An extended abstract appears in *Advances in Cryptology – Eurocrypt 98 Proceedings*, Lecture Notes in Computer Science Vol. 1403, K. Nyberg ed., Springer-Verlag, 1998. This is the full version.

Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-Invertible

Mihir Bellare[*] Ted Krovetz[†] Phillip Rogaway[†]

October 17, 1998

Abstract

We argue that the invertibility of a block cipher can reduce the security of schemes that use it, and a better starting point for scheme design is the non-invertible analog of a block cipher, that is, a pseudorandom function (PRF). Since a block cipher may be viewed as a pseudorandom permutation, we are led to investigate the *reverse* of the problem studied by Luby and Rackoff, and ask: "how can one transform a PRP into a PRF in as security-preserving a way as possible?" The solution we propose is *data-dependent re-keying.* As an illustrative special case, let $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be the block cipher. Then we can construct the PRF F from the PRP E by setting $F(k,x) = E(E(k,x),x)$. We generalize this to allow for arbitrary block and key lengths, and to improve efficiency. We prove strong quantitative bounds on the value of data-dependent re-keying in the Shannon model of an ideal cipher, and take some initial steps towards an analysis in the standard model.

Keywords: Birthday attacks, block ciphers, pseudorandom functions, symmetric encryption.

[*] Dept. of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: `mihir@cs.ucsd.edu`. Web page: `http://www-cse.ucsd.edu/users/mihir`. Supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

[†] Dept. of Computer Science, Engineering II Bldg., University of California at Davis, Davis, CA 95616, USA. E-mail: `{krovetz,rogaway}@cs.ucdavis.edu`. Web page: `http://www.cs.ucdavis.edu/~{krovetz,rogaway}`. Supported in part by NSF CAREER Award CCR-9624560 and a MICRO grant from RSA Data Security, Inc.

Contents

1 Introduction 3

2 The Problem 3

2.1 Invertibility can hurt when using block ciphers: An example . . . 3

2.2 PRPs, PRFs, and their relation to block ciphers . . . 4

2.3 Luby-Rackoff backwards . . . 5

2.4 History and related work . . . 6

3 The Fn Construction 6

4 Definitions 8

4.1 Complexity theoretic model . . . 8

4.2 Ideal block cipher model . . . 9

5 Security of the Fn Construction 10

5.1 Security in the complexity theoretic model . . . 10

5.2 Security in the ideal block cipher model . . . 10

5.3 Attacks / Lower bounds . . . 12

6 Proof of Theorem 5.1 13

7 Proof of Theorem 5.2 17

7.1 Lemmas . . . 18

7.2 Proof of Theorem 5.2, Part 1 . . . 19

7.3 Proof of Theorem 5.2, Part 2 . . . 21

8 Analysis of attacks 24

8.1 Proof of Proposition 5.3 . . . 24

8.2 Proof of Proposition 5.4 . . . 25

References 25

1 Introduction

This paper describes a transformation — turning a "pseudorandom permutation" (PRP) into a "pseudorandom function" (PRF) using "data-dependent re-keying." It can be applied to a block cipher to increase the block cipher's security in certain ways, and, in particular, the method leads to block cipher based message encryption and authentication techniques which are approximately as efficient as ones in current use, but have better security.

In Section 2 we explain our (at first paradoxical sounding) thesis: that invertibility of a block cipher can be a liability, not an asset, when it comes to the security of schemes that use the cipher. We will then explain what are PRFs and PRPs, how the former are a better starting point for constructions but the latter a better model for block ciphers, and how all this leads us to consider the problem of transforming PRPs into PRFs in a security-preserving way.

In Section 3 we describe our way to do the PRP to PRF transformation. We call our transform Fn^d, where d is a parameter on which the construction depends. (The impatient reader can jump to Section 3 to see how Fn^d works. It is very simple.)

Our main result is an analysis[1] in the Shannon model which shows that if the block cipher is ideal then its transform under Fn^d is close to an ideal random function. The provided bounds are strong, showing the transform is close to security preserving.

The interpretation of the above is that the Fn^d transform gives good security against "generic" attacks. To guage its strength against cryptanalytic attacks we also analyze it in the standard complexity theoretic or "reductionist" framework. We do succeed in providing a reduction, but the quality of the bounds is not as good as in the Shannon model, and thus we view these results as preliminary, hopefully to be improved.

The results are presented, discussed, and displayed graphically in Section 5. Just before that, in Section 4, we provide the precise definitions of the security notions, but these can be skipped at first reading, or skipped entirely by an expert. The rest of the paper is devoted to proofs.

2 The Problem

We begin with a simple example, then relate these issues to PRFs and PRPs, then describe the problem that results, and conclude with a discussion of related work.

2.1 Invertibility can hurt when using block ciphers: An example

A block cipher is a function E: $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ which transforms an n-bit message block x into an n-bit string y under the control of a κ-bit key k: $y = E(k,x)$. The function is invertible in the sense that for each key the map $E_k \stackrel{\text{def}}{=} E(k,\cdot)$ is a permutation of $\{0,1\}^n$, and knowledge of k permits computation of E_k^{-1}. Concrete examples are DES, triple-DES and RC5.

Message encryption is done by using the block cipher in some mode of operation, such as "CBC." Using even a very "good" block cipher (say triple-DES, or even an ideal cipher), CBC encryption becomes insecure once $2^{n/2}$ blocks have been encrypted, in the sense that at this point partial information about the message begins to leak[2], due to birthday attacks.[3] Furthermore, this is true

[1] All analyses in this paper are concrete and quantitative, meaning providing explicit, non-asymptotic bounds on the success probabiilty of an adversary as a function of its resources.

[2] A good encryption scheme is much more than one that prevents key recovery from a ciphertext: it should have the property that even partial information about the plaintext is not revealed [9, 4].

[3] The attacks are well known. See [4] for an analysis of their effectiveness relative to formal notions of security

for many other common modes of operation, too. Thus direct use of a 64-bit block size block cipher usually enables one to safely encrypt no more than 2^{32} blocks, which is quite small.

We stress that these attacks arise because the cipher is a permutation, and their cost depends only on the block length, not the key length or the security of the block cipher. So the attacks are just as effective for triple-DES, or even an ideal block cipher, as they are for DES. In summary, block cipher based schemes are often subject to birthday attacks arising from the very nature of block ciphers as permutations.

So how can we safely encrypt more than $2^{n/2}$ blocks? One answer is to use a slightly different type of primitive in an appropriate mode of operation: specifically, a "pseudorandom function" (PRF) in CTR (counter) mode, as discussed in [4, 11] and explained further below. This way to encrypt is easy and has no extra overhead if a PRF of cost comparable to the block cipher is available.

The above is only one example of an issue that arises in many places: that the permutivity of a block cipher can hinder the security of schemes which use it. To effectively address this we need to explain what are PRFs and PRPs and how they relate to block ciphers.

2.2 PRPs, PRFs, and their relation to block ciphers

Let us first back up and look at how the security of a block cipher is best captured.

Security of a block cipher: PRPs. It is natural to view a real block cipher as constructed to "approximate", as closely as possible, an ideal block cipher (that is, a random permutation) in the sense that if you don't know the key k and only see input/output examples of E_k then these should appear like input/output examples of a random permutation. The quality of a given block cipher E as a PRP (pseudorandom permutation) is thus captured by a function $\textsc{Sec}_E^{\mathrm{prp}}(q,t)$ which returns the maximum "advantage" that one can obtain in distinguishing E_k from a random permutation if you see q input/output examples and are allowed further computational resources bounded by t. (In the complexity-theoretic model, t will bound computing time; in the information-theoretic model, t will bound the number of known $(k, x, E_k(x))$ values. The advantage is a number between 0 and 1 given as the difference of two probabilities: the probability that the adversary outputs 1 given a random function E_k from E, and the probability that the adversary outputs 1 given a random permutation π. See Section 4 for more details.)

Each specific cipher (eg. DES) will have such an associated security function, which depends on (and to a large extent comprises) its cryptanalytic strength. Of course we won't know for sure what is this function, but we can work with what we know from cryptanalytic results. For example, if the linear cryptanalysis of [13] is the best attack on DES, we might assume $\textsc{Sec}_{\mathrm{DES}}^{\mathrm{prp}}(q,t)$ stays small (close to 0) until q, t reaches around 2^{43}. From now on, "block cipher" and "PRP" are synonymous, from the security point of view.

Ciphers without invertibility: PRFs. Like a block cipher, a pseudorandom function (PRF) is a map $F\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, but now $F_k \stackrel{\mathrm{def}}{=} F(k,\cdot)$ is not required to be invertible. The required security property is to approximate, as closely as possible, a random function. The quality of a given function F is captured by $\textsc{Sec}_F^{\mathrm{prf}}(q,t)$ which returns the maximum "advantage" that one can obtain in distinguishing F_k from a random function if you see q input-output examples and are allowed computational resources t. (This advantage is the difference between probability that the adversary outputs 1 given a random function F_k from F and the probability that the adversary outputs 1 given a random function ρ. See Section 4 for more details.)

for encryption.

