J. Cryptology (1998) 11: 125–140

Journal of
CRYPTOLOGY

The Black-Box Model for Cryptographic Primitives

Claus Peter Schnorr
FB Math./Inf., Universität Frankfurt,
Postfach 111932, 60054 Frankfurt a.M., Germany
Schnorr@informatik.uni-frankfurt.de

Serge Vaudenay
Ecole Normale Supérieure/DMI — CNRS,
45 rue d'Ulm, 75230 Paris cedex 5, France
Serge.Vaudenay@ens.fr

Communicated by Joan Feigenbaum

Received 25 October 1995 and revised 15 September 1997

Abstract. We introduce the *black-box model* for cryptographic primitives. In this model cryptographic primitives are given by a computation graph, where the computation boxes sitting on the vertices of the graph act as random oracles. We formalize and study a family of generic attacks which generalize exhaustive search and the birthday paradox. We establish complexity lower bounds for these attacks and we apply it to compression functions based on the FFT network.

Key words. Cryptographic primitives, Generic attacks.

Introduction

Cryptographic primitives for encryption, hashing, and pseudorandom generation are judged according to efficiency and security. Design methods for constructing cryptographic primitives are usually empiristic. The example of Rivest's MD4 hash function [11], which has been shown to be insecure by Dobbertin [7], demonstrates the need for a design theory that provides security in a realistic model.

Usually cryptographic primitives are defined as a computation graph in which the vertices are computation boxes. The cryptanalysis approach of Biham and Shamir [6] and Matsui [9] initiated an important study of the algebraic properties of the computation boxes. In this paper we take another view, we neglect the inner structure of the boxes. We study the security provided by a computation graph, where the boxes act as random oracles, i.e., as *black-boxes*.

Generic attacks that do not exploit the inner structure of the boxes play an important role. Nechaev [10] and Shoup [16] study generic algorithms for the discrete logarithm, assuming that the group operations are given as black-boxes. The random oracle model of

Bellare and Rogaway [5] has been used in security proofs for various signature schemes. Here the hash function is a black-box acting as a random oracle in a network comprising the signer, the verifier, and the attacker of a signature scheme. We propose a black-box model in which *all* boxes of the computation graph act as random oracles.

The black-box model covers powerful attacks, e.g., the iterative use of exhaustive search and the birthday paradox applied locally to arbitrary parts of the computation graph. We only exclude attacks that "split" the boxes. In the black-box model we can prove interesting and tight complexity bounds for generic attacks. These complexities correspond to the minimal workload of attacks. We study the average complexity of these attacks for relevant probability distributions for the boxes. Black-box cryptanalysis can determine optimal interconnection networks for the design of hash functions and symmetric encryption functions provided that strong computation boxes are given.

In particular we extend and analyze the FFT network, the computation graph of the Fast Fourier Transform that is known as the butterfly graph. This network has been used in several cryptographic proposals [8], [14], [15]. We give evidence that the FFT network with 2^k input nodes and $2k-2$ layers yields a family of compression functions with optimal security in the black-box model. This means that for the doubled FFT network there is no better black-box attack than exhaustive search over the inputs. While our lower and upper complexity bounds coincide for *linear* attacks, lower and upper bounds differ by a factor of 3 in the general case.

In Section 1 we present the black-box model. In Section 2 we consider the FFT hash network. We prove upper bounds and lower bounds for the complexity of inverting the function computed by this network. The particular case of linear algorithms for the FFT network has been studied in [15]. Here we prove lower bounds for general algorithms in the black-box model. A full formal study is also available in [20, pp. 31–87].

1. The Black-Box Model

1.1. *Computation Graphs with Random Boxes*

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E.[1] A "computation" along G associates with each edge a value in some finite *alphabet* Z. Associated with each vertex v is a set of possible local computations (or local solutions) $I(v) \subset Z^{E(v)}$ for $E(v)$, the set of edges adjacent to v. Thus $I(v)$ is the set of assignments of values in Z to the edges in $E(v)$ that are admissible for the box at v. For the degree $d(v) = \#E(v)$[2] of vertex v we must have $\#I(v) \leq \#Z^{d(v)}$. If $\#I(v) = \#Z^{d(v)}$ the box is trivial as all assignments are admissible. If vertex v has i "input edges," then $\#I(v) = \#Z^i$ since the input values determine the output values. A *solution* for the graph is a tuple $t \in Z^E$, i.e., a tuple on E such that, for all vertices v, the restriction $t_{|E(v)}$[3] is in $I(v)$.

We call I—the collection of all local solutions—an *interpretation* of G. We study resolution algorithms that work in general, i.e., for random local solutions. We study

[1] All graphs are finite and loop-free in the paper.

[2] The symbol # denotes the cardinality of a set.

[3] The notation $t_{|E'}$ is the restriction of the tuple t to the subset E', i.e., the tuple on E' which is equal to t on all entries in E'.

the average complexity of these algorithms. We associate with each vertex v a *degree of freedom* $\mathrm{df}(v)$—informally the (fractional) number of independent values in Z that appear in $I(v)$—which we define as $\log_{\#Z} \mathrm{Exp}_I \#I(v)$.[4] In the following all logarithms have the basis $\#Z$ and log means $\log_{\#Z}$.

Definition 1. A *computation graph* (G^{df}, Z) consists of an undirected graph $G = (V, E)$, a real-valued function $\mathrm{df}(v)$ satisfying $\mathrm{df}(v) \leq d(v)$, and an alphabet Z. A *random interpretation* I is a random map which associates with each vertex v a set $I(v) \subset Z^{E(v)}$ of local solutions so that $\mathrm{df}(v) = \log \mathrm{Exp}_I \#I(v)$.

