, namely $\{011 + 101, 011 + 111\} = \{110, 100\}$. Looking for 110 and 100 in the rank table of $\boldsymbol{G}$, both HSMs have ranks $(3,2)$. Thus, the HSM rank histogram of $h_1 = 011$ with respect to rank group $(2,1)$ has a single non-zero entry $(3,2)$ with the value of 2. We write this HSM rank histogram in short as $[(3,2) : 2]$.

We now consider the match of $h_1 = 011$ under A_1 which is $h'_1 = 110$ (namely, $h_1 \mapsto_{(A_1)} h'_1$). Similarly to h_1, we compute the HSM rank histogram of h'_1 with respect to the rank group $(2,1)$ for $\boldsymbol{F}$ (which is $\{100, 101\}$) and obtain the same HSM rank histogram $[(3,2) : 2]$. This is a particular case of Lemma 10, which shows that matching HSMs for $\boldsymbol{F}, \boldsymbol{G}$ have identical HSM rank histograms (with respect to a fixed rank group). Lemma 10 is derived using Lemma 4 which asserts that the mapping $\mapsto_{A_1}$ is additive: if $h_1 \mapsto_{(A_1)} h'_1$ and $h_2 \mapsto_{(A_1)} h'_2$, then $(h_1 + h_2) \mapsto_{(A_1)} (h'_1 + h'_2)$.

Fixing $h_1 = 011$ for $\boldsymbol{G}$ and its match $h'_1 = 110$ under A_1, let $h_2^i, h_2'^i$ for $i \in \{1, 2\}$ vary over the 2 elements of the rank groups with index $(2,1)$ in $\boldsymbol{G}, \boldsymbol{F}$, respectively. Then, as $h_1 \mapsto_{(A_1)} h'_1$ and $h_2^i \mapsto_{(A_1)} h_2'^i$ for $i \in \{1, 2\}$, we get $(h_1 + h_2^i) \mapsto_{(A_1)} (h'_1 + h_2'^i)$. By the aforementioned Theorem 1 and Lemma 6 (equating ranks for matching HSMs) we conclude that indeed the HSM rank histograms of 011 and 110 with respect to rank group $(2,1)$ are identical (which is a special case of Lemma 10).

HSM Rank Histogram Multi-Sets Since we do not know in advance that $h_1 \mapsto_{(A_1)} h'_1$, we have to compute the HSM rank histograms (with respect to rank group $(2,1)$) for all HSMs in rank group $(3,2)$. The outcome is the *HSM rank histogram multi-set* of rank group $(3,2)$ with respect to rank group $(2,1)$. It is computed by considering all the HSMs in the rank group $(3,2)$, namely $\{100, 001, 110, 011\}$ for $\boldsymbol{G}$ and $\{010, 011, 111, 110\}$ for $\boldsymbol{F}$. Lemma 11 (whose proof is based on Lemma 10) asserts that these HSM rank histogram multi-sets are identical as $\boldsymbol{F}, \boldsymbol{G}$ are affine equivalent.

We hope that these multi-sets contain unique HSM rank histograms (with multiplicity 1), which would allow us to derive more matching between HSMs. Unfortunately, the resultant multi-set (for both $\boldsymbol{F}$ and $\boldsymbol{G}$) is $\{[(3,2) : 2], [(3,2) : 2], [(3,2) : 2], [(3,2) : 2]\}$. It contains 4 identical elements and does not give us any new information about A_1. If the multiplicity of the element $[(3,2) : 2]$ (calculated above for h_1, h'_1) in this multi-set would have been 1, we could have derived the relation $h_1 \mapsto_{(A_1)} h'_1$.

Remark 3. Generally, when n is very small (as in our case), the direct application of the algorithm is more likely to fail to completely recover A_1. As we show later in this paper, for $n \geq 8$ the fraction of instances for which this occurs is very small (and tends to 0 as n grows). In some cases a failure to retrieve A_1 occurs since the affine mappings A_1, A_2 are not uniquely defined. In particular, if there are several solutions for A_1, then we cannot hope to obtain unique matchings that completely define A_1, but we can recover all possible solutions to the affine equivalence problem by enumerating several possibilities for the matchings.

In conclusion, we attached to each HSM in rank group $(3, 2)$ for $\boldsymbol{F}, \boldsymbol{G}$ its HSM rank histogram with respect to rank group $(2, 1)$ and in general such data may allow us to derive additional matchings $h \mapsto_{(A_1)} h'$. Once we obtain about n matchings, we can recover A_1 by solving a system of linear equations.

4 A Basic Property of Affine Equivalent Functions

Before proving the main result of this section, we state two useful lemmas (the first is proved in the extended version of this paper [9]).

Lemma 1. *Let* $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^{m}$, *let* $A_1 : \{0,1\}^{n'} \to \{0,1\}^n$, $A_2 : \{0,1\}^m \to \{0,1\}^{m'}$ *be affine functions, and* d *be a positive integer. Then,*

1. $(\boldsymbol{P}_{(\geq d)} \circ A_1)_{(\geq d)} = (\boldsymbol{P} \circ A_1)_{(\geq d)}$
2. $(A_2 \circ (\boldsymbol{P}_{(\geq d)}))_{(\geq d)} = (A_2 \circ \boldsymbol{P})_{(\geq d)}$ *and if* A_2 *is a linear function,* $A_2 \circ (\boldsymbol{P}_{(\geq d)}) = (A_2 \circ \boldsymbol{P})_{(\geq d)}$.

Essentially, the lemma states that removing all monomials of degree less than d from $\boldsymbol{P}$ can be done before or after composing it with an affine function and the outcomes are identical.

Note that a potentially simplified first part of the lemma which equates $\boldsymbol{P}_{(\geq d)} \circ A_1$ and $(\boldsymbol{P} \circ A_1)_{(\geq d)}$ is generally incorrect, as the first expression may contain monomials of degree less than d. For example, if $d = 2$ and we compose the affine transformation defined by $x[1] = y[1] + y[2]$ and $x[2] = y[2]$ with the polynomial $x[1]x[2]$, then we get the polynomial $y[1]y[2] + y[2]$ which has a monomial of degree 1.

Lemma 2. *Let* $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^{m}$, *and let* $A_1 : \{0,1\}^n \to \{0,1\}^n$ *be an invertible affine function. Then,* $deg(\boldsymbol{P}) = deg(\boldsymbol{P} \circ A_1)$.

Proof. We show that for $i \in \{1, 2, \ldots, m\}$, $deg(P^{(i)}) = deg(P^{(i)} \circ A_1)$. Observe that $deg(P^{(i)}) \geq deg(P^{(i)} \circ A_1)$ as composition with an affine function cannot increase the algebraic degree of a polynomial. By the same argument and by the invertibility of A_1, we also obtain $deg(P^{(i)} \circ A_1) \geq deg(P^{(i)} \circ A_1 \circ (A_1)^{-1}) = deg(P^{(i)})$.

■

Theorem 1. *Let* $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ *be two affine equivalent functions, represented by* $\boldsymbol{P}, \boldsymbol{Q}$, *respectively. Then, for every positive integer* d, $SR(\boldsymbol{P}_{(\geq d)}) = SR(\boldsymbol{Q}_{(\geq d)})$.

Proof. At a high level, the fact that $\boldsymbol{P}$ and $\boldsymbol{Q}$ have the same symbolic rank follows since rank is preserved by composition with invertible affine transformations. Moreover, this rank equality is preserved after truncating low degree monomials since they cannot affect the high degree monomials when composing with invertible affine transformations. The formal proof is below.

Write $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$, implying that $\boldsymbol{Q} = A_2 \circ \boldsymbol{P} \circ A_1$. Denote $\boldsymbol{P'} = \boldsymbol{P} \circ A_1$ and observe that

$$SR(\boldsymbol{P'}_{(\geq d)}) = SR((A_2 \circ (\boldsymbol{P'}_{(\geq d)}))_{(\geq d)}) = SR((A_2 \circ \boldsymbol{P'})_{(\geq d)}) = SR(\boldsymbol{Q}_{(\geq d)}),$$

where the first equality holds since rank is preserved by invertible linear transformations[5] and the second equality is due to the second part of Lemma 1.

It remains to show that $SR(\boldsymbol{P'}_{(\geq d)}) = SR(\boldsymbol{P}_{(\geq d)})$, or $SR((\boldsymbol{P} \circ A_1)_{(\geq d)}) = SR(\boldsymbol{P}_{(\geq d)})$. We first show that $SR(\boldsymbol{P}_{(\geq d)}) \geq SR((\boldsymbol{P} \circ A_1)_{(\geq d)})$.