THE EXAMPLE REVISITED. Counter mode encryption with a PRF F means that to encrypt an m-block plaintext $M = x_1 \cdots x_m$, send

$$(\,\mathrm{ctr},\ F_k(\langle \mathrm{ctr}+1\rangle)\oplus x_1 \parallel \cdots \parallel F_k(\langle \mathrm{ctr}+m\rangle)\oplus x_m\,)$$

where $\langle i\rangle$ is the binary encoding of i into n bits, " $\parallel$ " denotes concatenation, and where you increment ctr by m after doing each encryption. (Notice that to decrypt you need only apply F_k, so that you don't need this function to be invertible.) Counter-mode encryption with a good PRF is pretty much "ideal encryption": it is shown in [4] that an adversary's chance of obtaining partial information about some plaintext, after q blocks have been encrypted, is at most $\mathrm{SEC}_F^{\mathrm{prf}}(q,t)$, the strength of F as a PRF. In particular if we had a PRF F with the same numerical security as DES but as a PRF not a PRP, namely $\mathrm{SEC}_F^{\mathrm{prf}}(q,t) \approx \mathrm{SEC}_{\mathrm{DES}}^{\mathrm{prp}}(q,t)$, then we could encrypt nearly 2^{43} blocks, well above the birthday bound.

In contrast, when we use a block cipher (PRP) directly in CBC (or CTR) mode, we are not able to recoup all of the cryptographic strength captured by its $\mathrm{SEC}_E^{\mathrm{prp}}(\cdot,\cdot)$ value, because at $q = 2^{n/2}$ (which is $q = 2^{32}$ for DES) birthday attacks kill the encryption scheme.

The conclusion can be put like this: to get quantitatively good security, what is most useful and convenient about F is that $\mathrm{SEC}_F^{\mathrm{prf}}(q,t)$ be small, not $\mathrm{SEC}_F^{\mathrm{prp}}(q,t)$. To make the former as low as possible the family F must *not* be a family of permutations, since no family of permutation will have a good value of $\mathrm{SEC}_F^{\mathrm{prf}}(q,t)$ if $q \geq 2^{n/2}$. This is because of birthday attacks: if F is a family of permutations then the adversary $A(q)$ who guesses "random function" if and only if she sees a collision in the answers returned from q distinct but otherwise arbitrary queries already accrues advantage of about $1/e$ if $q = 2^{n/2}$. The adversary's advantage then goes quickly to 1 with $q \gg 2^{n/2}$.

2.3 Luby-Rackoff backwards

The above is part of an emerging view or understanding, emanating from works like [4, 5, 6, 20], that when it comes to designing higher-level primitives (like encryption schemes or MACs) a PRF is a better tool than a PRP, from two points of view: it permits easier and more effective *analysis* of the designed scheme, and the resulting schemes have a greater proven *quantitative security.* This leads us to suggest that for the purpose of protocol design, what we really want are PRFs, not block ciphers (PRPs).

So the question is how to get PRF families of high security and low cost. One possibility is to make these directly, in the same way we make block ciphers now. We suggest that this indeed be kept in mind for the future, but at the moment is not a very pragmatic view, for two reasons. First, we have lots of (good) block ciphers available, and we want to use them well. Second, permutivity may be important to the design process of block ciphers; for example, using the round structure of a Feistel-network gives rise to a permutation.[4]

We propose instead to transform PRPs into PRFs. That is, starting with a good PRP E (realized by a block cipher), convert it into a good PRF F. This is effectively the reverse of the problem considered by Luby and Rackoff [12], who wanted to turn PRFs into PRPs.

A crucial issue is to make transformations that are as "security preserving" as possible. We want $\mathrm{SEC}_F^{\mathrm{prf}}(q,t)$ to remain low even for $q \gg 2^{n/2}$. Ideally, $\mathrm{SEC}_F^{\mathrm{prf}}(q,t)$ would be close to $\mathrm{SEC}_E^{\mathrm{prp}}(q,t)$.

Let us now discuss some related work. Following that we present our construction.

[4] Another possibility is to make sure that the block size n is large enough ($n \geq 128$) that attacks of complexity $2^{n/2}$ are irrelevant. This too is a good idea, but the construction we give has merit which goes beyond the birthday attacks which we have been using to motivate this problem.

2.4 History and related work

Our construction is related to the cascade construction of [3].

The notion of a PRF was first defined in the polynomial-time framework by Goldreich, Goldwasser and Micali [8]. A concrete security treatment of PRFs, together with the idea that concretely defined PRFs/PRPs can be used to model block ciphers, originates with [6]. Luby and Rackoff use the term PRP to refer to a family of permutations that is a PRF family in the sense of [8]. Our notion is different in that we measure the advantage relative to random permutations, not functions. This makes no difference in the polynomial-time framework, but in the concrete-security framework the difference is crucial; indeed, if concrete security is ignored, the problem we are considering does not exist.

The ideal block cipher model we use for some of our results is that of [19], used also in [7, 10].

There are many natural ways to try to do the PRP-to-PRF conversion. One of the first to come to mind is to define $F_k(x) = x \oplus E_k(x)$. This construction is of value in some contexts, but not in ours. For if you are given an oracle for this $F_k(\cdot)$ you effectively have an oracle for $E_k(\cdot)$: for any query x you can compute $E_k(x)$ as $x \oplus F_k(x)$. So F_k will resemble a random function no more than E_k does.

There are many natural alternatives to the Fn^d transformation. For example, truncate $E_k(x)$, defining $F_k(x)$ to be some appropriate-length prefix of $E_k(x)$. This scheme was partially analyzed by [2]. Another natural method is $F_{k_1 k_2}(x) = E_{k_1}(x) \oplus E_{k_2}(x)$. This has not been analyzed.

Aiello and Venkatesan [1] give a general construction for turning a PRF $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ into a PRF $F : \{0,1\}^{6\kappa} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$. But this is a different problem. Although they too want to circumvent some birthday attacks, their starting point is a random function (not a permutation) and the problem is to double the number of bits the function can take as input. They are bound by the original security of the starting function as a PRF: birthday attacks are only prevented in the sense that the construction does not induce such attacks itself. So if a block cipher is the starting point, it is viewed as a PRF, meaning the security is only $2^{n/2}$. There is no notion of modeling a cipher as a random permutation. In contrast, we go above the original birthday threshold, to a security close to 2^n. Our construction is also more efficient, and it yields a map of the same key size and block length as the original one.

In constructing a Wegman-Carter message authentication code (MAC) [21] one needs to symmetrically encrypt the universal-hash of each message M. If a PRP is in hand for doing the encryption, one could define $\mathsf{MAC}_{k1,k2}(M) = (\text{ctr},\ E_{k2}(\text{ctr}) \oplus h_{k1}(M))$, but the security would degrade by $\Theta(q^2 2^{-n})$ compared to using a PRF. (Here q is the number of MACed messages.) Shoup [20] describes an alternative with better exact security. Our methods allow the simpler and more general $(\text{ctr},\ F_{k2}(\text{ctr}) \oplus h_{k1}(M))$, where F is the result of PRP-to-PRF conversion starting from E.

As we explained, Luby and Rackoff consider the complementary problem of turning a PRF into a block cipher [12]. Luby and Rackoff spawned much further work, including [14, 15, 16, 17, 22], and our work shares their emphasis on concrete bounds, efficiency, and tight reductions.

3 The Fn Construction

We have described in Section 2.4 some simple suggestions that don't work and some related constructions. Now we present our solution. We let E: $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be the given block cipher (PRP).

The values n and κ vary across real block ciphers; for example, for DES we have $\kappa = 56$ and $n = 64$; for (two-key) triple DES we have $\kappa = 112$ and $n = 64$. We want to handle all these cases.

Accordingly, our construction depends on the relative values of κ and n. It also depends on a parameter d, where $0 \le d < n$.

Simple Case. The simplest case of our construction is when the given PRP has the property that $\kappa = n$, and we choose $d = 0$. One then defines $F = \mathsf{Fn}^0 E$ by $F(k, x) = E(E(k, x), x)$. That is, $F_k(x) = E_{k'}(x)$, where $k' = E_k(x)$. We call this "data-dependent re-keying" since we are applying E to x, but using the data-dependent "derived key" $k' = E_k(x)$. The cost of computing F is twice the cost of computing E, in the sense that there are two applications of E for each application of F. The general construction includes a provision aimed at reducing this overhead.

The General Case. Let $0 \le d < n$ be given. If x' is an n-bit string, let $x' \gg d$ denote x' shifted to the right by d positions, with 0-bits filling the vacated positions. If k' is a string of length ℓ, let $[k']_{1..i}$ be the string consisting of the first i bits of k' (for $1 \le i \le \ell$). Set $j = \lceil \kappa/n \rceil$. The function $F = \mathsf{Fn}^d E$ takes a κj-bit key $k_1 \cdots k_j$ and an n-bit input x to return an n-bit output y as follows:

```
function $F(k_1 \cdots k_j,\ x)$
begin
    $x' \leftarrow x \gg d$                                      //Shift away low $d$ bits
    $k' \leftarrow E(k_1, x') \| \cdots \| E(k_j, x')$    //Construct the "extended" derived key
    $k \leftarrow [k']_{1..\kappa}$                                 //We only need $\kappa$ bits of derived key
    $y \leftarrow E(k, x)$                                     //Use derived key on the input
    return $y$
end
```

We call x' the *group selector* and k the *derived key*. The j applications of E_{k_i} are to deal with the possibility that $\kappa > n$, and the truncating of k' to κ bits is to handle the possibility that the key length κ might not be a multiple of the block length n. (More strange is the discarding of bits from the x, namely the $x \gg d$. This is for efficiency, as we will explain below.) As an example, if $E = \text{DES}$, so that $\kappa = 56$ and $n = 64$, we would have $j = 1$, so the key of F is just a 56-bit DES key k_1, the derived key k' is the first 56 bits of $\text{DES}_{k_1}(x')$, and the output is $\text{DES}_{k'}(x)$. If E is TDES (two-key triple-DES), so that $\kappa = 112$ and $n = 64$, we would have $j = 2$, so the key for F is a pair $k_1 k_2$ of TDES keys, the derived key k' is the first 112 bits of $\text{TDES}_{k_1}(x')\text{TDES}_{k_2}(x')$, and the output is $\text{TDES}_{k'}(x)$.