The computation graph G^{df} is undirected and so is the "computation flow." To stress the undirected nature of G^{df} it was called an *equation graph* rather than a *computation graph* in [20].

In the following we assume that all probability distributions for I have the following two properties:

Local Uniformity. For all $v \in V$ and $t \in Z^{E(v)}$, the probability $\Pr[t \in I(v)]$ is a constant which depends on v. Thus we have $\Pr[t \in I(v)] = \#Z^{\mathrm{df}(v)-d(v)}$.

Independence. The sets $I(v)$ are independent for distinct vertices v.

Examples of possible distributions are:

— the uniform distribution over all I so that $I(v)$ is a subset of $Z^{E(v)}$ with $\#I(v) = \#Z^{\mathrm{df}(v)}$;
— for integer degree of freedom $\mathrm{df}(v)$, the uniform distribution over all I so that $I(v)$ defines a function of edge values of some $\mathrm{df}(v)$ edges in $E(v)$ to the other $d(v) - \mathrm{df}(v)$ edge values;
— the uniform distribution over all I so that $I(v)$ defines a *multipermutation* on $Z^{E(v)}$. (Following Vaudenay [19], [20], a multipermutation with r inputs and n outputs is a set of $(r+n)$-tuples such that no different tuples t_1 and t_2 can be simultaneously equal on any r different entries. This way, it is a function of any r entries onto the remaining n ones. This concept formalizes the notion of perfect diffusion, that all inputs are perfectly diffused to the outputs in an information theoretic sense. This is a useful design criterion.)

1.2. *Resolution and Complexity of a Computation Graph*

For two subsets $E', E'' \subset E$ of edges and corresponding sets of tuples $X' \subset Z^{E'}$, $X'' \subset Z^{E''}$ we define the *join*

$$X' \bowtie X'' = \{t \in Z^{E' \cup E''} \mid t_{|E'} \in X',\ t_{|E''} \in X''\}$$

to be the set of all tuples t on $E' \cup E''$ with restrictions to E' in X' and to E'' in X''. This operation $\bowtie$ is similar to the join used in relational database theory. The join is

[4] Here Exp_I denotes the expected value over the distribution of I.

associative and commutative and the set of *global* solutions on E turns out to be the join of all $I(v)$.

We extend the interpretation I to arbitrary subsets of vertices using that the join is associative and commutative. For $U \subset V$ we let $E(U)$ denote the set of edges adjacent to the vertices in U. We define $I(U)$ to be the join of all $I(v)$ with $v \in U$, i.e.,

$$I(U) = \{t \in Z^{E(U)} \mid t_{|E(v)} \in I(v) \text{ for all } v \in U\}.$$

$I(U)$ is the set of *local* solutions for U. Obviously $I(U \cup W) = I(U) \bowtie I(W)$ holds for arbitrary subsets U and W of vertices.

Definition 2. A (resolution) *algorithm* A for the graph G is a term A with the two-ary operation $\bowtie$ and all v in V. Its *length* $|A|$ is the number of occurrences of $\bowtie$ plus the number of occurrences of v's.

Actually, a resolution algorithm specifies the order of all the $\bowtie$ operations starting from the $I(v)$ with $v \in V$. For example, the term $(v_1 \bowtie v_2) \bowtie (v_3 \bowtie v_4)$ means that we first form $I(v_1) \bowtie I(v_2)$, $I(v_3) \bowtie I(v_4)$, and then the join of these two sets. There is a natural notion of subterm, the above term has the subterms $v_1 \bowtie v_2$, $v_3 \bowtie v_4$. We write $B \leq A$ if B is subterm of A.

For an arbitrary subterm B of an algorithm A let $V(B)$ denote the set of vertices occurring in B. So $I(V(B))$ is the result, or the set of local solutions, of the subterm B.

We define the logarithmic *complexity* $C_I(A)$ of an algorithm A to be the maximal logarithmic size of the result of a subterm $B \leq A$ (taking the maximum yields a simple and clean measure that differs from an average/aggregate measure only by $\log|A|$):

$$C_I(A) = \log \max_{B \leq A} \#I(V(B)).$$

The logarithmic complexity roughly corresponds to a *workload* $\#Z^{C_I(A)}$. Counting only the size of the largest intermediate result is justified since the length of algorithms will be small and the costs for a join operation corresponds to the cardinality of the operands.

The *average complexity* $C(A^{\mathrm{df}})$ of algorithm A is defined to be

$$C(A^{\mathrm{df}}) = \log \mathrm{Exp}_I \max_{B \leq A} \#I(V(B)).$$

As the distribution of I is locally uniform, the expected value Exp_I depends on df and so does $C(A^{\mathrm{df}})$. We let the *complexity* $C(G^{\mathrm{df}})$ of a computation graph G^{df} be the minimum of $C(A^{\mathrm{df}})$ over all algorithms A for G^{df}.

1.2.1. *Getting Only One Solution*

In most cases we are only interested in getting on the average *one* solution of G^{df}. If there are many solutions we can decrease the complexity by restricting the resolution process to random subsets of all the $I(v)$. Thus, for a given mapping df', with $\mathrm{df}'(v) \leq \mathrm{df}(v)$ for all v, and a given interpretation I of (G^{df}, Z), we consider a random subinterpretation $I^{\mathrm{df}'}$ and a distribution defined by picking independently and uniformly random subsets $I^{\mathrm{df}'}(v)$ of $I(v)$ of size $\#Z^{\mathrm{df}'(v)}$. This means considering a computation graph with df' instead of df. Note that the construction of $I^{\mathrm{df}'}$ preserves local uniformity and independence. In the following we consider computation graphs with only one solution on the average.