If $\boldsymbol{P}_{(\geq d)}$ has full rank of m then the claim is trivial. Otherwise, let $v \in \{0,1\}^m$ be a non-zero vector in the kernel of $\boldsymbol{P}_{(\geq d)}$, namely $v(\boldsymbol{P}_{(\geq d)}) = 0$. Then,

$$v((\boldsymbol{P} \circ A_1)_{(\geq d)}) = (v((\boldsymbol{P} \circ A_1)_{(\geq d)}))_{(\geq d)} = (v(\boldsymbol{P}_{(\geq d)}) \circ A_1)_{(\geq d)} = 0,$$

where the first equality follows from the second part of Lemma 1 and the second equality follows from the first part of this lemma. This implies that v is also in the kernel of $(\boldsymbol{P} \circ A_1)_{(\geq d)}$, as required.

To prove that $SR(\boldsymbol{P}_{(\geq d)}) \leq SR((\boldsymbol{P} \circ A_1)_{(\geq d)})$, observe that if v is a non-zero vector in the kernel of $(\boldsymbol{P} \circ A_1)_{(\geq d)}$, then by the equality above we have $0 = v((\boldsymbol{P} \circ A_1)_{(\geq d)}) = (v(\boldsymbol{P}_{(\geq d)}) \circ A_1)_{(\geq d)}$. This implies that $deg(v(\boldsymbol{P}_{(\geq d)}) \circ A_1) < d$ and since A_1 is invertible, by Lemma 2, $deg(v(\boldsymbol{P}_{(\geq d)})) = deg(v(\boldsymbol{P}_{(\geq d)}) \circ A_1) < d$. This gives $v(\boldsymbol{P}_{(\geq d)}) = 0$, as the polynomial does not contain monomials of degree less than d. Hence, v is in the kernel of $\boldsymbol{P}_{(\geq d)}$ which completes the proof. ■

5 The Half-Space Mask Bijection and its Properties

Definition 1. *Let $A : \{0,1\}^n \to \{0,1\}^n$ be an invertible affine transformation. Define a mapping between HSMs using A as follows: $h \in \{0,1\}^n$ is mapped to h' if there exists $c \in \{0,1\}$ such that the affine ranges of $A \circ C_{|h,0}$ and $C_{|h',c}$ are equal. We write $h \mapsto_{(A)} h'$ and say that h and h' match (under A). The bit c is called the associated constant of $h \mapsto_{(A)} h'$.*

Lemma 3. *Let $A : \{0,1\}^n \to \{0,1\}^n$ be an invertible affine transformation. The mapping $\mapsto_{(A)}$ is a bijection and its inverse is given by $\mapsto_{(A^{-1})}$.*

Proof. The proof follows from the invertibility of A. Given that $h \mapsto_{(A)} h'$, there exists $c \in \{0,1\}$ such that the affine ranges of $A \circ C_{|h,0}$ and $C_{|h',c}$ are equal. According to Fact 1, this implies that there exists an invertible affine

[5] The affine transformation A_2 also adds a constant, but it does contribute to the rank as $d > 0$.

transformation $A' : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$ such that $A \circ C_{|h,0} = C_{|h',c} \circ A'$. Consequently, $C_{|h,0} \circ (A')^{-1} = A^{-1} \circ C_{|h',c}$ and the affine ranges of $C_{|h,0}$ and $A^{-1} \circ C_{|h',c}$ are equal (again, according to Fact 1). This implies that the affine ranges of $A^{-1} \circ C_{|h',0}$ and $C_{|h,c}$ are equal (flipping the HSC of both sides if $c = 1$), namely $h' \mapsto_{(A^{-1})} h$. ■

A property of $\mapsto_{(A)}$ which will be very useful is that it is additive. This is established by the lemma below (proved in the extended version of this paper [9]).

Lemma 4. *Let $A : \{0,1\}^n \to \{0,1\}^n$ be an invertible affine transformation. Let $h_1, h'_1, h_2, h'_2 \in \{0,1\}^n$ be HSMs where $h_1 \neq h_2$ and $h_1 \mapsto_{(A)} h'_1$, $h_2 \mapsto_{(A)} h'_2$ with associated constants c_1, c_2, respectively. Then $(h_1 + h_2) \mapsto_{(A)} (h'_1 + h'_2)$ with the associated constant $c_1 + c_2$.*

The following lemma (proved in the extended version of this paper [9]) shows that the bijection reveals information about the presumably unknown transformation A.

Lemma 5. *Let $A : \{0,1\}^n \to \{0,1\}^n$ be an invertible affine transformation such that $A(x) = L(x) + a$. Let $h, h' \in \{0,1\}^n$ be HSMs such that $h \mapsto_{(A)} h'$ with associated constant c. Then, A satisfies the following constraints.*

1. *For each $i \in \{1, 2, \ldots, n\}$, the i'th column of L, denoted by $L[i]$, satisfies the equation $h'(L[i]) = h[i]$, where $h[i]$ is the i'th bit of h.*
2. *The vector a satisfies the equation $h'(a) = c$.*

The following lemma asserts that affine equivalence is preserved under composition with matching HSMs.

Lemma 6. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions such that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Let $h, h' \in \{0,1\}^n$ be HSMs such that $h \mapsto_{(A_1)} h'$ with associated constant c. Then, $\boldsymbol{F} \circ C_{|h',c} \equiv \boldsymbol{G} \circ C_{|h,0}$ and $\boldsymbol{F} \circ C_{|h',c+1} \equiv \boldsymbol{G} \circ C_{|h,1}$.*

Proof. Since $h \mapsto_{(A_1)} h'$ with associated constant c, the affine ranges of $A_1 \circ C_{|h,0}$ and $C_{|h',c}$ are equal. According to Fact 1, there exists an invertible affine transformation $A'_1 : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$ such that $A_1 \circ C_{|h,0} = C_{|h',c} \circ A'_1$.

We obtain, $A_2 \circ \boldsymbol{F} \circ C_{|h',c} \circ A'_1 = A_2 \circ \boldsymbol{F} \circ A_1 \circ C_{|h,0} = \boldsymbol{G} \circ C_{|h,0}$, implying that $\boldsymbol{F} \circ C_{|h',c}$ and $\boldsymbol{G} \circ C_{|h,0}$ are affine equivalent.

The claim that $\boldsymbol{F} \circ C_{|h',c+1}$ and $\boldsymbol{G} \circ C_{|h,1}$ are affine equivalent follows by considering the complimentary half-space and observing that the affine ranges of $A_1 \circ C_{|h,1}$ and $C_{|h',c+1}$ are equal. The remainder of the proof is similar.

■

Definition 2. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$ be a function represented by $\boldsymbol{P}$, let d be a positive integer and let $h \in \{0,1\}^n$ be a HSM. Let $r_0 = SR((\boldsymbol{P} \circ C_{|h,0})_{(\geq d)})$, $r_1 = SR((\boldsymbol{P} \circ C_{|h,1})_{(\geq d)})$, $maxR = max\{r_0, r_1\}$ and $minR = min\{r_0, r_1\}$.*

1. *The HSM rank of h (with respect $\boldsymbol{F}, d$) is the ordered pair of integers $(maxR, minR)$, denoted as $R_{\boldsymbol{F},d,h}$,*

2. *The attached constant of h is the value $c \in \{0,1\}$ such that $maxR = SR((\boldsymbol{P} \circ C_{|h,c})_{(\geq d)})$ (if $maxR = minR$, the attached constant is undefined).*

The lemma below states that the HSM ranks of matching HSMs are equal for affine equivalent functions.

Lemma 7. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Let $h, h' \in \{0,1\}^n$ be HSMs such that $h \mapsto_{(A_1)} h'$. Then $R_{\boldsymbol{G},d,h} = R_{\boldsymbol{F},d,h'}$.*

Proof. Let $c \in \{0,1\}$ be the associated constant of $h \mapsto_{(A_1)} h'$. According to Lemma 6, $\boldsymbol{F} \circ C_{|h',c} \equiv \boldsymbol{G} \circ C_{|h,0}$ and $\boldsymbol{F} \circ C_{|h',c+1} \equiv \boldsymbol{G} \circ C_{|h,1}$.

Assume that $\boldsymbol{F}, \boldsymbol{G}$ are represented by $\boldsymbol{P}, \boldsymbol{Q}$, respectively. Denote $r'_0 = SR((\boldsymbol{F} \circ C_{|h',c})_{(\geq d)})$, $r'_1 = SR((\boldsymbol{F} \circ C_{|h',c+1})_{(\geq d)})$, $r_0 = SR((\boldsymbol{G} \circ C_{|h,0})_{(\geq d)})$, $r_1 = SR((\boldsymbol{G} \circ C_{|h,1})_{(\geq d)})$.

By the above affine equivalences and Theorem 1, we have $r_0 = r'_0$ and $r_1 = r'_1$. Hence $max(r_0, r_1) = max(r'_0, r'_1)$ and $min(r_0, r_1) = min(r'_0, r'_1)$ and the lemma follows.