Notice that for fixed $k_1 \cdots k_j$, if two n-bit strings determine the same group selector then they generate the same derived key, and this happens if the two strings agree in the first $n - d$ bits. Accordingly, we cluster together all points that have the same group selector into what we call a *common key group*. Thus there are a total of 2^{n-d} common key groups. For any $\alpha \in \{0,1\}^{n-d}$ we define $\text{ckg}_\alpha = \{x : [x]_{1...n-d} = \alpha\}$ as the α-th common key group. Identifying strings with integers in the natural way, the i-th common key group consists of the integers $(i-1)2^d, ..., i2^d - 1$.

Efficiency. Recall that the nominal way to encrypt using $F = \mathsf{Fn}^d E$ involves applying F to a single key k and successive ctr-values. By dropping the least significant d bits of this counter, one needs to recompute k' only once every 2^d invocations of F. Of course an implementation would need to to record the last derived key and refrain from re-computing it. Doing this makes the amortized cost to compute F just $(1 + j2^{-d})$ times the cost of computing E. For many ciphers this is an underestimate because of additional cost associated to changing the key. In fact, the cost of changing the key for some block ciphers is high, which is why we don't want to do it very often.

Variations. How exactly one drops bits of x is not so important. For example, instead of shifting to the right one could zero-out the least significant d bits. This makes no difference in the analysis.

We have constructed $F = \mathsf{Fn}^d E$ to be a map $F : \{0,1\}^{j\kappa} \times \{0,1\}^n \to \{0,1\}^n$. If one prefers, let $\overline{F}_k(x) = E_{k'}(x)$ where k' is the first κ bits of $E_{k_1}(x \gg d) \parallel \cdots \parallel E_{k_j}(x \gg d)$ and k_i is defined as $E_k(\langle i \rangle)$. Now $\overline{F}$ uses a κ-bit key, just like F. The analysis of F lifts to $\overline{F}$ with just a tiny loss in quantitative security.

4 Definitions

Here we give the more precise definitions of security in the two models in which we will be analyzing our construction, namely the (standard) "complexity theoretic" model and the Shannon model.

Recall that in Section 2 we discussed the security of F and E by way of functions $\textsc{Sec}^{\mathrm{prf}}_F(q,t)$ and $\textsc{Sec}^{\mathrm{prp}}_E(q,t)$. Their meaning changes according to the model in a simple way:

- In the complexity theoretic model they are $\mathrm{C}\textsc{Sec}^{\mathrm{prf}}_F(q,t)$ and $\mathrm{C}\textsc{Sec}^{\mathrm{prp}}_E(q,t)$, respectively, these quantities being defined in Section 4.1 below, and
- In the ideal cipher model, they are $\mathrm{I}\textsc{Sec}^{\mathrm{prf}}_{\mathsf{F}}(q,t)$ and $\mathrm{I}\textsc{Sec}^{\mathrm{prp}}_{\kappa,n}(q,t)$, respectively, these quantities being defined in Section 4.2 below, where F refers to the transformation that takes E into F. (In our case, $\mathsf{F} = \mathsf{Fn}^d$).

$\textsc{Preliminaries}$. If S is a probability space then $g \leftarrow S$ denotes the operation of selecting g at random according to the distribution specified by S. If S is a set it is viewed as endowed with the uniform distribution, so that $g \leftarrow S$ means that g is selected uniformly at random from set S. If y is not a set then $g \leftarrow y$ is a simple assignment statement, assigning g the value y. (It is thus equivalent to $g \leftarrow \{y\}$.) Let Perm_n denote the set of all permutations $\pi : \{0,1\}^n \to \{0,1\}^n$. Let Rand_n denote the set of all functions $\rho : \{0,1\}^n \to \{0,1\}^n$. Let $\mathrm{BC}_{\kappa,n}$ be the set of all maps $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ such that $E(k,\cdot) \in \mathrm{Perm}_n$ for all $k \in \{0,1\}^\kappa$. Let $\mathrm{RF}_{\kappa,n}$ be the set of all maps $R : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$.

A family of functions with key length κ and block length n is a map $G : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, that is, $G \in \mathrm{RF}_{\kappa,n}$. Each κ-bit key k specifies the map $G_k \stackrel{\mathrm{def}}{=} G(k,\cdot) \in \mathrm{Rand}_n$. This map is not necessarily a permutation. If G_k is a permutation for each $k \in \{0,1\}^\kappa$ (ie., $G \in \mathrm{BC}_{\kappa,n}$) then we call G a family of permutations, or a block cipher. We view G as a probability space over Rand_n given by choosing functions via a uniform choice of the underlying key; that is, $g \leftarrow G$ is the same as $k \leftarrow \{0,1\}^\kappa$; $g \leftarrow G_k$.

Given a block cipher E, the block cipher $E^{-1} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is defined by $E^{-1}(k,y)$ being the unique point x such that $E(k,x) = y$. We interchangeably write $E_k^{-1}(y)$ and $E^{-1}(k,y)$.

An adversary is an algorithm A with access to some number of oracles. Oracles are denoted as superscripts to A, as in $A^{E,E^{-1},F}$. An oracle responds to its query in unit time.

4.1 Complexity theoretic model

We will have two measures of security: the strength of G as a PRF and the strength of G as a PRP. We follow [6] in the manner in which the basic notion of [8] is "concretized."

First, we need the concept of advantage, which for emphasis we call the "computational advantage" and write $\mathrm{C}\textsc{Adv}$. Let D be an algorithm (a "distinguisher") taking an oracle for a function g, and let G_1, G_2 be two families of functions with the same block length. We define

$$\mathrm{C}\textsc{Adv}_{G_1,G_2}(D) \;=\; \Pr\left[\, g \leftarrow G_1 \,:\, D^g = 1 \,\right] - \Pr\left[\, g \leftarrow G_2 \,:\, D^g = 1 \,\right] .$$

Now, suppose F is a family of functions, and E is a family of permutations. We let

$$\begin{array}{rcl|rcl} \textsc{CAdv}_F^{\mathrm{prf}}(D) & = & \textsc{CAdv}_{F,\mathrm{Rand}_n}(D) & \textsc{CAdv}_E^{\mathrm{prp}}(D) & = & \textsc{CAdv}_{E,\mathrm{Perm}_n}(D) \\ \textsc{CSec}_F^{\mathrm{prf}}(q,t) & = & \max_D \{\textsc{CAdv}_F^{\mathrm{prf}}(D)\} & \textsc{CSec}_E^{\mathrm{prp}}(q,t) & = & \max_D \{\textsc{CAdv}_E^{\mathrm{prp}}(D)\} \end{array}$$

Here the first quantity measures the advantage D has in distinguishing random members of F (resp. E) from truly random functions (resp. permutations) of the same block length. The second quantity is the maximum advantage attainable using some amount of resources, in this case the number q of oracle queries and the running time t. For simplicity, when we speak of an adversary's time we mean the adversary's actual running time plus the size of the encoding of the adversary (relative to some fixed encoding scheme), so we have a single parameter t to capture time plus description size. The maximum here is over all distinguishers D that make up to q oracle queries and have running time bounded by t.

4.2 Ideal block cipher model

The Shannon model [19] treats E as a random block cipher. This means that each E_k is taken to be random permutation on n-bit strings. Let $\mathsf{F}E$ be some operator on E which returns a new family of functions, and say the new family has key length κ^* but the block length is still n. (For us, $\mathsf{F} = \mathsf{Fn}^d$ and $\kappa^* = j\kappa$ where $j = \lceil \kappa/n \rceil$.) As modeled by [7], the adversary that attacks F is given oracles for $E(\cdot,\cdot)$ and $E^{-1}(\cdot,\cdot)$ — as well as an oracle f where either $f(\cdot) = F(k^*,\cdot)$ for $F = \mathsf{F}E$ and k^* a randomly chosen key in $\{0,1\}^{\kappa^*}$, or else $f(\cdot) = \rho(\cdot)$, for a random function $\rho : \{0,1\}^n \to \{0,1\}^n$. We investigate the adversary's advantage in determining what type of oracle f is. This is defined as:

$$\begin{aligned} \textsc{IAdv}_{\mathsf{F}}^{\mathrm{prf}}(A) \;\; = \;\; & \Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}\,;\, k^* \leftarrow \{0,1\}^{\kappa^*}\,;\, f \leftarrow (\mathsf{F}E)_{k^*} \;:\; A^{E,E^{-1},f} = 1 \right] \\ & - \Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}\,;\, f \leftarrow \mathrm{Rand}_n \;:\; A^{E,E^{-1},f} = 1 \right] . \end{aligned}$$

The advantage A gains depends, in part, on the number of queries q she asks of f and the total number of queries t she asks of E and E^{-1}. We are interested in

$$\textsc{ISec}_{\mathsf{F}}^{\mathrm{prf}}(q,t) \;=\; \max_A \{\textsc{IAdv}_{\mathsf{F}}^{\mathrm{prf}}(A)\} ,$$

the maximum being over all adversaries that make up to q queries to the f oracle and up to t queries to the E and E^{-1} oracles.