1.2.2. *Examples of Algorithms*

We demonstrate how to use exhaustive search and the birthday paradox iteratively on parts of the computation graph. Enumerating all values of a function f is formalized by the algorithm $x^1 \bowtie f^1$. Solving exhaustively $f(x) = a$ for given f, a is the intention of $(x^1 \bowtie f^1) \bowtie a^0$, where degrees of freedom $\mathrm{df}(v)$ are denoted by a superscript on v. Searching for *one* solution of $f(x, y) = a$ by picking x and y at random is the meaning of $((x^{1/2} \bowtie y^{1/2}) \bowtie f^1) \bowtie a^0$. Here the degree of freedom of x and y have been decreased to $\frac{1}{2}$ to get one solution on the average. We can also decrease it as $((x^0 \bowtie y^1) \bowtie f^1) \bowtie a^0$ which means to fix x arbitrarily and then to search for y. Using the birthday paradox to get one solution of $f(x) = g(y)$ can be written as $(x^{1/2} \bowtie f^1) \bowtie (y^{1/2} \bowtie g^1)$. This algorithm makes two lists of $\sqrt{\#Z}$ x- and y-values and searches for matches $f(x) = g(y)$.

1.2.3. *Linear Complexity*

In [15] we only considered *linear* resolution algorithms, i.e., algorithms of the form $v_1 \bowtie (v_2 \bowtie (\cdots))$ that are characterized by the order of traversing the vertices $v_1, \ldots, v_n$. It is tempting to believe that linear algorithms are already the most powerful ones and that nothing can be gained from nonlinear ones. The following counterexample shows this is not the case. Consider the network of Fig. 1 representing the computation graph of a supposed one-way function which maps six inputs $x_1, \ldots, x_6$ onto six outputs, all 24-bits long. The direction of the edges in Fig. 1 indicates the underlying network for computing the function. The problem of inverting this function is defined by the following computation graph:

$$V = \{x_1, \ldots, x_6, v_1, \ldots, v_6, w_1, \ldots, w_6\}$$

and

$$\mathrm{df}(x_i) = 1, \quad \mathrm{df}(v_i) = 2, \quad \mathrm{df}(w_i) = 1, \qquad i = 1, \ldots, 6.$$

It is straightforward to imagine a linear attack to invert the function with logarithmic complexity 3, that is within workload 2^{72}: enumerate x_1, x_3 and x_5 exhaustively and solve the leftmost third of the graph from x_1 and x_3 and get x_2 and x_4 as

$$A = ((((((x_1 \bowtie x_3) \bowtie v_1) \bowtie w_1) \bowtie w_2) \bowtie v_2) \bowtie x_2) \bowtie x_4.$$

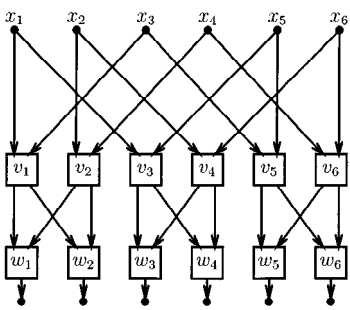

Fig. 1. A powerful nonlinear algorithm.

Then solve the rightmost third from x_5 and get x_6 by the algorithm

$$A' = (((((A \bowtie x_5) \bowtie v_5) \bowtie w_5) \bowtie w_6) \bowtie v_6) \bowtie x_6.$$

Finally check that this yields a solution of the whole graph via

$$A'' = (((A' \bowtie v_3) \bowtie w_3) \bowtie w_4) \bowtie v_4.$$

It is easy to see that 3 is the lowest complexity for linear algorithms. On the other hand, one can solve the leftmost third from x_1 and x_3 by the algorithm A and *independently* solve the rightmost third from x_3 and x_5 by

$$B = ((((((x_3 \bowtie x_5) \bowtie v_5) \bowtie w_5) \bowtie w_6) \bowtie v_6) \bowtie x_4) \bowtie x_6$$

then join the two sets of partial solutions to get a set $A \bowtie B$ with rank 2 and finally check whether it contains a global solution

$$((((A \bowtie B) \bowtie v_3) \bowtie w_3) \bowtie w_4) \bowtie v_4.$$

This is done with a logarithmic complexity 2, that is with workload 2^{48}.

1.3. *Approximating the Complexity*

With a computation graph G^{df} we associate its *quadratic form*, which we also denote G^{df}, represented by the symmetric matrix $(G^{\mathrm{df}}_{v,w})_{v,w\in V}$, where

$$G^{\mathrm{df}}_{v,w} =_{\mathrm{def}} \begin{cases} -\frac{1}{2} & \text{if} \quad v \neq w \quad \text{and} \quad \{v,w\} \in E, \\ \mathrm{df}(v) & \text{if} \quad v = w, \\ 0 & \text{otherwise.} \end{cases}$$

The quadratic form induces a function $G^{\mathrm{df}}\colon \mathbf{Z}^V \to \mathbf{Z}$ as

$$G^{\mathrm{df}}(g) = \sum_{v\in V} \sum_{w\in V} g(v)g(w)G^{\mathrm{df}}_{v,w}$$

for $g \in \mathbf{Z}^V$. We identify a subset $U \subset V$ with its characteristic function in $\{0,1\}^V$, and thus

$$G^{\mathrm{df}}(U) = \sum_{v\in U} \sum_{w\in U} G^{\mathrm{df}}_{v,w}.$$

Let $\mathrm{Int}(U) = \{\{v,w\} \in E \mid v,w \in U\}$ be the set of interior edges of U. It can easily be seen that

$$G^{\mathrm{df}}(U) = \sum_{v\in U} \mathrm{df}(v) - \#\mathrm{Int}(U) = \#E(U) + \sum_{v\in U} (\mathrm{df}(v) - d(v)),$$

where the latter equality comes from $\sum_{v\in U} d(v) = \#E(U) + \#\mathrm{Int}(U)$.