■

6 Rank Tables, Rank Histograms and their Properties

Definition 3. *Given a function $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$ and a positive integer d, define the following mappings.*

1. *The rank table of $\boldsymbol{F}$ with respect to d is a mapping $\mathcal{T}_{\boldsymbol{F},d}$, whose keys (indexes) are integer pairs $(maxR, minR) \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$ such that $maxR \geq minR$. It is defined as*

$$\mathcal{T}_{\boldsymbol{F},d}(maxR, minR) = \{h \in \{0,1\}^n \mid R_{\boldsymbol{F},d,h} = (maxR, minR)\}.$$

   Moreover, along with each such HSM h, the table stores its attached constant $c \in \{0,1\}$ (if defined).
   An entry in the rank table $\mathcal{T}_{\boldsymbol{F},d}(maxR, minR)$ (containing all HSMs with this rank) is called a rank group.
2. *The rank histogram of $\boldsymbol{F}$ with respect to d is a mapping $\mathcal{H}_{\boldsymbol{F},d} : \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1} \to \mathbb{Z}$ such that $\mathcal{H}_{\boldsymbol{F},d}(maxR, minR) = |\mathcal{T}_{\boldsymbol{F},d}(maxR, minR)|$.*

To simplify our notation, in the following we refer to a HSM rank $(maxR, minR) \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$ such that $maxR \geq minR$ using a single symbol $\boldsymbol{r}$.

The lemma below states that if $\boldsymbol{F} \equiv \boldsymbol{G}$, then each HSM with rank $\boldsymbol{r}$ for $\boldsymbol{G}$ is matched in the rank group with the same HSM rank $\boldsymbol{r}$ for $\boldsymbol{F}$.

Lemma 8. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$ and let $\boldsymbol{r} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$. Then, for each $h \in \mathcal{T}_{\boldsymbol{G},d}(\boldsymbol{r})$, there exists $h' \in \mathcal{T}_{\boldsymbol{F},d}(\boldsymbol{r})$ such that $h \mapsto_{(A_1)} h'$.*

Proof. By Lemma 7, given $h \in \mathcal{T}_{\boldsymbol{G},d}(\boldsymbol{r})$, its match h' under A_1 satisfies $R_{\boldsymbol{F},d,h'} = R_{\boldsymbol{G},d,h} = \boldsymbol{r}$ hence $h' \in \mathcal{T}_{\boldsymbol{F},d}(\boldsymbol{r})$ as claimed.

■

Lemma 9. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Then the rank histograms of $\boldsymbol{F}$ and $\boldsymbol{G}$ with respect to d are equal, namely $\mathcal{H}_{\boldsymbol{F},d} = \mathcal{H}_{\boldsymbol{G},d}$.*

Proof. Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Given a histogram entry with index $\boldsymbol{r}$, for each $h \in \mathcal{T}_{\boldsymbol{G},d}(\boldsymbol{r})$ let h' be its match $h \mapsto_{(A_1)} h'$. Then, by Lemma 8, $h' \in \mathcal{T}_{\boldsymbol{F},d}(\boldsymbol{r})$. Since $h \mapsto_{(A_1)} h'$ is a bijection, this shows that $\mathcal{H}_{\boldsymbol{G},d}(\boldsymbol{r}) = |\mathcal{T}_{\boldsymbol{G},d}(\boldsymbol{r})| \leq |\mathcal{T}_{\boldsymbol{F},d}(\boldsymbol{r})| = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r})$. On the other hand, as $\mathcal{H}_{\boldsymbol{G},d}(\boldsymbol{r}) \leq \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r})$ holds for all histogram entries $\boldsymbol{r}$ and the sum of entries in both histograms is $2^n - 1$, this implies that $\mathcal{H}_{\boldsymbol{F},d} = \mathcal{H}_{\boldsymbol{G},d}$.

■

Definition 4. *Given a function $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, a positive integer d, a HSM $h_1 \in \{0,1\}^n$ and $\boldsymbol{r} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$, we define the HSM rank histogram of h_1 with respect to (or relative to) the rank group $\boldsymbol{r}$ and denote it by $\mathcal{HG}_{\boldsymbol{F},d,h_1,\boldsymbol{r}}$. As the standard histogram, it is a mapping $\mathcal{HG}_{\boldsymbol{F},d,h_1,\boldsymbol{r}} : \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1} \to \mathbb{Z}$, where*

$$\mathcal{HG}_{\boldsymbol{F},d,h_1,\boldsymbol{r}}(\boldsymbol{r'}) = |\{h_1 + h_2 \mid h_2 \in \{0,1\}^n \wedge h_1 \neq h_2 \wedge R_{\boldsymbol{F},d,h_2} = \boldsymbol{r} \wedge R_{\boldsymbol{F},d,h_1+h_2} = \boldsymbol{r'}\}|.$$

Note that unlike the (standard) rank histogram, the HSM rank histogram is defined for a specific HSM with respect to a rank group. We further remark that the HSM rank histogram of h_1 can also be defined with respect to its own the rank group (this is assured by the condition $h_1 \neq h_2$).

The following lemma equates HSM rank histograms for matching HSMs in affine equivalent functions.

Lemma 10. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Let $h_1, h_1' \in \{0,1\}^n$ be such that $h_1 \mapsto_{(A_1)} h_1'$. Then, for every $\boldsymbol{r} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$, $\mathcal{HG}_{\boldsymbol{G},d,h_1,\boldsymbol{r}} = \mathcal{HG}_{\boldsymbol{F},d,h_1',\boldsymbol{r}}$.*

Proof. The proof follows from the fact that the mapping $\mapsto_{(A_1)}$ preserves HSM ranks for affine equivalent functions (Lemma 7), and by exploiting its additive property (Lemma 4).

Fix a HSM rank histogram entry $\boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$. Define the following two sets:

$$D_1 = \{h_1 + h_2 \mid h_2 \in \{0,1\}^n \wedge h_1 \neq h_2 \wedge R_{\boldsymbol{G},d,h_2} = \boldsymbol{r} \wedge R_{\boldsymbol{G},d,h_1+h_2} = \boldsymbol{r'}\}$$

and

$$D_2 = \{h_1' + h_2' \mid h_2' \in \{0,1\}^n \wedge h_1' \neq h_2' \wedge R_{\boldsymbol{F},d,h_2'} = \boldsymbol{r} \wedge R_{\boldsymbol{F},d,h_1'+h_2'} = \boldsymbol{r'}\}.$$

To prove the lemma, we need to show that $|D_1| = |D_2|$. Let $h_1 + h_2 \in D_1$ and denote by $\hat{h} \in \{0,1\}^n$ the vector such that $h_1 + h_2 \mapsto_{(A_1)} \hat{h}$. We show that $\hat{h} \in D_2$.

Since $h_1 + h_2 \mapsto_{(A_1)} \hat{h}$, by Lemma 7, $R_{\boldsymbol{F},d,\hat{h}} = R_{\boldsymbol{G},d,h_1+h_2} = \boldsymbol{r'}$. Next, write $\hat{h} = h'_1 + (h'_1 + \hat{h})$. Since $h_1 \mapsto_{(A_1)} h'_1$ and $h_1 + h_2 \mapsto_{(A_1)} \hat{h}$, by Lemma 4, $h_2 \mapsto_{(A_1)} h'_1 + \hat{h}$, and by Lemma 7, $R_{\boldsymbol{F},d,h'_1+\hat{h}} = R_{\boldsymbol{G},d,h_2} = \boldsymbol{r}$, giving $\hat{h} \in D_2$. Since $\mapsto_{(A_1)}$ is a bijection this implies that $|D_2| \geq |D_1|$.

As $|D_2| \geq |D_1|$ holds for all HSM histogram entries $\boldsymbol{r'}$ and the sum of HSM histogram entries in both $\mathcal{HG}_{\boldsymbol{G},d,h_1,\boldsymbol{r}}$ and $\mathcal{HG}_{\boldsymbol{F},d,h'_1,\boldsymbol{r}}$ is equal to size of the rank group[6] $\boldsymbol{r}$ (which is equal to $\mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r}) = \mathcal{H}_{\boldsymbol{G},d}(\boldsymbol{r})$), the equality $|D_1| = |D_2|$ holds. ■

Definition 5. *Given a function $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, a positive integer d, HSMs ranks $\boldsymbol{r}, \boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$, we define the HSM rank histogram multi-set of rank group r with respect to rank group $\boldsymbol{r'}$ as*

$$\mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}} = \{\mathcal{HG}_{\boldsymbol{F},d,h,\boldsymbol{r'}} \mid R_{\boldsymbol{F},d,h} = \boldsymbol{r}\}.$$

The HSM rank histogram multi-set collects all the HSM histograms for HSMs in rank group $\boldsymbol{r}$ with respect to the rank group $\boldsymbol{r'}$. Note that it is possible to have $\boldsymbol{r} = \boldsymbol{r'}$.

The following lemma equates HSM rank histogram multi-set in affine equivalent functions.

Lemma 11. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Then, for every $\boldsymbol{r}, \boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$, $\mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}} = \mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}}$.*

Proof. Fix $\boldsymbol{r}, \boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$. We define a mapping between the elements (HSM histograms) of the multi-sets $\mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}}$ and $\mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$. Naturally, the mapping is based on the bijection $\mapsto_{(A_1)}$.

Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Let $h \in \{0,1\}^n$ be such that $R_{\boldsymbol{G},d,h} = \boldsymbol{r}$ which implies that $\mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r'}} \in \mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}}$. Let $h' \in \{0,1\}^n$ be the HSM such that $h \mapsto_{(A_1)} h'$. By Lemma 10, $\mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}}$. Furthermore, by Lemma 7 we have $R_{\boldsymbol{F},d,h'} = R_{\boldsymbol{G},d,h} = \boldsymbol{r}$, hence $\mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}} \in \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$. Since $\mapsto_{(A_1)}$ is a bijection, we obtain $\mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}} \subseteq \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$ as multi-sets.

On the other hand, the number of elements (HSM histograms) in both multi-sets is equal to the size of the rank group $\boldsymbol{r}$ (which is equal to $\mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r}) = \mathcal{H}_{\boldsymbol{G},d}(\boldsymbol{r})$), hence $\mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}} = \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$.

■

Definition 6. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, let d be a positive integer and let $\boldsymbol{r}, \boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$. A HSM $h \in \{0,1\}$ such that $R_{\boldsymbol{F},d,h} = \boldsymbol{r}$ is called unique (with respect to $\boldsymbol{F}, d, \boldsymbol{r'}$) if $\mathcal{HG}_{\boldsymbol{F},d,h,\boldsymbol{r'}} \in \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$ has multiplicity 1 in this multi-set.*

[6] Unless the HSM rank of h_1 is $\boldsymbol{r}$, in which case the sum of HSM histogram entries is $\mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r})$ - 1.

The following theorem establishes the importance of unique HSMs in recovering matchings between HSMs for affine equivalent functions.

Theorem 2. *Let $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^m$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^m$ be two affine equivalent functions and let d be a positive integer. Then for every $\boldsymbol{r}, \boldsymbol{r'} \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$, if $h \in \{0,1\}^n$ (such that $R_{\boldsymbol{G},d,h} = \boldsymbol{r}$) is unique with respect to $\boldsymbol{G}, d, \boldsymbol{r'}$, then the following statements hold:*

1. *There exists $h' \in \{0,1\}^n$ such that $R_{\boldsymbol{F},d,h'} = \boldsymbol{r}$ and h' is unique with respect to $\boldsymbol{F}, d, \boldsymbol{r'}$.*
2. *$\mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}}$.*
3. *Assume that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. Then, $h \mapsto_{(A_1)} h'$. Moreover, if the attached constants of h, h' are defined and equal to c, c', respectively, then the associated constant of $h \mapsto_{(A_1)} h'$ is $c + c'$.*

Proof. By Lemma 11, we have equality of the multi-sets $\mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}} = \mathcal{HM}_{\boldsymbol{G},d,\boldsymbol{r},\boldsymbol{r'}}$ which immediately implies the first two statements. Denote by $h'' \in \{0,1\}^n$ the HSM such that $h \mapsto_{(A_1)} h''$. To complete the proof of the third statement we show that $h'' = h'$.

By Lemma 7, we have $R_{\boldsymbol{F},d,h''} = R_{\boldsymbol{G},d,h} = \boldsymbol{r}$. Hence $\mathcal{HG}_{\boldsymbol{F},d,h'',\boldsymbol{r}} \in \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$ (and also $\mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r}} \in \mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$ from the first statement). Since h' is unique with respect to $\boldsymbol{F}, d, \boldsymbol{r'}$, then $\mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}}$ has multiplicity 1 in $\mathcal{HM}_{\boldsymbol{F},d,\boldsymbol{r},\boldsymbol{r'}}$. Thus if we show that $\mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{F},d,h'',\boldsymbol{r}}$, then $h'' = h'$ must hold.

According to Lemma 10, $\mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r}} = \mathcal{HG}_{\boldsymbol{F},d,h'',\boldsymbol{r}}$ and by the second statement we obtain $\mathcal{HG}_{\boldsymbol{F},d,h',\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{G},d,h,\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{F},d,h'',\boldsymbol{r}}$ as required.

Finally, we examine the attached constants c, c' of h, h', respectively (assuming they are defined). If $c = c'$, then the affine ranges of $A_1 \circ C_{|h,0}$ and $C_{|h',0}$ are equal implying that the associated constant of $h \mapsto_{(A_1)} h'$ is $0 = c + c'$. Otherwise $c = c' + 1$ and the affine ranges of $A_1 \circ C_{|h,0}$ and $C_{|h',1}$ are equal implying that the associated constant of $h \mapsto_{(A_1)} h'$ is $1 = c + c'$.

■

7 Analysis of the Distribution of Rank Histogram Entries for Random Permutations

In this section we analyze the distribution of entries of the rank histogram $\mathcal{H}_{\boldsymbol{F},d}$ for a random permutation $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^n$. The analysis is performed for $d = n - 2$, which is the value that we use in our algorithm as explained in detail next.

Assume that $\boldsymbol{F}$ is represented by $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^n$. For a given $h \in \{0,1\}^n$, we consider $SR((\boldsymbol{P} \circ C_{|h,0})_{(\geq n-2)})$ and $SR((\boldsymbol{P} \circ C_{|h,1})_{(\geq n-2)})$. For $c \in \{0,1\}$, every one of the n polynomials of $(\boldsymbol{P} \circ C_{|h,c})_{(\geq n-2)}$ has $n-1$ variables (the number of variables in $\boldsymbol{P}$ is reduced by 1 after composition with $C_{|h,c}$). Hence, the number of possible non-zero monomial coefficients in each such polynomial is $\binom{n-1}{n-1} + \binom{n-1}{n-2} = 1 + n - 1 = n$. Therefore, $(\boldsymbol{P} \circ C_{|h,c})_{(\geq n-2)}$ can be represented by an $n \times n$ Boolean matrix and we are interested in its rank.

Choosing $d = n-1$ would leave at most one non-zero monomial which almost always would be present in $(\boldsymbol{P} \circ C_{|h,c})_{(\geq n-1)}$. Hence, essentially all HSMs would fall into a single rank group and the affine equivalence algorithm would not be able to distinguish and match them. On the other hand, choosing $d \leq n-3$ would leave $\Omega(n^2)$ non-zero monomials and $(\boldsymbol{P} \circ C_{|h,c})_{(\geq r)}$ would almost always have full rank, leading once again to a single rank group. We conclude that $d = n-1$ is indeed the optimal choice.

Our analysis is based on the following heuristic assumption.

Assumption 1 *For a random permutation $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^n$ represented by $\boldsymbol{P}$, for every $h \in \{0,1\}^n$ and $c \in \{0,1\}$, the entries of the $n \times n$ Boolean matrix $(\boldsymbol{P} \circ C_{|h,c})_{(\geq n-2)}$ are uniform independent random variables.*

The $n \times n$ Boolean matrix $(\boldsymbol{P} \circ C_{|h,c})_{(\geq n-2)}$ is indeed uniform for a random function $\boldsymbol{F}$ (rather than a random permutation), given any $h \in \{0,1\}^n$ and $c \in \{0,1\}$. However, even for a random function the Boolean matrices obtained for different h, c values are correlated. Nevertheless, these correlations (and the fact that $\boldsymbol{F}$ is a permutation) do not seem to have a noticeable influence on our algorithm in practice (as we demonstrate in Section 8.5 and the extended version of this paper [9]).

The rank of random matrices is a well-studied problem. For large n and a non-negative integer $r \leq n$, we denote the probability that a random Boolean $n \times n$ matrix has rank r by β_r. We can lower bound β_r by considering the event where we first select r linearly independent rows to form a subspace of size 2^r (which occurs with constant probability) and then select the remaining $n-r$ rows within this subspace (which occurs with probability $2^{-(r-n)^2}$). This gives a lower bound of $\Omega(2^{-(r-n)^2})$ on β_r. The exact formula is given by the theorem below, taken and adapted from [13].

Theorem 3 ([13], page 126, adapted). *For $n \to \infty$, the probability that a random Boolean $n \times n$ matrix has rank r is*

$$\beta_r = 2^{-(r-n)^2} \cdot \alpha \cdot \prod_{i=1}^{n-r} (1 - 1/2^i)^{-2}, \tag{2}$$

where $\alpha = \prod_{i=1}^{\infty}(1 - 1/2^i) \approx 0.2888$.