This is an information-theoretic setting: the adversary has unlimited computational power. If we think of E as a concrete block cipher, and not an idealized one, then attacks in this model correspond to attacks in which the adversary exploits no characteristics specific to the block cipher, only "generic" features of the construction F we are analyzing. Thus, security guarantees from results in this model are weaker than those from results in the model above, yet they do have some meaning. We use the Shannon model when technical difficulties prevent us from getting bounds as good as we would like in the complexity theoretic model.

Note. The goal will be to upper bound $\textsc{ISec}_{\mathsf{F}}^{\mathrm{prf}}(q,t)$ as a function of t, q, κ, n. As such we don't really need any notion of $\textsc{ISec}_{\kappa,n}^{\mathrm{prp}}(q,t)$, the security of the block cipher, because the latter is assumed ideal, but there are two reasons to define it anyway. First, to maintain a uniform security treatment across the models, and in particular be consistent with Section 2; second, because it is indeed the quantity with which we wish to compare $\textsc{ISec}_{\mathsf{F}}^{\mathrm{prf}}(q,t)$.

We define $\textsc{ISec}_{\kappa,n}^{\mathrm{prp}}(q,t)$ as the maximum, over all adversaries A of the specified resources, of the quantity

$$\Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}\,;\, k \leftarrow \{0,1\}^{\kappa}\,;\, f \leftarrow E_k \;:\; A^{E,E^{-1},f} = 1 \right]$$

$$- \Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n} \,;\, f \leftarrow \mathrm{Perm}_n \;:\; A^{E,E^{-1},f} = 1 \right] .$$

Notice that this quantity is not zero. For $q > 1$ and large n we would expect it to be about $t \cdot 2^{-\kappa}$, corresponding to an exhaustive key search attack.

5 Security of the Fn Construction

We summarize both proven security guarantees and attacks that indicate the tightness of the bounds in them.

5.1 Security in the complexity theoretic model

Here we refer to the notions of security of Section 4.1. We assume E is a PRP family and show our construction is a PRF family, via a reduction. We do this only for the case where the key length, κ, is identical to the block length, n, and we drop no bits, namely $d = 0$.

Theorem 5.1 *Let $\kappa = n$ be a positive integer and let E: $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a family of permutations whose security as a PRP family is described by security function $\mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(\cdot,\cdot)$. Let F: $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be our construction for the case of no bit dropping, namely $F = \mathsf{Fn}^0 E$. Its security as a PRF is described by function $\mathrm{CS{\scriptstyle EC}}_F^{\mathrm{prf}}(\cdot,\cdot)$ which for any number of queries $q \le 2^n/2$ and time t can be bounded as follows:*

$$\mathrm{CS{\scriptstyle EC}}_F^{\mathrm{prf}}(q,t) \;\le\; \mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(q,t') + q \cdot \mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(3,t') + \frac{q^2}{2^{2n}}$$

where $t' = t + O(q) \cdot (\kappa + n + \mathrm{Time}_E)$.

Proof: See Section 6. ▌

The bound here looks good at first glance. The first term, namely $\mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(q,t')$, is saying the security of F as a PRF is related to that of E as a PRP for essentially the same resources: we can't ask better. The last term, namely $q^2/2^{2n}$, is negligible. What about the middle term, namely $q \cdot \mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(3,t')$? Intuitively, $\mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(3,t')$ is small: what can you do in three queries? This view is deceptive because one should not forget the time t'. One can spend it in exhaustive key search, and thus $\mathrm{CS{\scriptstyle EC}}_E^{\mathrm{prp}}(3,t')$ can be $\Omega(t'2^{-\kappa})$. But (dropping constants) this is at least $q2^{-\kappa}$ so the second term in our bound looks like $q^2 2^{-\kappa}$. Since $\kappa = n$ this is $q^2 2^{-n}$.

So these bounds are not proof that the security of F goes beyond the birthday bound. It would be nice to improve the above result. However, even the proof of the above is not exactly trivial, and this is one reason we include the result in this paper: we hope its ideas are food for thought towards an extension.

As far as we can tell, the difficulties in extending the above result are techncial rather than arising from any weakness in the construction. (We could be wrong.) Is there any other way we can give some meaningful evidence of the strength of the construction? We do this by analyzing it in the Shannon model.

5.2 Security in the ideal block cipher model

The theorem below looks at the most general version of the $F = \mathsf{Fn}^d E$ construction, when the number d of bits dropped is arbitrary and no restrictions are made on κ, n, in the model of Section 4.2,

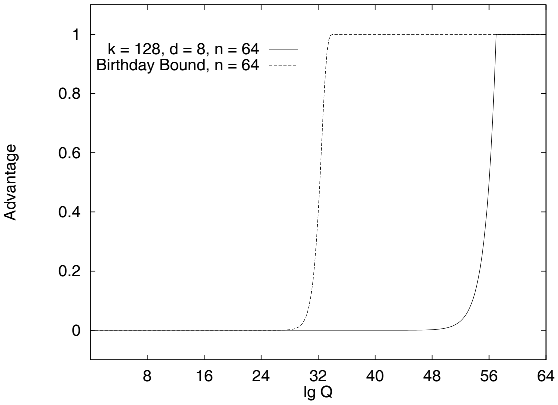

Figure 1: ***Right curve:*** *Illustrating Theorem 5.2, our upper bound on the adversary's advantage in distinguishing $F = \mathsf{Fn}^d E$ from a random function, assuming $n = 64$, $\kappa = 128$, and $d = 8$. Here E is a random permutation and the horizontal axis $Q = \max(q,t)$ is the maximum of the number of consecutive f-queries and the total number of E, E^{-1} queries.* ***Left curve:*** *The birthday bound for the same choice of parameters.*

where E is an ideal cipher. We obtain very strong results, showing security not only beyond the birthday bound, but nearly as good as one could hope for.

As we noted in Section 2, an important mode of operation for our construction will be when the values to which $F_{k_1 \ldots k_j}$ are being applied are successive counter values. Indeed, the bit dropping is done precisely to have maximum efficiency in this mode: as explained in Section 3, in this case, the amortized cost of computing F is just $(1 + j/2^d)$ times that of computing E, a negligible overhead. Accordingly, this is the case to which the following security analysis pertains. (Though later analyses are more general.)

Theorem 5.2 *Let n, κ be positive integers and $d, q, t, \hat{t}$ be non-negative integers with $0 \le d < n$ and let $\mathsf{F} = \mathsf{Fn}^d$. Let A be an adversary with three oracles, $E(\cdot,\cdot), E^{-1}(\cdot,\cdot)$, and $f(\cdot)$, who asks the numbers $0, \ldots, q-1$ of its f-oracle (so that these refer to $\hat{q} = \lceil q 2^{-d} \rceil$ common key groups), and asks at most t total queries of its E- and E^{-1}-oracles, these referring to no more than $\hat{t}$ common key groups. Let $j = \lceil \kappa/n \rceil$. Then $\mathrm{I}\textsc{Adv}^{\mathrm{prf}}_{\mathsf{F}}(A) \ \le$*

$$\frac{\hat{q}^5 + \hat{t}^5}{120} \cdot 2^{-4\kappa} + \Big(j^2 + 2j\hat{q} + tj + t\Big) \cdot 2^{-\kappa} + \hat{q} 2^{2d-n+3} + t\hat{t} 2^{d-n-\kappa+2} \ .$$

Proof: See Section 7. ▌

The first term bounding $\mathrm{I}\textsc{Adv}^{\mathrm{prf}}_{\mathsf{F}}(A)$ remains low until $q \approx 2^{4\kappa/5}$ or $t \approx 2^{4\kappa/5}$. We speculate that these conditions can be further improved to $2^{(1-\epsilon)\kappa}$ (and they are already very small in their current form), so a reasonable summary of $\mathrm{I}\textsc{Adv}^{\mathrm{prf}}_{\mathsf{F}}(A)$ is to say that the construction is good until $q \approx \min\{2^\kappa, 2^{n-d}\}$ or $t \approx \min\{2^\kappa, 2^{(n+\kappa)/2}\}$.

In Figure 1 we illustrate our bound for the case of a block cipher with parameters $n = 64$, $\kappa = 128$, and dropping $d = 8$ bits. The bound indicates that one must ask about 2^{55} queries before one can hope to distinguish F_k from a random function with advantage $1/e$. (This $1/e$-convention is a convenient way to summarize security.) For comparison, if you let $F = E$ you get the usual birthday-attack curve, which indicates that it takes but 2^{32} queries before an adversary can get like advantage at distinguishing E_k from a random function.