If $\mathrm{df}(v) = \frac{1}{2}d(v)$ holds for all vertices v, then, for any subset U of vertices, $G^{\mathrm{df}}(U)$ equals half the number of edges of the perimeter of U (i.e., the edges between U and $V - U$). We call this the *locally invertible* case as all the boxes look like permutations with the same number of inputs and outputs.

The role of the quadratic form becomes apparent in the following lemma.

Lemma 3. *Every subset $U \subset V$ of vertices satisfies* $\log \mathrm{Exp}_I \, \#I(U) = G^{\mathrm{df}}(U)$.

Proof. We note that $\mathrm{Exp}_I \#I(U) = \sum_t \Pr[t \in I(U)]$ where the sum is over all tuples t on $E(U)$. The event $\{t \in I(U)\}$ is the intersection of all the independent events $\{t_{|E(v)} \in I(v)\}$ for $v \in U$. There are $\#Z^{\#E(U)}$ many tuples on $E(U)$. Local uniformity and independence of the distribution for I yields

$$\log \mathrm{Exp}_I \, \#I(U) = \#E(U) + \sum_{v \in U} (\mathrm{df}(v) - d(v))$$

which, as we have already seen, equals $G^{\mathrm{df}}(U)$. □

Theorem 4. *Every algorithm A for G^{df} with length $|A|$ satisfies*

$$\max_{B \le A} G^{\mathrm{df}}(V(B)) \le C(A^{\mathrm{df}}) \le \log |A| + \max_{B \le A} G^{\mathrm{df}}(V(B)).$$

Proof. By the definition of $C(A^{\mathrm{df}})$, and since a maximum of nonnegative values is lower than their sum we have

$$\begin{aligned}
C(A^{\mathrm{df}}) &= \log \mathrm{Exp}_I \max_{B \le A} \#I(V(B)) \\
&\le \log \mathrm{Exp}_I \sum_{B \le A} \#I(V(B)) \\
&= \log \sum_{B \le A} \mathrm{Exp}_I \, \#I(V(B)) \\
&\le \log \left(|A| \cdot \max_{B \le A} \mathrm{Exp}_I \, \#I(V(B)) \right) \\
&= \log|A| + \max_{B \le A} \log \mathrm{Exp}_I \, \#I(V(B)) \\
&= \log|A| + \max_{B \le A} G^{\mathrm{df}}(V(B)),
\end{aligned}$$

where the last equality comes from Lemma 3.

The first inequality of the claim is straightforward by Lemma 3. □

As an immediate consequence of Theorem 4 the expression

$$C'(G^{\mathrm{df}}) := \min_A \max_{B \le A} G^{\mathrm{df}}(V(B))$$

is a close approximation to the complexity $C(G^{\mathrm{df}})$ which only depends on the quadratic form G^{df} and not on Z.

Corollary 5. $C'(G^{\mathrm{df}}) \le C(G^{\mathrm{df}}) \le \log|A_{\mathrm{opt}}| + C'(G^{\mathrm{df}})$, *where A_{opt} is an optimal algorithm.*

1.4. *The Spectral Approach*

We consider a locally invertible graph $G^{d/2}$ so that $\mathrm{df}(v) = \frac{1}{2}d(v)$. Its quadratic form turns out to have properties similar to the Laplacian operator. In this context, lower bounds on the complexity can be proven in a similar way to the expander graphs theory using a well-known link with the spectral values [17], [2], [1]. In this section let G be an undirected graph with n vertices and let $\lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of the quadratic form $G^{d/2}$.

Lemma 6. *If the graph G is connected, then $\lambda_1 = 0$, $\lambda_2 > 0$, and every set U of c vertices satisfies $G^{d/2}(U) \geq \lambda_2\, c\,(1 - c/n)$.*

Proof. 0 is an eigenvalue since $G^{d/2}(U) = 0$ holds for all connected components U of G. The fact that the quadratic form is positive and that $\lambda_2 > 0$ (if G is connected) is an easy algebra exercise. We note that λ_2 is the smallest eigenvalue of the quadratic form in the hyperplane $V^\perp$ orthogonal to the set V of all the vertices (i.e., the vector having all coordinates 1).

For an orthonormal basis of eigenvectors $v_1, \ldots, v_n$ with $v_1 = (1/\sqrt{n})V$ we have (the dot $\cdot$ denotes the scalar product)

$$G^{d/2}(U) = \sum_{i=1}^{n} \lambda_i (U \cdot v_i)^2 \geq \lambda_2 \sum_{i=2}^{n} (U \cdot v_i)^2 = \lambda_2((U \cdot U) - (U \cdot v_1)^2).$$

Now, the claim follows from $U \cdot U = c$ and $U \cdot v_1 = c/\sqrt{n}$. □

Then we get a lower bound:

Theorem 7. *If the graph G is connected, then $C(G^{d/2}) \geq (2\lambda_2/9)n$.*

Proof. Let A be an algorithm for $G^{d/2}$ such that $C'(G^{d/2}) = \max_{B \leq A} G^{d/2}(V(B))$. Lemma 6 yields $C'(G^{d/2}) \geq \max_{B \leq A} \lambda_2 \#(V(B))(1 - \#(V(B))/n)$. Let x be an arbitrary integer between 0 and $n = \#V$. Every minimal subterm B with the property that $\#(V(B)) \geq x$ also satisfies $\#(V(B)) \leq 2x$ since it can at best be the join of two subterms which each contain $x - 1$ vertices. For such a subterm we have

$$\#(V(B)) \left(1 - \frac{\#(V(B))}{n}\right) \geq \min_{x \leq y \leq 2x} y\left(1 - \frac{y}{n}\right)$$

and thus by Corollary 5

$$C(G^{d/2}) \geq C'(G^{d/2}) \geq \max_{0 \leq x \leq n} \min_{x \leq y \leq 2x} \lambda_2\, y\left(1 - \frac{y}{n}\right) = \frac{2\lambda_2}{9} n.$$