Since $\alpha \leq \alpha \cdot \prod_{i=1}^{n-r}(1 - 1/2^i)^{-2} < 1/\alpha$, the initial probability estimation of $\approx 2^{-(r-n)^2}$ is correct up to a small constant. We also note that (2) is a good estimation even for relatively small values of n (e.g., $n \geq 8$).
Let

$$p_{maxR,minR} = \begin{cases} 2\beta_{maxR}\beta_{minR} & \text{if } minR < maxR \\ \beta_{maxR}\beta_{minR} & \text{otherwise } (minR = maxR), \end{cases} \tag{3}$$

where

$$\beta_{maxR}\beta_{minR} = \alpha^2 \cdot 2^{-(maxR-n)^2-(minR-n)^2} \cdot \prod_{i=1}^{n-maxR}(1-1/2^i)^{-2} \cdot \prod_{i=1}^{n-minR}(1-1/2^i)^{-2}.$$

Then, based on Assumption 1 and Theorem 3, for every $(maxR, minR) \in \mathbb{Z}_{m+1} \times \mathbb{Z}_{m+1}$ such that $maxR \geq minR$, given $h \in \{0,1\}^n$, we have $Pr[R_{\boldsymbol{F},n-2,h} = (maxR, minR)] \approx p_{maxR,minR}$. Hence, according to Assumption 1, the entries of $\mathcal{H}_{\boldsymbol{F},n-2}$ are distributed multinomially, with parameter 2^n (the number of HSMs[7]) and probabilities given by $p_{maxR,minR}$. In particular, each individual histogram entry $\mathcal{H}_{\boldsymbol{F},n-2}(maxR, minR)$ is distributed binomially with parameter 2^n and probability $p_{maxR,minR}$.

Experimental results that support this conclusion are given in the extended version of this paper [9].

Asymptotic Analysis of Specific Histogram Entries For large n, the binomial variable $\mathcal{H}_{\boldsymbol{F},n-2}(maxR, minR)$ is with high probability very close to its expectation, which is about

$$2^n \cdot p_{maxR,minR}.$$

If we ignore constant multiplicative factors, we can approximate this expectation by

$$2^n \cdot 2^{-(maxR-n)^2-(minR-n)^2}, \tag{4}$$

as $p_{maxR,minR} \approx 2^{-(maxR-n)^2-(minR-n)^2}$.

We now approximate (up to constant multiplicative factors) the expected values of two specific histogram entries which will be useful for our algorithm. Denote $\gamma_n = \lfloor (n/2)^{1/2} \rfloor$, and let $\boldsymbol{r_1} = (n+1-\gamma_n, n-\gamma_n)$ and $\boldsymbol{r_2} = (n, n-\gamma_n)$. Define the random variables $S_1 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_1})$ and $S_2 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_2})$. Below, we estimate their expected values according to (4).

Write $\gamma_n = \lfloor (n/2)^{1/2} \rfloor = (n/2)^{1/2} - k$, where $0 \leq k < 1$. Hence, with very high probability we have $S_2 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_2}) = \mathcal{H}_{\boldsymbol{F},d}(n, n-\gamma_n) \approx 2^n \cdot p_{n,n-\gamma_n} \approx 2^n \cdot 2^{-(\gamma_n)^2} = 2^n \cdot 2^{-((n/2)^{1/2}-k)^2} = 2^n \cdot 2^{-n/2+2k(n/2)^{1/2}-k^2} = 2^{n/2+O(n^{1/2})}$. Therefore, S_2 is close to $2^{n/2}$.

Similarly $S_1 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_1}) = \mathcal{H}_{\boldsymbol{F},d}(n+1-\gamma_n, n-\gamma_n) \approx 2^n \cdot p_{n+1-\gamma_n,n-\gamma_n} \approx 2^n \cdot 2^{-(\gamma_n-1)^2-(\gamma_n)^2} = 2^n \cdot 2^{-2(\gamma_n)^2+2\gamma_n-1} = 2^n \cdot 2^{-n+(4k+2)(n/2)^{1/2}-2k^2-2k-1} = 2^{\Theta(n^{1/2})}$. Hence S_1 is sub-exponential in n.

8 Details of the New Affine Equivalence algorithm

In this section we describe and analyze our new affine equivalence algorithm. We start with a description of the auxiliary algorithms it uses.

[7] More accurately, the parameter is $2^n - 1$ as HSMs are non-zero.

8.1 The Rank Table and Histogram Algorithm

For $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^n$ represented by $\boldsymbol{P} = \{P^{(i)}(x[1], \ldots, x[n])\}_{i=1}^n$, the following algorithm computes the rank histogram $\mathcal{H}_{\boldsymbol{F},d}$ and rank table $\mathcal{T}_{\boldsymbol{F},d}$ for $d = n-2$. The algorithm is given as input $\boldsymbol{P}_{\geq(n-2)}$.

1. For each non-zero HSM $h \in \{0,1\}^n$:
   (a) Compute $R_{\boldsymbol{F},n-2,h} = (maxR, minR)$ as follows. Compute $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,0})_{(\geq n-2)} = (\boldsymbol{P} \circ C_{|h,0})_{(\geq n-2)}$ and calculate its symbolic rank r_0 using Gaussian elimination. Similarly, compute $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,1})_{(\geq n-2)} = (\boldsymbol{P} \circ C_{|h,1})_{(\geq n-2)}$ and its symbolic rank r_1. Let $maxR = max\{r_0, r_1\}$ and $minR = min\{r_0, r_1\}$.
   (b) Insert h into $\mathcal{T}_{\boldsymbol{F},n-2}(maxR, minR)$, along with the value of the attached constant $c \in \{0,1\}$ such that $maxR = SR((\boldsymbol{F} \circ C_{|h,c})_{(\geq n-2)})$ (if $maxR > minR$). In addition, increment entry $\mathcal{H}_{\boldsymbol{F},n-2}(maxR, minR)$.

Note that $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,0})_{(\geq n-2)} = (\boldsymbol{P} \circ C_{|h,0})_{(\geq n-2)}$ holds according to the first part of Lemma 1.

The time complexity of the algorithm depends on how a polynomial is represented. Here, we represent it using a bit array that specifies the values of its monomial coefficients.

We first analyze the complexity of computing the composition $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,c})_{(\geq n-2)}$ in Step 1.(a), which is performed for each non-zero $h \in \{0,1\}^n$ and $c \in \{0,1\}$. Each of the n polynomials of $\boldsymbol{P}_{(\geq n-2)}$ contains at most $\binom{n}{n} + \binom{n}{n-1} + \binom{n}{n-2} < n^2$ non-zero monomials. As described in Section 2, computing the composition $\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,c}$ requires substituting one of the n variables with a linear combination of the remaining $n-1$ variables (while renaming the variables of the monomials).

In total, for each polynomial of $\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,c}$, we compose its n^2 monomials with a linear combination of size n, which requires $n^2 \cdot n = n^3$ bit operations. However, as we are only interested in monomials of degree at least $n-2$, the outcome $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,0})_{(\geq n-2)}$ is a polynomial of at most $\binom{n-1}{n-1} + \binom{n-1}{n-2} = 1 + n - 1 = n$ monomials, and the average complexity can be easily reduced to n^2 using low-level optimization techniques.[8]

In conclusion, the average complexity of computing the n polynomials of $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,0})_{(\geq n-2)}$ is $n \cdot n^2 = n^3$ and the total time spent on composition is $n^3 \cdot 2^n$ bit operations (up to multiplicative constant factors). Similarly, Gaussian elimination requires n^3 bit operations, hence the total time complexity of the algorithm is $n^3 \cdot 2^n$ bit operations.

[8] For example, we can exploit the fact that the composition $(\boldsymbol{P}_{(\geq n-2)} \circ C_{|h,c})_{(\geq n-2)}$ is computed for each $h \in \{0,1\}^n$ and $c \in \{0,1\}$, and the effect of flipping a bit in h on the outcome can be precomputed. Consequently, we iterate over $h \in \{0,1\}^n$ using a Gray code.

8.2 The Unique HSM Algorithm

The following algorithm computes the HSM rank histogram multi-set $\mathcal{HM}_{\boldsymbol{F},n-2,\boldsymbol{r},\boldsymbol{r'}}$ and uses it to compute a set of unique HSMs, denoted by $U_{\boldsymbol{F}}$. This set contains triplets of the form $(h, c, \mathcal{HG}_{\boldsymbol{F},n-2,h,\boldsymbol{r'}})$, where $h \in \mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r})$ is unique with respect to $\boldsymbol{F}, n-2, \boldsymbol{r'}$ and $c \in \{0,1\}$ is its attached constant. Note that for the attached constant to be defined, we must have $maxR > minR$, where $(maxR, minR) = \boldsymbol{r}$.

The algorithm is given as input the rank table $\mathcal{T}_{\boldsymbol{F},n-2}$ and rank group indexes $\boldsymbol{r}, \boldsymbol{r'}$.

1. For each $h \in \mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r})$, compute $\mathcal{HG}_{\boldsymbol{F},n-2,h,\boldsymbol{r'}}$ as follows:
   (a) for each $h' \in \mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r'})$:
      i. Compute $h + h'$, find its rank $\boldsymbol{r''} = R_{\boldsymbol{F},n-2,h+h'}$ in $\mathcal{T}_{\boldsymbol{F},n-2}$ and increment $\mathcal{HG}_{\boldsymbol{F},n-2,h,\boldsymbol{r'}}(\boldsymbol{r''})$.
   (b) Insert $\mathcal{HG}_{\boldsymbol{F},n-2,h,\boldsymbol{r'}}$ along with h and its attached constant c into the multi-set $\mathcal{HM}_{\boldsymbol{F},n-2,\boldsymbol{r},\boldsymbol{r'}}$.
2. For each unique HSM h in $\mathcal{HM}_{\boldsymbol{F},n-2,\boldsymbol{r},\boldsymbol{r'}}$, add the triplet $(h, c, \mathcal{HG}_{\boldsymbol{F},n-2,h,\boldsymbol{r'}})$ to $U_{\boldsymbol{F}}$.