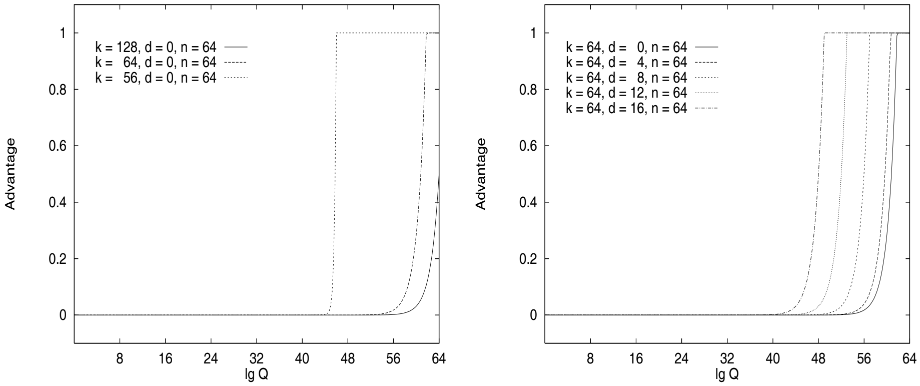

Figure 2: *Varying the parameters of Theorem 5.2 — our upper bound on the adversary's advantage in distinguishing $F = \mathsf{Fn}^d E$ from a random function, with the horizontal axis $Q = \max(q, t)$ as in the previous figure.* ***Left:*** *Varying key length κ.* ***Right:*** *Varying bits dropped d. For both pictures $n = 64$.*

In Figure 2 we illustrate our bound by showing the effect on advantage of changing either the key length (left-hand plot) or the value of d (right-hand plot). We assume a block size of $n = 64$ bits. The adversary's maximum advantage decreases with increasing key length, but this effect soon saturates. The construction has worse demonstrated security for larger values of d, but the effect is not that dramatic, and there is little reason to select a very large value of d, anyway.

It is important to understand the difference between the results here and those of Section 5.1. The "type" of security guarantee is better in the latter, since we are saying that security in the sense of a PRP (using the standard notion of a PRP) translates into security in the sense of a PRF (using the standard notion of a PRF). The results here are only about ideal ciphers, which only guarantees security against generic attacks. Yet, generic attacks are an important and easy to mount class of attacks, and a proof of security against them, especially with such strong bounds, is certainly meaningful. Eventually we hope strong results will emerge in the other model (as well as for other PRP-to-PRF constructions).

5.3 Attacks / Lower bounds

In Propositions 5.3 and 5.4 we present the best attacks that we know on our construction. These translate into lower bounds on the security of $\mathsf{Fn}^d E$. We present two adversaries: one which becomes successful when $q \approx 2^{n-d}$, and one which becomes successful when $t \approx 2^{\kappa}$. This is done in the Shannon model, but in this case (of attacks) this is not a restriction; if we can attack ideal ciphers we can certainly attack real ones. Thus, the results here should be viewed as complementing Theorem 5.2, telling us how close to tight is the analysis in the latter.

Proposition 5.3 *Let n, κ be positive integers and d, q non-negative integers with $0 \le d < n$, and let $\mathsf{F} = \mathsf{Fn}^d$. Then there is an adversary CS which asks at most q queries of an f oracle, no queries of the E or E^{-1} oracles, and achieves advantage*

$$\mathrm{I}\textsc{Adv}^{\mathrm{prf}}_{\mathsf{F}}(CS) \;\ge\; 1 - e^{-\lfloor q 2^{-d} \rfloor \cdot (2^d - 1) \cdot 2^{d-n-1}} \;.$$

Proof: See Section 5.3. ▌

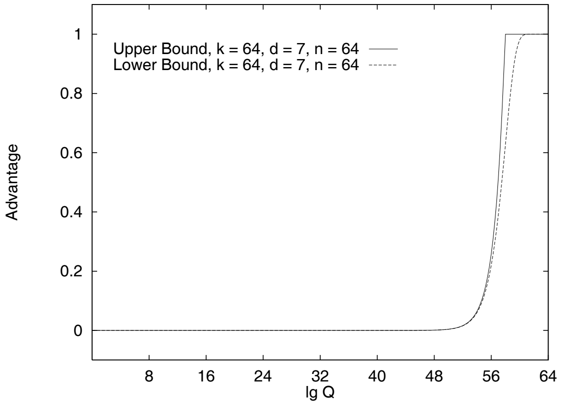

Figure 3: *With typical parameters our bounds are tight. Illustrating Propositions 5.3 and 5.4 and Theorem 5.2 for $n = 64$, $\kappa = 64$, $d = 7$. The horizontal axis Q is the same as in the previous figures.*

Proposition 5.4 *Let n, κ be positive integers and t, d, c be non-negative integers with $0 \leq d < n$, let $\mathsf{F} = \mathsf{Fn}^d$, and let $j = \lceil \kappa/n \rceil$. Then there is an adversary KS which asks c queries of her f oracle, t queries of her E oracle, and achieves advantage*

$$\mathrm{IA}\textsc{dv}^{\mathrm{prf}}_{\mathsf{F}}(KS) \;=\; \min\{1, \lfloor t/(cj+c) \rfloor \cdot 2^{-j\kappa}\} - t2^{-cn} \;.$$

Proof: See Section 5.3. ▌

The first lower bound is around $1 - e^{-q2^{d-n-1}}$, while the second one is around $t2^{-j\kappa}$. These become significant when $q \approx 2^{n-d}$ or $t \approx 2^{j\kappa}$. The point of giving these lower bounds is to see how tight is Theorem 5.2. As Figure 3 illustrates, the bounds are quite close for realistic parameters. On the same plot we graph our upper and lower bound for $\kappa = 56$, $n = 64$, and $d = 7$. The curves almost coincide.

6 Proof of Theorem 5.1

Refer to Section 5.1 for the theorem statement and to Section 4.1 for the definitions of security. We now provide the proof.

Since the oracles we provide our adversaries are deterministic, we assume throughout and without loss of generality that no adversary ever repeats an oracle query. By Time_E we mean the worst-case amount of time required to calculate function E in our underlying (fixed) model of computation.

We use the notion of *multi-oracles* as in [3], to provide a framework in which to reason about intermediate constructions that arise in our analysis. A *multi-oracle* Ω is simply a sequence of oracles, with some rules as to how queries to the multi-oracle are answered by the individual oracles. In our setting, an adversary making q queries will be provided with a multi-oracle consisting of q functions, $f_1, \ldots, f_q$, each mapping n bits to n bits. The adversary's j-th query to the multi-oracle will be answered by f_j, for $j = 1, \ldots, q$. (That is, if the j-th query to Ω is x_j then the response is $f_j(x_j)$.) Note that in this game it is not possible to ask two queries of a single oracle, nor to ask queries in some different order: the adversary is effectively restricted to sequentially querying

$f_1, \ldots, f_q$ in that order, with exactly one query to each function. Furthermore, all queries $x_1, \ldots, x_q$ are distinct strings.

We will consider various possible multi-oracles. The first, represented pictorially, is

$$\Omega(0) \ : \qquad E_{E_k} \quad \cdots \quad \cdots \quad E_{E_k} \ ,$$

where $k \leftarrow \{0,1\}^\kappa$ is a random key and there is a total of q instances of E_{E_k} above. Next come two classes, or types, of multi-oracles, and in each type there are $q+1$ different multi-oracles, so that we have $\Omega(s,i)$ for $i = 0, \ldots, q$ and $s = 1, 2$. We typically want to visualize and compare the i-th members of each class. These are represented pictorially below. In each case $\pi_{i+1}, \ldots, \pi_q \leftarrow \mathrm{Perm}_n$ are randomly and independently chosen permutations, and $k_1, \ldots, k_{i-1}$ are random, *distinct* κ-bit keys.

$$\begin{array}{lllllllll} \Omega(1,i) \ : & E_{k_1} & \cdots & E_{k_{i-1}} & E_{k_i} & \pi_{i+1} & \cdots & \pi_q & \qquad k_i \leftarrow \{0,1\}^\kappa - \{k_1, \ldots, k_{i-1}\} \\ \Omega(2,i) \ : & E_{k_1} & \cdots & E_{k_{i-1}} & \pi_i & \pi_{i+1} & \cdots & \pi_q & \qquad \pi_i \leftarrow \mathrm{Perm}_n \end{array}$$

In other words, in $\Omega(1,i)$, the i-th oracle is encryption under a key k_i distinct from those of the previous oracles. In $\Omega(2,i)$ the i-th oracle is a random permutation independent of anything else. Observe that $\Omega(1,i) = \Omega(2,i-1)$ for $i = 1, \ldots, q$; this is something we will use later. Now, for $s = 0, 1, 2$ and $i = 0, \ldots, q$ we let

$$\omega(s,i) \ = \ \Pr\left[A^{\Omega(s,i)} = 1\right] \quad \text{and} \quad \omega(0) \ = \ \Pr\left[A^{\Omega(0)} = 1\right]$$

be the probability that A outputs 1 in the game where it is provided with the corresponding multi-oracle, the probability being over the choice of the multi-oracle as discussed above, and over the coins of A, if any. We now claim that

$$\begin{aligned} \omega(0) &= \Pr\left[\, k \leftarrow \{0,1\}^\kappa \, ; g \leftarrow E_{E_k} \ : \ A^g = 1 \,\right] \\ \omega(1,0) &= \Pr\left[\, g \leftarrow \mathrm{Rand}_n \ : \ A^g = 1 \,\right] . \end{aligned}$$

The first equality follows from the definition of $\Omega(0)$. For the second, observe that $\Omega(1,0)$ consists of q random, independent permutations, $\pi_1, \ldots, \pi_q$. The adversary is making exactly one query to each of these, so the responses are independently and uniformly distributed over $\{0,1\}^n$. Thus the equality is true.