The max min is obtain with $x = y = n/3$. □

1.5. *The Symmetric Approach*

Similarly, we can apply a theorem due to Babai and Szegedy [3] used in context with Cayley graphs. We reformulate it in our context and refer to [3] for the proof. Let G be an arbitrary undirected, edge-transitive graph with n vertices, let d be the harmonic mean of the degrees of the vertices and let δ be the average distance between two vertices.

Lemma 8 [3]. *Every set U of c vertices satisfies*

$$G^{d/2}(U) \geq \frac{d}{2\delta} c \left(1 - \frac{c}{n}\right).$$

A graph is edge-transitive if every edge can be mapped onto any other one by a graph automorphism. In such a graph the vertices can only have one or two possible degrees. This suggests using the harmonic mean $2/(1/d_1 + 1/d_2)$ of the two degrees d_1 and d_2. A straightforward application of the proof of Theorem 7 to Lemma 8 shows:

Theorem 9. *Every undirected, edge-transitive graph G satisfies $C(G^{d/2}) \geq (d/9\delta)n$.*

2. Parallel FFT Hashing

Two previous proposals of a cryptographic hash function based on the FFT network [12], [13] have been broken (see [4] and [18]). Then, by a joint effort, a family of hash functions based on the same graph has been proposed in [14] and discussed in [15]. We now prove the conjectures announced in the latter paper. Interestingly, the FFT network has independently been used by Massey for the SAFER encryption function [8].

Let $G_{k,s}$ be the graph defined by the set of vertices

$$V = \{v_{i,j}; 0 \leq i < 2^{k-1}, 0 \leq j \leq s\}$$

and the set of edges

$$E = \{\{v_{i,j}, v_{i,j+1}\}, \{v_{i,j}, v_{i \oplus 2^{j \bmod k-1}, j+1}\}; 0 \leq i < 2^{k-1}, 0 \leq j < s\}.$$

$G_{k,s}$ is roughly the graph of the FFT network for 2^k values extended to $s+1$ layers. Considering all vertices as boxes with two inputs coming from a lower layer and two

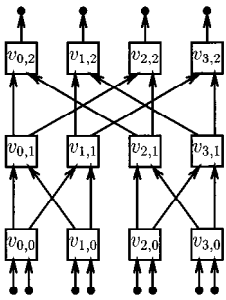

Fig. 2. The $G_{3,2}$ compression functions family.

outputs going to a higher layer, this corresponds to a function with 2^k inputs entering in layer 0 and 2^{k-1} outputs going out from layer s. Given two message blocks m and m' with 2^{k-1} values, we let the ith value of each block enter into vertex $v_{i,0}$ and write the first output of $v_{i,s}$ as the ith value of the output string h. The mapping $(m, m') \mapsto h$ defines a compression function. We propose studying the family of the compression functions defined by $G_{k,s}$ and a relevant distribution of interpretations I that defines the boxes. In [14] we considered the uniform distribution on all multipermutations. The aim of the present study is to find, by a graph theoretic analysis of $G_{k,s}$, the minimal s for optimal security in the context of black-box cryptanalysis.

The one-wayness of the compression function means hardness of finding, for *given* m and h, one m' such that $(m, m') \mapsto h$. (Note: finding for given h *some* m, m' with $(m, m') \mapsto h$ is trivial as we can arbitrarily complement the "half"-output h to (h, h') and compute the inverse permutation $(h, h') \mapsto (m, m')$.) The inversion problem is defined by the computation graph $G_{k,s}$ together with

$$\text{df: } v_{i,j} \mapsto \begin{cases} 1 & \text{if} \quad j = 0 \quad \text{or} \quad j = s, \\ 2 & \text{otherwise.} \end{cases}$$

Here one input of the $v_{i,0}$ and one output of the $v_{i,s}$ are already known, that is $\mathrm{df} = \frac{1}{2}d = 1$ holds for the first and the last layer. The exhaustive search consists in joining all $v_{i,0}$ to guess m' and joining successively all the other vertices layer by layer. This has complexity 2^{k-1}. So, we are interested in the *ratio* $C(G_{k,s}^{d/2})/2^{k-1}$.

Finally consider the length of the bit string for specifying the $(s+1)2^{k-1}$ boxes. Choosing an alphabet with cardinality q, the number of bits to encode the input is $n = 2^k \log q$ whereas the length of the description of the function (that is the interpretation) is $(s+1)2^k q^2 \log q$. The family is quite huge, but we hope to find an interesting smaller subfamily in which the following analysis will be possible too. For instance, if we take the same box for all the vertices i, j and concatenate these boxes with independent random permutations along the inner edges of $G_{k,s}$ we decrease the length of the interpretation to $s2^k \log q!$ and we preserve local uniformity and independence.

2.1. *The Upper Bounds*

Theorem 10. *For $k - 1 \le s \le 2k - 2$ we have $C(G_{k,s}^{d/2}) \le 2^{k-2}(1 + 2^{2-s})$.*

Thus, for $s < 2k - 2$ there is an attack faster than exhaustive search. We conjecture that this inequality is in fact an equality for $s = 2k - 2$, that is to say the exhaustive search is the best black-box attack on $G_{k,2k-2}$.