The time complexity of the algorithm is the product of sizes of the rank groups $|\mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r})| \cdot |\mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r'})| = \mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r}) \cdot \mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r'})$.

Since the goal of the affine equivalence algorithm will be to find n linearly independent unique HSMs, it is useful to estimate their number. In the extended version of this paper [9] we lower bound the expected number of unique HSMs in $\mathcal{HM}_{\boldsymbol{F},n-2,\boldsymbol{r},\boldsymbol{r'}}$ asymptotically (ignoring constant factors) based on Assumption 1, given that $\boldsymbol{F}$ is a random permutation. More specifically, we obtain the lower bound of $S - S^2/\sqrt{S'}$, where $S = \mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r})$, and $S' = \mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r'})$.

8.3 The Affine Transformation A_1 Recovery Algorithm

Assume that we have affine equivalent functions $\boldsymbol{F}$ and $\boldsymbol{G}$ such that $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$ and $A_1(x) = L(x) + a$.

The following algorithm recovers A_1 using sets of unique HSMs $U_{\boldsymbol{F}}$ and $U_{\boldsymbol{G}}$, computed with the previous algorithm of Section 8.2 (where its invocations for $\boldsymbol{F}$ and $\boldsymbol{G}$ use the same parameters values of $\boldsymbol{r}, \boldsymbol{r'}$). Since $\boldsymbol{F}$ and $\boldsymbol{G}$ are affine equivalent, the HSM rank histograms of the HSMs in these sets have to match according to Theorem 2. Each equal HSM histogram pair reveals the matching $h \mapsto_{A_1} h'$ and its associated constant is revealed by adding the attached constants of h, h' (which are defined in case $maxR > minR$, where $(maxR, minR) = \boldsymbol{r}$), again by Theorem 2.

Each matching $h \mapsto_{A_1} h'$ and its associated constant give linear equations on the columns of L and on a (respectively) according to Lemma 5. Assuming that $U_{\boldsymbol{F}}$ and $U_{\boldsymbol{G}}$ contain n linearly independent unique HSMs, A_1 is recovered by linear algebra.

1. Allocate $n+1$ linear equation systems $\{E_i\}_{i=1}^{n+1}$, each of dimension $n \times n$: the first n equation systems are on the columns $L[i]$ of L and the final equation system E_{n+1} is on a.
2. Locate n linearly independent HSMs in $U_{\boldsymbol{G}}$. For each such HSM h:
   (a) Recover the triplet $(h, c, \mathcal{HG}_{\boldsymbol{G},n-2,h,\boldsymbol{r'}})$ from $U_{\boldsymbol{G}}$.
   (b) Search $U_{\boldsymbol{F}}$ for a triplet $(h', c', \mathcal{HG}_{\boldsymbol{F},n-2,h',\boldsymbol{r'}})$ such that $\mathcal{HG}_{\boldsymbol{F},n-2,h',\boldsymbol{r'}} = \mathcal{HG}_{\boldsymbol{G},n-2,h,\boldsymbol{r'}}$. If no match exists, return "Not Equivalent".
   (c) Based on Lemma 5, for $i = 1, 2, \ldots, n$ add equation $h'(L[i]) = h[i]$ to E_i.
   (d) Based on Lemma 5, add equation $h'(a) = c + c'$ to E_{n+1}.
3. Solve each one of $\{E_i\}_{i=1}^{n+1}$, recover A_1 and return its matrix L and vector a.

The complexity of the algorithm is about $n \cdot n^3 = n^4$ bit operations, which is polynomial in n. Since we solve the same linear equation (with coefficients given by the h' vectors) $n + 1$ times with different constants, the complexity can be reduced to n^3 by inverting the matrix which defines the linear equations.

8.4 The New Affine Equivalence Algorithm

We describe the new affine equivalence algorithm below. Let $\boldsymbol{r_1} = (n+1-\gamma_n, n-\gamma_n)$ and $\boldsymbol{r_2} = (n, n-\gamma_n)$ for $\gamma_n = \lfloor (n/2)^{1/2} \rfloor$, as defined in Section 7.

1. Given $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^n$, $\boldsymbol{G} : \{0,1\}^n \to \{0,1\}^n$, compute their corresponding ANF representations $\boldsymbol{P}_{\geq(n-2)}$ and $\boldsymbol{Q}_{\geq(n-2)}$.
2. Run the algorithm of Section 8.1 to compute the rank table $\mathcal{T}_{\boldsymbol{F},n-2}$ and rank histogram $\mathcal{H}_{\boldsymbol{F},n-2}$ for $\boldsymbol{F}$ using $\boldsymbol{P}_{\geq(n-2)}$, and similarly compute $\mathcal{T}_{\boldsymbol{G},n-2}$ and $\mathcal{H}_{\boldsymbol{G},n-2}$ for $\boldsymbol{G}$. If $\mathcal{H}_{\boldsymbol{F},n-2} \neq \mathcal{H}_{\boldsymbol{G},n-2}$, return "Not Equivalent".
3. Run the unique HSM algorithm of Section 8.2 for $\boldsymbol{F}$ on inputs $\mathcal{T}_{\boldsymbol{F},n-2}$ and $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ defined above, and obtain the set $U_{\boldsymbol{F}}$. Similarly, obtain the set for $U_{\boldsymbol{G}}$ by running this algorithm on inputs $\mathcal{T}_{\boldsymbol{G},n-2}$ and $\boldsymbol{r_1}$, $\boldsymbol{r_2}$.
   If $|U_{\boldsymbol{F}}| \neq |U_{\boldsymbol{G}}|$, return "Not Equivalent". Otherwise, if $U_{\boldsymbol{F}}$ does not contain n linearly independent HSMs, return "Fail".
4. Run the affine transformation recovery algorithm of Section 8.3 on inputs $U_{\boldsymbol{F}}$ and $U_{\boldsymbol{G}}$. If it returns "Not Equivalent", return the same output. Otherwise, it returns a candidate for A_1.
5. Recover a candidate for $A_2 = L_2(x)+a_2$ by evaluating inputs $v \in \{0,1\}^n$ to $\boldsymbol{F} \circ A_1$ and $\boldsymbol{G}$: each input v gives n linear equations on L_2 and a_2. Hence, after a bit more than n evaluations, we expect the linear equation system to have a single solution which gives a candidate for A_2.

6. Test the candidates A_1, A_2 by equating the evaluations of $\boldsymbol{G}$ and $A_2 \circ \boldsymbol{F} \circ A_1$ on all 2^n possible inputs. If $\boldsymbol{G}(v) \neq A_2 \circ \boldsymbol{F} \circ A_1(v)$ for some $v \in \{0,1\}^n$, return "Not Equivalent". Otherwise, return A_1, A_2.

The correctness of the algorithm follows from the correctness of the subprocedures is executes and from the results obtained so far. In particular, Step 2 is correct according to Lemma 9, while the correctness of Step 3 is based on Theorem 2. The correctness of the final step is trivial.

Step 3 is the most complex to analyze in terms of success probability and complexity (which is the product $\mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r_1}) \cdot \mathcal{H}_{\boldsymbol{F},n-2}(\boldsymbol{r_2})$). We first focus on the time complexity analysis of the other steps.

The complexities of steps 4, 5 are at most polynomial in n and can be neglected. Step 1 interpolates the 2^n ANF coefficients for each of the n output bits of $\boldsymbol{F}, \boldsymbol{G}$. Each such interpolation can be performed in $n2^n$ bit operations using the Moebius transform [12]. Hence this step requires $n^2 2^n$ bit operations and 2^n function evaluations in total. The complexity of Step 2 was shown to be $n^3 2^n$ bit operations, while the complexity of Step 6 is 2^n function evaluations.

In total, the time complexity of the algorithm is at most $n^3 2^n$ bit operations and 2^n function evaluations, assuming that the complexity of Step 3 does not dominate the algorithm (as we show below).

The memory complexity is 2^n words of n bits, but it can be significantly reduced in some cases as described in Section 8.6.

Asymptotic Analysis of the Unique HSM Algorithm As in Section 7, denote $S_1 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_1})$ and $S_2 = \mathcal{H}_{\boldsymbol{F},d}(\boldsymbol{r_2})$ and recall that their expected values are $2^{\Theta(n^{1/2})}$ and $2^{n/2+O(n^{1/2})}$, respectively.