Thus our goal is to bound $\omega(0) - \omega(1,0)$. We will do so by comparing both to $\omega(1,q)$. The proofs of the following lemmas appear later.

Lemma 6.1 $\omega(0) - \omega(1,q) \le \mathrm{CSEC}_E^{\mathrm{prp}}(q, t')$.

Lemma 6.2 $\omega(1,q) - \omega(1,0) \ \le \ q \cdot \mathrm{CSEC}_E^{\mathrm{prp}}(3, t') + \dfrac{q^2}{2^{2n}}$.

Now we can write

$$\omega(0) - \omega(1,0) \ = \ [\omega(0) - \omega(1,q)] \ + \ [\omega(1,q) - \omega(1,0)]$$

And then apply the two lemmas above to obtain the bound in the theorem. So to complete the proof of the theorem we need to prove the two lemmas. The first is quite straightforward; the second will take work.

Proof of Lemma 6.1: We bound the quantity in question via the advantage of a distinguisher D (for E versus Perm_n) that we will construct below. It gets an oracle for a function g which is either E_k for a random k or is $\pi \leftarrow \mathrm{Perm}_n$ and wants to tell which. It uses A as a subroutine and will respond to oracle queries in such a way that A is working with multi-oracle $E_g, E_g, \ldots, E_g$. The code for D is as follows:

```
Algorithm $D^g$
    Run $A$, replying to the $j$-th oracle query $x_j$ of $A$ by $E_{g(x_j)}(x_j)$
    Output whatever $A$ outputs and halt
```

We now claim that

$$\begin{aligned}\Pr\left[\, k \leftarrow \{0,1\}^\kappa \,;\, g \leftarrow E_k \;:\; D^g = 1 \,\right] &= \omega(0) \\ \Pr\left[\, g \leftarrow \mathrm{Perm}_n \;:\; D^g = 1 \,\right] &= \omega(1,q)\ .\end{aligned}$$

The first is clear. For the second, note the sequence of auxiliary keys used to answer the queries when $g \leftarrow \mathrm{Perm}_n$ will be outputs of g on distinct points, hence random, distinct keys, which matches the definition of $\Omega(1,q)$.

Now, note that D makes q oracle queries and has a running time bounded by that of A plus $q\cdot\mathrm{Time}_E$ plus overhead, making it at most t'. Thus, we know that its advantage is at most $\mathrm{CSEC}_E^{\mathrm{prp}}(q,t')$. ▮

Proof of Lemma 6.2: We bound the quantity in question via the advantage of a distinguisher D (for E versus Perm_n) that we will construct below. It gets an oracle for a function g which is either E_k for a random k or is $\pi \leftarrow \mathrm{Perm}_n$ and wants to tell which. It uses A as a subroutine. Before specifying the code and analysis let us try to give an idea of the issues.

D will try to respond to oracle queries of A in such a way that A is working with multi-oracle

$$E_{k_1} \quad \cdots \quad E_{k_{i-1}} \quad g \quad \pi_{i+1} \quad \cdots \quad \pi_q \tag{1}$$

where $k_1,\ldots,k_{i-1}$ are random but distinct keys, and $\pi_{i+1},\ldots,\pi_q$ are random, independent permutations. D can "simulate" the first $k-1$ oracles by choosing random but distinct keys $k_1,\ldots,k_{i-1}$ and responding to a query to the j-th oracle ($j = 1,\ldots,i-1$) via $E_{k_j}(\cdot)$. Simulation of the $(i+1)$-th to q-th oracles is even easier: since each is called exactly once, D can just return a random number in response to each query. Now, we would like that if $g \leftarrow E_k$ for a random k then the oracle provide to A in the simulation looks like $\Omega(1,i)$, and if $g \leftarrow \pi$ for a random permutation π then it looks like $\Omega(2,i)$. However, neither of these wishes is easily realizable. Consider the first, namely the case where $g = E_k$ for a random k. For the oracle provided to A in the simulation to be $\Omega(1,i)$ it must be that $k \notin \{k_1,\ldots,k_{i-1}\}$. Although this happens with some probability, namely $1-(i-1)/2^\kappa$, D does not know whether or not this happens. (And we can't just neglect this, because then it turns out the bound would not be of good quality.) Therefore the idea is to have D try to figure this out: it will run a certain test whose purpose is to accept if $k \in \{k_1,\ldots,k_{i-1}\}$ and reject otherwise. The test is to compute g on m values, where m is some parameter whose value influences the analysis, and compare this to E_{k_j} evaluated on the same values, for $j = 1,\ldots,i-1$. Now the problem is that this test might accept even though k is not in fact one of $k_1,\ldots,k_{i-1}$, and the analysis must take that into account.

Let us now specify the code. We will then give the analysis. Below, $m > 0$ is an integer parameter whose value we will specify later and $\langle l\rangle$ is the n-bit binary representation of integer l.

```
Algorithm $D^g$
    Let $i \leftarrow \{1,\ldots,q\}$
    Let $k_1,\ldots,k_{i-1}$ be random but distinct $\kappa$-bit strings
    Let $r_{i+1},\ldots,r_q \leftarrow \{0,1\}^n$
```

```
For $l = 0$ to $m-1$ do
    $y_l \leftarrow g(\langle l \rangle)$
end for
$j \leftarrow 1$
While $(j \le i-1)$ do
    If $[E_{k_j}(\langle 0 \rangle) = y_0$ and $\ldots$ and $E_{k_j}(\langle m-1 \rangle) = y_{m-1}]$
        then return 1 (and halt)
    $j \leftarrow j+1$
end while
Run $A$, replying to the $j$-th oracle query $x_j$ of $A$ as follows:
    if $j < i$ then reply by $E_{k_j}(x_j)$
    if $j = i$ then reply by $g(x_j)$
    if $j > i$ then reply by $r_j$
Return whatever $A$ outputs (and halt)
```

We refer to $[E_{k_j}(\langle 0 \rangle) = y_0$ **and** $\ldots$ **and** $E_{k_j}(\langle m-1 \rangle) = y_{m-1}]$ as the "equality test for key k_j". For the analysis, let

$$\begin{aligned} \tau(1) &= \Pr\left[\, k \leftarrow \{0,1\}^{\kappa} \,;\, g \leftarrow E_k \,:\, D^g = 1 \,\right] \\ \tau(0) &= \Pr\left[\, g \leftarrow \mathrm{Perm}_n \,:\, D^g = 1 \,\right] . \end{aligned}$$

We now claim a certain lower bound on $\tau(1)$ which will be justified below:

$$\tau(1) \;\ge\; \frac{1}{q}\sum_{i=1}^{q}\left(1 - \frac{i-1}{2^{\kappa}}\right)\omega(1,i) + \frac{i-1}{2^{\kappa}} \tag{2}$$

$$\ge\; \frac{1}{q}\sum_{i=1}^{q}\omega(1,i) \; . \tag{3}$$

The second inequality is just arithmetic, but we do have to justify the first. In particular, it would appear that we have not accounted for the equality test at all, but in fact we have.

Equation (2) is justified like this. With probability $(i-1)/2^{\kappa}$ it will be the case that $k \in \{k_1, \ldots, k_{i-1}\}$. (The probability is *exactly* this because $k_1, \ldots, k_{i-1}$ are distinct.) In this case, the appropriate equality test (namely the one for k_j where $k_j = k$) is sure to return true and D will certainly output 1. This accounts for the second term in Equation (2). Now, with probability $1 - (i-1)/2^{\kappa}$, $k \notin \{k_1, \ldots, k_{i-1}\}$. In this case, we would like to have the equality tests fail so that we are providing A with the multi-oracle of Equation (1). If this would happen, we would have Equation (2) with an equality, not an inequality. But some test may succeed. In fact for any key $k \notin \{k_1, \ldots, k_{i-1}\}$ there is a certain probability $p(k)$ that the test succeeds, and this means that each key reaches the simulation part of the code with a different probability. However, the key observation is that if the test succeeds in these bad cases, D will output 1. So the overall probability of outputting one cannot decrease relative to the case where the tests do not succeed, so what we have written is indeed a lower bound.

Now, we upper bound $\tau(0)$ as follows:

$$\tau(0) \;\le\; \frac{1}{q}\sum_{i=1}^{q}\left(\omega(2,i) + (i-1)\cdot\prod_{l=0}^{m-1}\frac{1}{2^n - l}\right) \tag{4}$$

$$\leq \left[\frac{1}{q}\sum_{i=1}^{q}\omega(1,i-1)\right] + \frac{q-1}{2}\cdot\frac{1}{2^n}\frac{1}{2^{(m-1)(n-1)}} \ . \tag{5}$$

Equation (4) is justified by observing that the chance of an equality test for a particular key k_j succeeding when g is a random permutation is at most the product above, and there are $i-1$ keys tested. On the other hand, the probability of reaching the simulation is certainly only decreased, so the probability of D outputting 1 via A can't exceed $\omega(2,i)$. To get Equation (5) we are first using the observation made above that $\Omega(2,i)$ is just $\Omega(1,i-1)$. On the other hand we are simplifying the second term, using our assumption that $q \leq 2^{n-1}$.