Proof. First we show that $C(G_{k,k-1}^{d/2}) \le 2^{k-2}$. For this we guess the first 2^{k-2} inputs, that is we join the first 2^{k-2} vertices $v_{i,0}$. This allows us to compute half of the edges, namely all edges adjacent to $v_{i,j}$ for $i < 2^{k-2}$. Then the degree of freedom of all $v_{i,k-1}$ becomes 0, so that we can compute the other half of the edges backward and solve the graph. This has complexity 2^{k-2}.

We can do similar things on $G_{k,s}$: we guess the first 2^{k-2} inputs and solve half of the vertices from layer 0 to layer $k - 1$. Then all connected subgraphs from layer $k - 1$ to

layer s are isomorphic to $G_{k',k'-1}$ with $k' = s - k + 2$. We solve them iteratively within a complexity $2^{k'-2} = 2^{s-k}$. Having solved all of these subgraphs, we backwardly process $G_{k,s}$. The resolution has complexity $2^{k-2} + 2^{k-s}$. □

In the following sections we show how to apply the different approaches to find lower bounds. Finally, we prove that the ratio for $s = 2k - 2$ is lower bounded by a constant $\frac{2}{3}$.

2.2. *The Lower Bounds*

2.2.1. *Use of the Spectral Approach*

We use the notation introduced in Sections 1.4 and 2.

Lemma 11.

$$\lambda_2\left(G_{k,s}^{d/2}\right) = \begin{cases} 4\sin^2\dfrac{\pi}{2(2k-1)} & \text{if} \quad k \le s < 2k-1, \\ 4\sin^2\dfrac{\pi}{2(s+1)} & \text{if} \quad 2k-1 \le s. \end{cases}$$

The proof is a difficult but unenlightening exercise in calculus which can be found in [20, pp. 76–80].

Proof. (sketch) We study the spectral properties of the adjacency matrix

$$M_{k,s} = \begin{pmatrix} I & -\frac{1}{2}A & 0 & \cdots & 0 & 0 \\ -\frac{1}{2}{}^tA & 2I & -\frac{1}{2}A & \cdots & 0 & 0 \\ 0 & -\frac{1}{2}{}^tA & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -\frac{1}{2}A \\ 0 & 0 & 0 & \cdots & -\frac{1}{2}{}^tA & I \end{pmatrix},$$

where I is the identity matrix and

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

We let f_α denote the Hadamard vector basis: for any boolean vector α with $k-1$ coordinates, f_α is a real vector with 2^{k-1} entries. Each entry index corresponds to a boolean vector i with $k-1$ coordinates and f_α is defined by $(f_\alpha)_i = (-1)^{\alpha \cdot i}$ where $\cdot$

is the dot product. In the basis of all block vectors $(0, \ldots, 0, f_\alpha, 0, \ldots, 0)$, the matrix $M_{k,s}$ is block-diagonal with $(s+1)m \times (s+1)m$-blocks of the form

$$B_s(J) = \begin{pmatrix} I & -J & 0 & \cdots & 0 & 0 \\ -{}^{\mathrm{t}}J & 2I & -J & \cdots & 0 & 0 \\ 0 & -{}^{\mathrm{t}}J & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -J \\ 0 & 0 & 0 & \cdots & -{}^{\mathrm{t}}J & I \end{pmatrix},$$

where J is a Jordan $m \times m$-block and ${}^{\mathrm{t}}J$ its transpose. Studying the spectral properties of the $B_s(J)$ is a technical algebra exercise. □

Then Theorem 7 implies:

Corollary 12.

$$C\left(G_{k,s}^{d/2}\right) \geq 2^{k-1} \begin{cases} \dfrac{8(s+1)}{9} \sin^2 \dfrac{\pi}{2(2k-1)} & \text{if} \quad k \leq s < 2k-1, \\ \dfrac{8(s+1)}{9} \sin^2 \dfrac{\pi}{2(s+1)} & \text{if} \quad 2k-1 \leq s. \end{cases}$$

This suggests using $s = 2k-1$ to get optimal security. The lower bound of the ratio is here equivalent to $2\pi^2/9s$.

2.2.2. *Use of the Symmetric Approach*

Though $G_{k,s}$ is not edge-transitive, it is possible to use the symmetric approach for the case $s = k$. If we contract layers 0 and 1 following the rule that adjacent edges are merged, we get the same result (i.e., isomorphic) as if we add symmetrical edges between the first and the last edges of $G_{k-1,2k-5}$. The obtained graph $G'_{k-1,2k-5}$ turns out to be edge-transitive for $k \geq 3$. This graph has degree $d = 4$.

Lemma 13. *The average distance of $G'_{k-1,2k-5}$ is asymptotically $\frac{3}{2}(k-2)$.*

Proof. (sketch) By studying the distance between two vertices, we obtain that the average distance is

$$\delta = \tfrac{3}{2}(k-2) + \frac{C_{k-2}}{k-2} - \frac{2}{k-2} \sum_{i=1}^{k-2} C_i,$$

where C_i is the average length of the longest all-zero subsequence of a random boolean sequence of length i. Then we prove that $C_i \leq 1 + \log_2 i$. So, $\delta \sim \frac{3}{2}(k-2)$. The complete proof can be found in [20, pp. 74–76]. □

We let $C_{\mathrm{lin}}(G_{k,k}^{d/2})$ denotes the linear complexity of $G_{k,k}^{d/2}$, that is

$$C_{\mathrm{lin}}(G_{k,k}^{d/2}) = \min_A C(A^{\mathrm{df}})$$

over all linear algorithms A. The last lemma allows us to prove a technical corollary we mention for completeness.