The expected asymptotic complexity of the unique HSM algorithm is therefore at most

$$S_1 \cdot S_2 = 2^{n/2+O(n^{1/2})} \ll 2^n.$$

Hence, the complexity of Step 3 is negligible compared to the complexity of the remaining steps of the affine equivalence algorithm described above.

According to the analysis of the unique HSM algorithm given in the extended version of this paper [9], the asymptotic lower bound on the expected number of unique HSMs in $\mathcal{HM}_{\boldsymbol{F},n-2,\boldsymbol{r},\boldsymbol{r'}}$ is

$$S_1 - (S_1)^2/\sqrt{S_2} > 2^{\Theta(n^{1/2})} - 2^{\Theta(n^{1/2})}/2^{n/4+O(n^{1/4})} = 2^{\Theta(n^{1/2})} \gg n.$$

Out of these unique HSMs, n are very likely to be linearly independent. This shows that asymptotically the algorithm succeeds with overwhelming probability.

We remark that there are many possible ways to select the rank group indexes $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ that give similar results.

8.5 Experimental results

In practice we do not pre-fix the rank groups of $\boldsymbol{F}, \boldsymbol{G}$ for which we run the unique HSM algorithm. Instead, we select a *reference rank group* $\boldsymbol{r'}$ such that $|\mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r'})| \approx 2^{n/2}$ (as $\boldsymbol{r_2}$ defined above). We then iterate over the rank groups $\boldsymbol{r}$ from the smallest to the largest, while collecting unique HSMs using repeated executions of the unique HSM algorithm with inputs $\boldsymbol{r}, \boldsymbol{r'}$. We stop once we collect n linearly independent unique HSMs. This practical variant is more flexible and succeeds given that the variant above succeeds.

We implemented the algorithm and tested it for various values of $8 \leq n \leq 28$. In each trial we first selected the permutation $\boldsymbol{F}$ uniformly at random. We then chose invertible affine mappings A_1, A_2 uniformly at random and defined $\boldsymbol{G} = A_2 \circ \boldsymbol{F} \circ A_1$. After calculating these inputs, we executed the algorithm and verified that it correctly recovered A_1, A_2.

Following this initial verification, our goal was to collect statistics that support the asymptotic complexity analysis of the unique HSM algorithm above. For this purpose, we selected the permutation $\boldsymbol{F}$ at random and calculated the success rate and complexity of Step 3, which executes the unique HSM algorithm on $\boldsymbol{F}$ (after running steps 1,2). If Step 3 succeeds to return n linearly independent unique HSMs with a certain complexity for $\boldsymbol{F}$, then it would succeed with identical complexity on any linearly equivalent $\boldsymbol{G}$, hence analyzing a single permutation is sufficient for the purpose of gathering statistics.

Our results for $n \in \{8, 12, 16, 20, 24, 28\}$ are summarized in Table 1. This table shows that all the trials for the various choices of n were successful. In terms of complexity, for $n = 8$, the unique HSM algorithm had to iterate over $2^{10.5} > 2^8$ HSMs on average in order to find 8 unique linearly independent HSMs. This relatively high complexity is due to the fact that our asymptotic analysis ignores constants whose effect is more pronounced for smaller values of n. Nevertheless, the complexity of the algorithm for $n = 8$ in terms of bit operations remains roughly $8^3 \cdot 2^8$, as the unique HSM algorithm does not perform linear algebra.

For $n \geq 12$, the average complexity of the unique HSM algorithm is below 2^n, and this gap increases as n grows (as predicted by the asymptotic analysis). Note that the complexity drops in two cases (between $n = 16$ and $n = 20$ and between $n = 24$ and $n = 28$) since for larger n we have more non-empty rank groups of various sizes and hence more flexibility in the algorithm (which happens to be quite substantial for $n = 20$ and $n = 28$). Finally, we note that we did not optimize the index $\boldsymbol{r'}$ of the reference rank group and better options that improve the complexities are likely to exist. However, since the unique HSM algorithm does not dominate the overall complexity, such improvements would have negligible effect.

In addition to the experiments on random permutations, we also performed simulations on random functions and obtained similar results.

8.6 Additional Variants of the New Affine Equivalence Algorithm

We describe several variants of the affine equivalence algorithm.

Table 1. Experimental Results for the Unique HSM Algorithm

n	Number of Trials	Number of Successful Trials	Average Complexity
8	1000	1000	$1486 \approx 2^{10.5}$
12	1000	1000	$3229 \approx 2^{11.7}$
16	1000	1000	$25599 \approx 2^{14.6}$
20	1000	1000	$15154 \approx 2^{13.9}$
24	1000	1000	$126777 \approx 2^{17}$
28	100	100	$40834 \approx 2^{15.3}$

Using Rank Group Sums The first additional variant we describe uses the rank tables of $\boldsymbol{F}, \boldsymbol{G}$ to directly recover several matchings in the initial stage of the algorithm. It is based on the observation that for each non-empty rank group $\boldsymbol{r}$, the HSM obtained by summing of all HSMs in $\mathcal{T}_{\boldsymbol{G},n-2}(\boldsymbol{r})$ has to match (under $\mapsto_{(A_1)}$) the HSM obtained by summming of all HSMs in $\mathcal{T}_{\boldsymbol{F},n-2}(\boldsymbol{r})$ due to the additive property of the HSM bijection. Simple analysis (based on Assumption 1 and backed up by experimental results) shows that the number of non-empty rank groups for a random permutation $\boldsymbol{F}$ is at least $n/4$ with very high probability. Hence we can initially recover at least $n/4$ matchings using this approach. There are several ways to recover the remaining matchings by exploiting the fact that we have essentially reduced the size of the problem from 2^n to at most $2^{3n/4}$. We can also continue in a similar way, further exploiting additive properties of the bijection: we take a uniquely matched HSM pair h, h'. For $\boldsymbol{G}$, we compute the *HSM rank table* for h with respect to some rank group $\boldsymbol{r'}$ by adding it to all HSMs in this group. We do the same for $\boldsymbol{F}$ by computing the HSM rank table of h' with respect to $\boldsymbol{r'}$. As in the initial observation, the sum of HSMs in each non-empty rank group of these smaller tables for $\boldsymbol{F}, \boldsymbol{G}$ match under $\mapsto_{(A_1)}$, revealing additional matchings. We repeat this process for several uniquely matched HSM pairs (computing additional HSM rank tables) until we identify the required n linearly independent matchings.

Reducing the Memory complexity The memory complexity of the algorithm is about 2^n words of n bits. If the functions $\boldsymbol{F}, \boldsymbol{G}$ are given as truth tables, then the memory complexity cannot be reduced by much. However, if we are given access to $\boldsymbol{F}, \boldsymbol{G}$ via oracles (e.g., they are implemented by block ciphers with a fixed key), then we can significantly reduce the memory complexity with no substantial effect on the time complexity.

First, instead of using the Moebius transform in Step 1 in order to interpolate all the coefficients of $\boldsymbol{F}, \boldsymbol{G}$, we simply interpolate each of the relevant $\approx n^2$ coefficients of degree at least $n-2$ independently, increasing the complexity of Step 1 by a factor of about n. Next, in Step 2 we do not store the entire rank table, but only the relevant rank groups with indexes $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$. As a result, we now have to recompute the ranks of $S_1 \cdot S_2$ HSMs in Step 3, but this requires much lower complexity than 2^n.

Overall, the memory complexity of this low-memory variant is dominated by the size of largest rank group stored in memory S_2, which is bit more than $2^{n/2}$. Finally, by a different choice of rank groups of indexes $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$, it is possible reduce the memory to be sub-exponential in n.

Multiple Solutions to the Affine Equivalence Problem Consider the case where there are two or more solutions of the form $(A_1^{(i)}, A_2^{(i)})$ to an instance of the affine equivalence problem $\boldsymbol{F}, \boldsymbol{G}$. This may occur (for example) if $\boldsymbol{F}$ is self-affine equivalent, namely, there exist (A_1, A_2) (that are not both identities) such that $\boldsymbol{F} = A_2 \circ \boldsymbol{F} \circ A_1$. We note that this case is extremely unlikely if $\boldsymbol{F}$ is chosen uniformly at random for $n \geq 8$, but it may occur for specific choices (e.g., the AES Sbox is self-affine equivalent).

In case of multiple solutions, a straightforward application of the affine equivalence algorithm would fail, as a HSM h would most likely match a different $h'^{(i)}$ for each solution $A_1^{(i)}$, namely $h \mapsto_{A_1^{(i)}} h'^{(i)}$. Consequently, we would not be able to find sufficiently many unique HSMs in Step 4. However, we can tweak the algorithm to deal with this case by working on each match $h \mapsto_{A_1^{(i)}} h'^{(i)}$ separately. More specifically, according to Lemma 6 we know that $\boldsymbol{F} \circ C_{|h'^{(i)},c} \equiv \boldsymbol{G} \circ C_{|h,0}$ and we can apply the algorithm recursively on these functions.