We can now lower bound the difference (namely the advantage of D):

$$\begin{aligned}
\mathrm{CA{\scriptsize DV}}_E^{\mathrm{prp}}(D) &= \tau(1)-\tau(0) \\
&\geq \left[\frac{1}{q}\sum_{i=1}^{q}\omega(1,i)-\omega(1,i-1)\right] - \frac{q-1}{2^{n+1}}\frac{1}{2^{(m-1)(n-1)}} \\
&= \frac{1}{q}[\omega(1,q)-\omega(1,0)] - \frac{q-1}{2^{n+1}}\frac{1}{2^{(m-1)(n-1)}} \ .
\end{aligned}$$

The simplification came about because the sum "telescoped". Now, multiply both sides of the above by q and transpose terms to get

$$\omega(1,q)-\omega(1,0) \ \leq \ q\cdot \mathrm{CA{\scriptsize DV}}_E^{\mathrm{prp}}(D) + \frac{q(q-1)}{2^{n+1}}\frac{1}{2^{(m-1)(n-1)}} \ .$$

The second term can be made arbitrarily small by increasing the parameter m. Let us decide to set $m=2$. Now, notice that D makes $m+1=3$ queries to its oracle g, and its running time is bounded by t', so that $\mathrm{CA{\scriptsize DV}}_E^{\mathrm{prp}}(D) \leq \mathrm{CS{\scriptsize EC}}_E^{\mathrm{prp}}(m,t')$. Thus we conclude that

$$\omega(1,q)-\omega(1,0) \ \leq \ q\cdot \mathrm{CS{\scriptsize EC}}_E^{\mathrm{prp}}(3,t') + \frac{q^2}{2^{2n}} \ .$$

This completes the proof of Lemma 6.2. ▌

7 Proof of Theorem 5.2

Refer to Section 5.1 for the theorem statement. We now provide the proof.

Since the oracles we provide our adversaries are deterministic, we assume throughout and without loss of generality that no adversary ever repeats an oracle query. Sometimes we regard a block cipher as a two-dimensional table with 2^κ rows and 2^n columns, where $E(k,x)$ is the value in the cell of the k-th row and x-th column.

Given a partial function f from (a subset of) $\{0,1\}^n$ to (a subset of) $\{0,1\}^n$, we denote the domain and range of f (the points where f has been defined and the values those domain points map to) by $\mathrm{Dom}(f)$ and $\mathrm{Range}(f)$, respectively. Define $\overline{\mathrm{Dom}}(f) = \{0,1\}^n - \mathrm{Dom}(f)$ and $\overline{\mathrm{Range}}(f) = \{0,1\}^n - \mathrm{Range}(f)$.

When an oracle's algorithm is specified in pseudo-code having a Boolean variable bad_i, BAD_i is the event that flag bad_i is set true and is the first such bad flag to be set by the algorithm.

7.1 Lemmas

The proofs in this section use two lemmas which are independent of the rest of the section. We give them here.

The first lemma bounds the ability of an adversary to distinguish the output from two nearly identical programs. When we write two algorithms which simulate two oracles, we specify the algorithms to be syntactically identical for as much of their specification as possible. Where their specifications diverge, a flag is set, and we bound the advantage of an adversary based on her ability to set one of these flags. See Figures 4 and 5 for examples. The basis for this approach is founded on Lemma 7.1.

The second standard lemma gives upper and lower bounds for the birthday phenomena in Lemma 7.3.

Distinguishing Nearly Identical Programs. Consider an adversary A and her oracle f, and assume A is defined to output either 0 or 1. Say that f is set to either program P_1 or P_2, and that the advantage A has in distinguishing which is the case is $\mathrm{Adv}_A = \Pr_1[A=1] - \Pr_2[A=1]$. Now consider the case where P_1 and P_2 are syntactically identical except for some **if**-guarded instructions in P_2 which, if executed, set a boolean flag *bad*. Let BAD be the event in P_2 that *bad* is set.

Lemma 7.1 $\mathrm{Adv}_A \le \Pr_2[\mathsf{BAD}]$.

Proof: Let C be the set of all infinite strings representing the coins used in the experiment. Classify the elements of C into four non-overlapping sets, C_{12}, $C_{\bar{1}2}$, $C_{1\bar{2}}$ and $C_{\bar{1}\bar{2}}$, where the elements of C_{12} cause $(A^{P_1} = 1 \wedge A^{P_2} = 1)$, and the elements of $C_{\bar{1}2}$ cause $(A^{P_1} = 0 \wedge A^{P_2} = 1)$, etc. Then,

$$\begin{aligned}
\mathrm{Adv}_A &= \Pr_C[A^{P_1} = 1] - \Pr_C[A^{P_2} = 1] \\
&= \frac{|C_{12}| + |C_{1\bar{2}}|}{|C|} - \frac{|C_{12}| + |C_{\bar{1}2}|}{|C|} \\
&= \frac{|C_{1\bar{2}}| - |C_{\bar{1}2}|}{|C|} \\
&\le \frac{|C_{1\bar{2}}| + |C_{\bar{1}2}|}{|C|} \\
&= \Pr_C[A^{P_1} \ne A^{P_2}] \\
&\le \Pr_2[\mathsf{BAD}]
\end{aligned}$$

To see the last step, if a set of coins does not cause the *bad* flag to be marked, then only shared code is executed, and P_1 and P_2 have identical output. Therefore, A can only have advantage on the coins selected which set *bad*. ▮

Corollary 7.2 *If P_1 and P_2 are identical except for some* **if***-guarded instructions in P_1 which if executed set bad_1 and some* **if***-guarded instructions in P_2 which if executed set bad_2, then* $\mathrm{Adv}_A \le \Pr_1[\mathsf{BAD}_1] + \Pr_2[\mathsf{BAD}_2]$.

Proof: Let program P_3 be identical to the common parts of P_1 and P_2. Then, $\Pr_1[A = 1] - \Pr_2[A = 1] \le (\Pr_1[A = 1] - \Pr_3[A = 1]) + (\Pr_3[A = 1] - \Pr_2[A = 1]) \le \Pr_1[\mathsf{BAD}_1] + \Pr_2[\mathsf{BAD}_2]$ ▮

Lemma 7.3 *[Birthday Phenomenon] Given n balls tossed independently and randomly into m bins, the probability that at least one bin has more than one ball, $C(n,m)$, satisfies $1-e^{-n(n-1)/2m} \le C(n,m) \le n^2/2m$.* ▮

7.2 Proof of Theorem 5.2, Part 1

In the ideal model the adversary has access to E, E^{-1}, and $(\mathsf{Fn}^d E)(k,\cdot)$ oracles. However, we initially envision an adversary with access only to the last of these. Later we correct for this simplifying assumption. Modularizing the proof in this way makes this already-complex argument easier to follow.

Lemma 7.4 *Let n, κ be positive integers and $d, \hat{q}$ be non-negative integers. Let $j = \lceil \kappa/n \rceil$. Let A be an adversary with a single oracle, f, and suppose A asks f queries referring to no more than $\hat{q}$ common key groups. Then* $\mathrm{Adv}^1_A \overset{\text{def}}{=}$

$$\Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}, f(\cdot) \overset{\text{def}}{=} \mathsf{Fn}^d E(k,\cdot) : A^f = 1\right] - \Pr\left[\rho \leftarrow \mathrm{Rand}_n : A^\rho = 1\right]$$

$$\le j^2 2^{-\kappa-1} + \frac{\hat{q}^5}{120} \cdot 2^{-4\kappa} + \hat{q}2^{2d-n+3} + \hat{q}j2^{-\kappa} \ .$$

Note that if an adversary is restricted to referring to no more than $\hat{q}$ common key groups, implicitly she is restricted to no more than $\hat{q}2^d$ total queries.

Proof: To prove the bound we devise an algorithm to simulate an oracle for the adversary. Actually, there are two algorithms developed. Both are indicated in Figure 4, the difference being whether or not we set the flag *Game2*. We call "Game 1" the result of running the specified algorithm with the flag *Game2* set to false, and we call "Game 2" the result of running the specified algorithm with the flag *Game2* set to true.

The idea of these games is to simulate one of two experiments —the exact two experiments used in the definition of Adv^1_A— and to structure these simulations so that they are "identical" until this can be maintained no longer. Game 1 simulates the experiment used to define the second addend of the adversary's advantage. Game 2 simulates the experiment used to define the first addend of the adversary's advantage. When Games 1 and 2 "diverge," a flag will be set. Bounding the probability that any of the game's flags get set will serve to bound Adv^1_A.

Let $p_2 = \Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}, f(\cdot) \overset{\text{def}}{=} \mathsf{Fn}^d E(k,\cdot) : A^f = 1\right]$ denote the first addend of the adversary's advantage in Lemma 7.4. Similarly, let $p_1 = \Pr\left[\rho \leftarrow \mathrm{Rand}_n : A^\rho = 1\right]$ denote the second addend. Let $\Pr_i[E]$ denote the probability of event E with respect to the probability space induced by Game i. Our definitions of an oracle F in Game 1 and Game 2 (Figure 4) make the following clear.

Claim 7.5 $\Pr_1\left[A^F = 1\right] = p_1$.

Claim 7.6 $\Pr_2\left[A^F = 1\right] = p_2$.