Corollary 14. *For $k \geq 3$ we obtain $C_{\mathrm{lin}}(G_{k,k}^{d/2}) \geq (2^{k-1}/3)(1 + o(1))$.*

This establishes the lower bound $\frac{1}{3}$ for the ratio. Unfortunately, we could not prove a similar result for the general complexity following this approach.

Proof. (sketch) Any linear algorithm A can be rewritten, without increasing its complexity, as an algorithm such that, for any subterm B which involves a vertex $v_{i,j}$ for $j = 0, 1, s-1$, or s, every one of the other vertices which are merged with $v_{i,j}$ either already occurs in B or will be joined immediately hereafter. In such an algorithm, there is at least one out of four consecutive subterms B which has all of its merged *classes complete*, i.e., closed with respect to merging. Let B' be the merged image of such a B in $G'_{k-1,2k-5}$. The completeness property of B implies $G_{k,k}(B) = G'_{k-1,2k-5}(B')$. Thus, using the Babai–Szegedy theorem together with the previous lemma, we obtain

$$G_{k,k}^{d/2}(V(B)) \geq \frac{4}{3(k-2)} \#(V(B')) \left(1 - \frac{\#(V(B'))}{(2k-4)2^{k-2}}\right).$$

We observe that

$$\#(V(B)) - 3 \cdot 2^{k-1} \leq \#(V(B')) \leq \#(V(B))$$

and that at least one out of four consecutive B's satisfies the *completeness* property, hence the result. □

2.2.3. *Use of the Flow Approach*

In the particular case of the graph $G_{k,2k-2}$, there is an independent method, dedicated to this graph and based on the min-cut max-flow theorem.

Lemma 15. *Let V_0 and V_s be respectively the first and the last layer of $G_{k,2k-2}$. For every function r from $V_0 \cup V_s$ to $\mathbf{Z}$ such that $|r(v)| \leq 2$ and $\sum_v r(v) = 0$ there exists a flow f with source r, that is to say a function from the set E, the set of directed edges, to $\mathbf{Z}$ such that, for all v and w:*

1. $f(v, w) = -f(w, v)$,
2. $|f(v, w)| \leq 1$,
3. $$\sum_u f(u, v) = \begin{cases} r(v) & \text{if} \quad v \in V_0 \cup V_s, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Let $\{v_{i,j}, v_{i',j+1}\}$ be an edge with $j + 1 \leq k - 1$. We define

$$f(v_{i,j}, v_{i',j+1}) = \tfrac{1}{2} \operatorname{Mean}_{v_{i'',0}} r(v_{i'',0}),$$

where the arithmetic mean is taken over all vertices $v_{i'',0}$ such that there exists a straight path in $G_{k,2k-2}$ from $v_{i'',0}$ to $v_{i,j}$. $f(v_{i,j}, v_{i',j+1})$ is thus equal to half the mean of all $r(v_{i'',0})$, the incoming flow in $v_{i,j}$ from previous layers which is equally spread into its

two outgoing edges. Hence, the incoming flow in all $v_{i,k-1}$ will be a constant. Defining $f(v_{i',j+1}, v_{i,j}) = -f(v_{i,j}, v_{i',j+1})$, it is easy to show that the above three conditions are satisfied for all edges before the layer $k-1$.

Similarly, for $j \geq k-1$, we define $f(v_{i,j}, v_{i',j+1})$ to be half the incoming flow in $v_{i',j+1}$ from the next layers starting at V_s. The conditions are satisfied for all edges after the layer $k-1$. The flow coming from all the upper layers to the layer $k-1$ is also equally spread into all the vertices. Then, due to the condition that the sum of all $r(v)$ is 0, the third condition is also satisfied in the layer $k-1$. □

Using the previous lemma we show:

Lemma 16. *For any set U in $G_{k,2k-2}$, if $c = \#(U \cap (V_0 \cup V_s))$ where V_0 is the first layer and V_s is the last one, we have $G^{d/2}_{k,2k-2}(U) \geq 2^{k-1} - |2^{k-1} - c|$.*

Proof. We note that $G^{d/2}_{k,2k-2}(U) = G^{d/2}_{k,2k-2}(V-U)$ (this comes from the fact that V is in the kernel of the quadratic form). Thus, possibly replacing U by $V-U$, we can assume $c \leq 2^{k-1}$. Now, for $v \in V_0 \cup V_s$, we can define $r(v)$ to be 2 if $v \in U$ and $r(v) = -2c/(2^k - c)$ otherwise. Since r satisfies the conditions of the lemma, there exists a flow with source r, hence with capacity $2c$. We note that $2G^{d/2}_{k,2k-2}(U)$ is equal to the cardinality of the border of A, which is a cut for the flow. Hence, the min-cut max-flow theorem says $2G^{d/2}_{k,2k-2}(U) \geq 2c$. □

Corollary 17. $C(G^{d/2}_{k,2k-2}) \geq \frac{2}{3}\,2^{k-1}$.

Proof. In a similar way as in the proof of Theorem 7, we get

$$C(G^{d/2}_{k,2k-2}) \geq \max_{0 \leq x \leq 2^k} \min_{x \leq y \leq 2x} (2^{k-1} - |2^{k-1} - x|),$$

which is equal to $\frac{2}{3}2^{k-1}$. □

A similar method applied on $G_{k,k-1}$ (basically, in taking only V_0 into account in the source of the flow) enables us to prove:

Corollary 18. $C(G^{d/2}_{k,k-1}) \geq \frac{1}{3}\,2^{k-1}$.

This confirms the partial result obtained by the symmetric approach.