Affine Equivalences Among a Set of Functions We consider a generalization of the affine equivalence problem that was described in [3]. Given a set of K functions $\{\boldsymbol{F_i}\}_{i=1}^{K}$, our goal is to partition them into groups of affine equivalent functions. The naive approach is to run the affine equivalence algorithm on each pair of functions, which results in complexity of $K^2 \cdot n^3 2^n$.

We can improve this complexity by noticing that up to Step 4 of the affine equivalence algorithm the functions $\boldsymbol{F}, \boldsymbol{G}$ are analyzed independently. In particular, we can compute the rank histogram $\mathcal{H}_{\boldsymbol{F_i},n-2}$ for each function $\boldsymbol{F_i}$ independently (as done in Step 2) in time $n^3 2^n$, and then sort the functions and classify them according to their rank histograms.[9] This reduces the time complexity to about $K \cdot n^3 2^n + \tilde{O}(K^2)$ (where $\tilde{O}$ hides a small polynomial factor in n), improving upon the time complexity of $K \cdot n^3 2^{2n} + \tilde{O}(K^2)$, obtained in [3].

9 Applications

We describe applications of the affine equivalence algorithm and then focus on additional applications of the new objects and algorithms defined in this paper.

9.1 Applications of the New Affine Equivalence Algorithm

Algorithms for the affine equivalence problem are useful in several contexts such as classification of Sboxes [6, 14], producing equivalent representations of block

[9] We can also attach more data to each function by computing HSM histogram multisets between groups $\mathcal{HM}_{\boldsymbol{F_i},n-2,\boldsymbol{r},\boldsymbol{r'}}$.

ciphers [3] and attacking white-box ciphers [15]. In all of these contexts, if the goal is to apply the algorithm a few times to functions with a small domain size n, then the main algorithm of Biryukov et al. [3] is already practical and there is little to be gained by using our algorithm.

On the other hand, our algorithm may provide an advantage if the goal is to solve the affine equivalence problem on functions with a larger domain sizes (e.g., the domain size of the CAST Sbox [1] is $n = 32$). Furthermore, our algorithm may be beneficial if we need to solve the affine equivalence problem for many functions with domain size $n \geq 8$. For example, if we want to classify a large set of 8-bit Sboxes produced based on some design criteria, we can use the variant that searches for affine equivalences among a set of functions (described in Section 8.6).

An additional application (which is also described in [3]) is cryptanalysis of a generalization of the Even-Mansour scheme. The original scheme [11] builds a block cipher using a public permutation $\boldsymbol{F} : \{0,1\}^n \rightarrow \{0,1\}^n$ and a pair of n-bit keys k_1, k_2 by defining the encryption function on a plaintext $p \in \{0,1\}^n$ as $\boldsymbol{E}(p) = \boldsymbol{F}(p+k_1)+k_2$. Breaking the scheme may be considered as a special case of solving the affine equivalence problem where the linear matrices are identities. Thus, in the generalized scheme, arbitrary affine transformations A_1, A_2 are used as the key and the encryption function is defined as $\boldsymbol{E}(p) = A_2 \circ \boldsymbol{F} \circ A_1(p)$. Clearly, breaking the generalized Even-Mansour scheme reduces to solving the affine equivalence problem. The currently best know attack on this scheme (given in [3]) requires about $2^{3n/2}$ time and memory. It uses a birthday paradox based approach that generalizes Daemen's attack on the original Even-Mansour cipher [8]. Hence, we improve the complexity of the best known attack on the generalized Even-Mansour cipher from about $2^{3n/2}$ to 2^n.

9.2 Additional Applications

We describe additional applications of the rank table and histogram objects defined in this paper, and the algorithm used to compute them.

Application to Decomposition of the ASASA Construction The ASASA construction is an SP-network that consists of three secret affine layers (A) interleaved with two secret Sbox layers (S). At ASIACRYPT 2014, Biryukov et al. [2] proposed several concrete ASASA block cipher designs as candidates for white-box cryptography, whose security was based on the alleged difficulty of recovering their internal components. These designs were subsequently broken in [16] and [10].

Of particular interest is the integral attack of [10]. While the full details of this attack are out of the scope of this paper, we focus on its heaviest computational step that consists of summing over about 2^n affine subspaces of dimension slightly less than n (where n is the block size of the scheme). This step was performed in [10] in complexity of about $2^{3n/2}$. We can improve the complexity of this step (and the complexity of the full attack) to about 2^n by using a symbolic algorithm which is similar to the one used for computing the rank table.

Application to Distinguishers on Sboxes and Block Ciphers In [4] Biryukov and Perrin considered the problem of reverse-engineering Sboxes and proposed techniques to check whether a given Sbox was selected at random or was designed according to some unknown criteria. These techniques are based on the linear approximation table (LAT) and difference distribution table (DDT) of the Sbox. Here, we provide another method based on the distribution of entries in the rank histogram of the Sbox. More specifically, an Sbox would be considered suspicious if its rank histogram entry sizes differ significantly from their expected values according to the distribution derived in Section 7 (supported by the experimental results of the extended version of this paper [9]).

An advantage of our proposal is that the LAT and DDT require about 2^{2n} time and memory to compute and store, whereas the rank histogram can be computed in time of about 2^n. Hence, our proposal can be used to analyze larger Sboxes. We can also use additional properties of HSM rank histogram multi-sets (such as the number of unique HSMs) as possible distinguishing techniques.

In a related application, the rank table (and additional structures defined in this paper) can be used to experimentally construct distinguishers on block ciphers with a small block size (e.g., 32 bits). This is done be selecting a few keys for the block cipher at random and detecting consistent deviations from random among the resultant permutations. In particular, if there is a linear combination of the output bits that is a low-degree function of some $(n-1)$-dimensional input subspace, then we can detect it in time complexity of about 2^n. Since there are 2^{n+1} possible $(n-1)$-dimensional affine subspaces and 2^n linear combinations of output bits, we search over a space of 2^{2n+1} possible distinguishers in about 2^n time. This can be viewed as an improvement over known experimental methods [19] that search a much smaller space containing about n^2 potential high-order differential distinguishers in similar complexity (these methods only consider the input and output bits, but not their linear combinations). Finally, the technique can also be used on block ciphers with larger block sizes by considering linear subspaces of the input domain and output range.

10 Conclusions and Open Problems

In this paper we described an improved algorithm for the affine equivalence problem, focusing on randomly chosen permutations. The main open problem is to further improve the algorithm's complexity and applicability. An additional future work item is to find more applications for the rank table and related structures defined in this paper.

References

1. C. M. Adams. Constructing Symmetric Ciphers Using the CAST Design Procedure. *Des. Codes Cryptography*, 12(3):283–316, 1997.

2. A. Biryukov, C. Bouillaguet, and D. Khovratovich. Cryptographic Schemes Based on the ASASA Structure: Black-Box, White-Box, and Public-Key (Extended Abstract). In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 63–84. Springer, 2014.
3. A. Biryukov, C. D. Cannière, A. Braeken, and B. Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2003.
4. A. Biryukov and L. Perrin. On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 116–140. Springer, 2015.
5. C. Bouillaguet, P. Fouque, and A. Véber. Graph-Theoretic Algorithms for the "Isomorphism of Polynomials" Problem. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 211–227. Springer, 2013.
6. M. Brinkmann and G. Leander. On the classification of APN functions up to dimension five. *Des. Codes Cryptography*, 49(1-3):273–288, 2008.
7. A. Canteaut and J. Roué. On the Behaviors of Affine Equivalent Sboxes Regarding Differential and Linear Attacks. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 45–74. Springer, 2015.
8. J. Daemen. Limitations of the Even-Mansour Construction. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 495–498. Springer, 1991.
9. I. Dinur. An Improved Affine Equivalence Algorithm for Random Permutations. *IACR Cryptology ePrint Archive*, 2018:115, 2018.
10. I. Dinur, O. Dunkelman, T. Kranz, and G. Leander. Decomposing the ASASA Block Cipher Construction. *IACR Cryptology ePrint Archive*, 2015:507, 2015.
11. S. Even and Y. Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *J. Cryptology*, 10(3):151–162, 1997.
12. A. Joux. *Algorithmic Cryptanalysis*. Chapman & Hall, 2009.
13. V. F. Kolchin. *Random Graphs*. Cambridge Univ. Press, 1999.
14. G. Leander and A. Poschmann. On the Classification of 4 Bit S-Boxes. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer, 2007.

15. W. Michiels, P. Gorissen, and H. D. L. Hollmann. Cryptanalysis of a Generic Class of White-Box Implementations. In R. M. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2008.
16. B. Minaud, P. Derbez, P. Fouque, and P. Karpman. Key-Recovery Attacks on ASASA. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2015.
17. J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.
18. J. Patarin, L. Goubin, and N. Courtois. Improved Algorithms for Isomorphisms of Polynomials. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 184–200. Springer, 1998.
19. Q. Wang, Z. Liu, K. Varici, Y. Sasaki, V. Rijmen, and Y. Todo. Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In W. Meier and D. Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2014.