These results establish a constant ratio between the upper bounds and the lower bounds. Actually, the same method applied to linear algorithms shows that $C_{\text{lin}}(G_{k,s}) = 2^{k-1}$, which proves the conjecture in [15]. We conjecture that the upper bounds are the real complexities also in the general case for $s = 2k-2$. This means that $s = 2k-2$ has to be chosen to get the optimal security for the $G_{k,s}$ compression function family.

3. Possible Extensions and Conclusion

The analysis on cryptographic primitives proposed here can be extended in a more general context. To allow several edge domains to exist together in the same primitive, we can add the notion of edge degree of freedom in the definition of the computation graphs. This would be the logarithm (in any basis) of the cardinality of the domain. We mention that all the results still hold if we replace $d(v)$ by the sum of all the adjacent edge degrees. We can also allow the value of an edge to be involved in more than two different vertices replacing the notion of graph by the notion of hypergraph.

We have proposed a new framework for the study of the security of cryptographic primitives defined as a computation graph. We showed that the complexity of resolving a computation graph is related to the local expansion properties of the graph. This theory enables one to prove the one-wayness of a family of compression functions with respect to the black-box attacks. It can be applied to the compression functions based on the FFT network. It turns out that the function is the most secure possible (in context with the black-box cryptanalysis) for a doubled FFT network, that is for $s = 2k - 2$.

Acknowledgment

We thank László Babai and Jacques Stern for helpful discussions. We also thank the anonymous referees for extensive and useful remarks.

References

[1] N. Alon. Eigenvalues and Expanders. *Combinatorica*, vol. 6, pp. 83–96, 1986.

[2] N. Alon, V. D. Milman. λ_1, Isoperimetric Inequalities for Graphs, and Superconcentrators. *Journal of Combinatorical Theory, Series B*, vol. 38, pp. 73–88, 1985.

[3] L. Babai, M. Szegedy. Local Expansion of Symmetrical Graphs. *Combinatorics, Probability and Computing*, vol. 1, pp. 1–11, 1992.

[4] T. Baritaud, H. Gilbert, M. Girault. FFT Hashing Is Not Collision-Free. In *Advances in Cryptology—EUROCRYPT* '92, Balatonfüred, Lecture Notes in Computer Science 658, pp. 34–44, Springer-Verlag, Berlin, 1993.

[5] M. Bellare, P. Rogaway. Random Oracles Are Practical: a Paradigma for Designing Efficient Protocols. In *Proceedings of the* 1*st ACM Conference on Computer Communication Security*, pp. 62–73, 1993.

[6] E. Biham, A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York, 1993.

[7] H. Dobbertin. Cryptanalysis of MD4. In *Fast Software Encryption—Proceedings of the Cambridge Security Workshop*, Cambridge, Lecture Notes in Computer Science 1039, pp. 53–69, Springer-Verlag, Berlin, 1996.

[8] J. L. Massey. SAFER K-64: a Byte-Oriented Block-Ciphering Algorithm. In *Fast Software Encryption—Proceedings of the Cambridge Security Workshop*, Cambridge, December 1993, Lecture Notes in Computer Science 809, pp. 1–17, Springer-Verlag, Berlin, 1994.

[9] M. Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology—CRYPTO* '94, Santa Barbara, California, Lecture Notes in Computer Science 839, pp. 1–11, Springer-Verlag, Berlin, 1994.

[10] V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, vol. 55, no. 2, pp. 165–172, 1994.

[11] R. L. Rivest. The MD4 Message-Digest Algorithm. In *Advances in Cryptology—CRYPTO* '90, Santa Barbara, California, Lecture Notes in Computer Science 537, pp. 303–311, Springer-Verlag, Berlin, 1991.

[12] C. P. Schnorr. FFT-Hashing: an Efficient Cryptographic Hash Function. Presented at the CRYPTO '91 Conference. Unpublished.
[13] C. P. Schnorr. FFT-Hash II, Efficient Cryptographic Hashing. In *Advances in Cryptology—EUROCRYPT '92*, Balatonfüred, Lecture Notes in Computer Science 658, pp. 45–54, Springer-Verlag, Berlin, 1993.
[14] C. P. Schnorr, S. Vaudenay. Parallel FFT-Hashing. In *Fast Software Encryption—Proceedings of the Cambridge Security Workshop*, Cambridge, December 1993, Lecture Notes in Computer Science 809, pp. 149–156, Springer-Verlag, Berlin, 1994.
[15] C. P. Schnorr, S. Vaudenay. Black Box Cryptanalysis of Hash Networks Based on Multipermutations. In *Advances in Cryptology—EUROCRYPT '94*, Perugia, Lecture Notes in Computer Science 950, pp. 47–57, Springer-Verlag, Berlin, 1995.
[16] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology—EUROCRYPT '97*, Konstanz, Lecture Notes in Computer Science 1233, pp. 256–266, Springer-Verlag, Berlin, 1997.
[17] R. M. Tanner. Explicit Concentrators from Generalized N-Gons. *SIAM Journal of Algebraic Discrete Methods*, vol. 5, pp. 287–293, 1984.
[18] S. Vaudenay. FFT-Hash II Is Not Yet Collision-Free. In *Advances in Cryptology—CRYPTO '92*, Santa Barbara, California, Lecture Notes in Computer Science 740, pp. 587–593, Springer-Verlag, Berlin, 1993.
[19] S. Vaudenay. On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In *Fast Software Encryption*, Leuven, Lecture Notes in Computer Science 1008, pp. 286–297, Springer-Verlag, Berlin, 1995.
[20] S. Vaudenay. La Sécurité des Primitives Cryptographiques, Thèse de Doctorat de l'Université de Paris 7, Technical Report LIENS-95-10 of the Laboratoire d'Informatique de l'Ecole Normale Supérieure, 1995.