To bound $|p_1 - p_2|$ we bound an adversary's advantage in differentiating between Game 1 and Game 2. The following claim is a direct result of Lemma 7.1.

```
On initialization:

$E(\cdot,\cdot)$ is undefined
$bad_1, bad_2, bad_3 \leftarrow false$
$k_1, \ldots, k_j \leftarrow \{0,1\}^\kappa$
$ukey \leftarrow \{k_1^*\} \cup \cdots \cup \{k_j^*\}$
if $Game2$ and $|ukey| < j$ then
    $bad_1 = true$
for $i \leftarrow 1, \ldots, j$ do
    $E(k_i, \cdot) \leftarrow \mathrm{Perm}_n$
```

```
On oracle query $F(x)$:

$y \leftarrow \{0,1\}^n$
$x' \leftarrow x \gg d$
$k' \leftarrow [E(k_1,x') \| \cdots \| E(k_j,x')]_{1..\kappa}$
if $Game2$ and $k' \in ukey$ then
    $bad_2 = true$
    return $E(k',x)$
if $Game2$ and $y \in \mathrm{Range}(E(k',\cdot))$ then
    $bad_3 = true$
    $y \leftarrow \overline{\mathrm{Range}}(E(k',\cdot))$
define $E(k',x) = y$
return $y$
```

Figure 4: Game 1 *(when Game2 = false) and* Game 2 *(otherwise).*

Claim 7.7 $|\Pr_1\left[A^F = 1\right] - \Pr_2\left[A^F = 1\right]| \le \sum_{i=1}^{3} \Pr_2[\,\mathsf{BAD}_i\,]$.

As a result of this claim, we need only bound the three events BAD_1, BAD_2 and BAD_3 in Game 2.

BOUNDING BAD_1. The values $k_1 \ldots, k_j$ are uniformly sampled and their collision probability is upper bounded with the birthday bound, giving $\Pr_2[\,\mathsf{BAD}_1\,] \le j^2 2^{-\kappa-1}$.

BOUNDING BAD_2. Recall that, by our convention, the event BAD_I can only occur when BAD_i does not occur for all $i < I$. Event BAD_I is the event that bad_I is the *first bad* flag to be set. Therefore, $\Pr_2[\,\mathsf{BAD}_2\,] \le \Pr_2[\,\mathsf{BAD}_2|\overline{\mathsf{BAD}_1}\,]$, and we analyze BAD_2 in the context that $k_1, \ldots, k_j$ are random but distinct values. It is clear that each distinct set of keys $k_1, \ldots, k_j$ gives rise to the *same* distribution on derived keys: the value of the underlying key is not significant, it is only a "name" for referring to one of the permutations. Thus we could just as well have *first* choosen the derived keys from the appropriate distribution, and only then chosen the underlying keys $k_1, \ldots, k_j$ (all of them distinct). Conducting the experiment in this way makes it clear that the chance that an underlying key and a derived key coincide (given that the underlying keys are distinct) is at most $\hat{q} j 2^{-\kappa}$, since there are at most $\hat{q}$ derived keys out of the 2^κ possible ones, and whatever the derived keys are, we subsequently choose j random distinct keys and look to see if there is a collision.

BOUNDING BAD_3. Let BAD^* be the event in Game 2 that some collection of more than 4 common key groups all map to the same k' value. (We choose the number 4 to be concrete; the proof works with other numbers, but 4 yields a good result and simplifies the exposition.) We bound $\Pr_2[\,\mathsf{BAD}_3\,]$ by

$$\begin{aligned}\Pr_2[\,\mathsf{BAD}_3\,] &= \Pr_2[\,\mathsf{BAD}_3|\mathsf{BAD}^*\,] \cdot \Pr_2[\,\mathsf{BAD}^*\,] + \Pr_2[\,\mathsf{BAD}_3|\overline{\mathsf{BAD}^*}\,] \cdot \Pr_2[\,\overline{\mathsf{BAD}^*}\,] \\ &\le \Pr_2[\,\mathsf{BAD}^*\,] + \Pr_2[\,\mathsf{BAD}_3|\overline{\mathsf{BAD}^*}\,]\end{aligned}$$

We now bound each summand.

BOUNDING $\Pr_2[\,\mathsf{BAD}^*\,]$. If $\kappa \ge n$, then $\Pr_2[\,\mathsf{BAD}^*\,] = 0$ because the first n bits of each k' will be the result of a permutation on differing group selector x' values, hence these values will be different for each common key group. In the case where $\kappa < n$, each k' is generated by a single n-bit permutation, with the trailing $n-\kappa$ bits deleted. This results in as many as $\min(2^{n-d}, 2^{n-\kappa})$ common key groups mapping to each k'-value. For some 5 common key groups $x'_1, \ldots, x'_5$ to map to the same derived key, the permuted values of $x'_1, \ldots, x'_5$ must agree in the first κ bits. The probability of this is no more than $2^{-4\kappa}$. The adversary is restricted to $\hat{q}$ common key groups, so given $\binom{\hat{q}}{5}$ ways of grouping the common key groups into groups of size 5, $\Pr_2[\,\mathsf{BAD}^*\,] \le \binom{\hat{q}}{5} \cdot 2^{-4\kappa} \le \frac{\hat{q}^5}{5!} \cdot 2^{-4\kappa}$.

Bounding $\Pr_2[\,\mathsf{BAD}_3 | \overline{\mathsf{BAD}^*}\,]$. Each k' has no more than 4 common key groups mapped to it, each of size 2^d. Also, no more than $\hat{q}$ different k' values are mapped to. Using a birthday bound (Lemma 7.3), $\Pr_2[\,\mathsf{BAD}_3 | \overline{\mathsf{BAD}^*}\,] \le \hat{q} \cdot C(4 \cdot 2^d, 2^n) = \hat{q} 2^{2d-n+3}$. ▮

7.3 Proof of Theorem 5.2, Part 2

We now include the oracles E, E^{-1}. Clearly we can give our adversary A from the previous section a block cipher G, G^{-1} unrelated to $(\mathsf{Fn}^d E)(k, \cdot)$ and her advantage will not be increased by querying G, G^{-1}. So another way to express Adv_A^1 is

$$\begin{aligned}\mathrm{Adv}_A^1 \;=\; & \Big| \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; G \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : \\ & \quad A^{G,G^{-1},f} = 1 \Big] - \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; \rho \leftarrow \mathrm{Rand}_n : A^{E,E^{-1},\rho} = 1 \Big] \Big|\end{aligned}$$

Recall that we aim to bound

$$\begin{aligned}\mathrm{Adv}_A \;=\; & \Big| \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : A^{E,E^{-1},f} = 1 \Big] - \\ & \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; \rho \leftarrow \mathrm{Rand}_n : A^{E,E^{-1},\rho} = 1 \Big] \Big| \;.\end{aligned}$$

The second summand for each of the above two expressions are identical, so the next goal is to bound the the difference of the first summands. In summary, the idea is to show that there is very little difference in adding oracles G, G^{-1} unrelated to $(\mathsf{Fn}^d E)(k, \cdot)$ and adding the "real" oracles E, E^{-1}.

Lemma 7.8 *Let n, κ be positive integers and $d, q, t, \hat{t}$ be non-negative integers. Let $j = \lceil \kappa/n \rceil$ and $\hat{q} = \lceil q 2^{-d} \rceil$. Let A be an adversary with three oracles, $E(\cdot,\cdot), E^{-1}(\cdot,\cdot)$, and $f(\cdot)$, who asks the numbers $0, \ldots, q-1$ of its f-oracle, and at most t total queries, referring to no more than $\hat{t}$ common key groups, of its E- and E^{-1}-oracles. Then*

$$\begin{aligned}\mathrm{Adv}_A^2 \;\stackrel{\mathrm{def}}{=}\; & \Big| \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : A^{E,E^{-1},f} = 1 \Big] - \\ & \Pr\Big[E \leftarrow \mathrm{BC}_{\kappa,n}; G \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : A^{G,G^{-1},f} = 1 \Big] \Big| \\ \le\; & \left(\frac{j^2}{2} + \hat{q} j + t j + t \right) \cdot 2^{-\kappa} + \frac{\hat{t}^5}{120} \cdot 2^{-4\kappa} + t \hat{t} 2^{d-n-\kappa+2} \;.\end{aligned}$$

Proof: To prove the bound we devise an algorithm to simulate a triple of oracles $\langle E, E^{-1}, F \rangle$ for the adversary. Actually, there are two algorithms developed. Both are indicated in Figure 5, the difference being whether or not we set the flag *Game3*. We call "Game 3" the result of running the

```
On initialization:

$bad_1, \ldots, bad_7 \leftarrow false$
$F, HE, HF \leftarrow$ undefined
$k_1^*, \ldots, k_j^* \leftarrow \{0,1\}^\kappa$
$ukey \leftarrow \{k_1^*\} \cup \cdots \cup \{k_j^*\}$
if $Game3$ and $|ukey| < j$ then
   $bad_1 \leftarrow true$
$HF(k_1^*, \cdot), \ldots, HF(k_j^*, \cdot) \leftarrow \mathrm{Perm}_n$


On oracle query $F(x)$:

$x' \leftarrow x \gg d$
$k' \leftarrow [E(k_1, x') \| \cdots \| E(k_j, x')]_{1..\kappa}$
if $HF(k', x)$ defined then
   return $HF(k', x)$
if $Game3$ and $HE(k', x)$ defined then
   $bad_2 \leftarrow true$
   return $HE(k', x)$
$y \leftarrow \overline{\mathrm{Range}}(HF(k', \cdot))$
if $Game3$ and $y \in \mathrm{Range}(HE(k', \cdot))$ then
   $bad_3 \leftarrow true$
   $y \leftarrow \overline{\mathrm{Range}}(HE(k, \cdot)) \cap \overline{\mathrm{Range}}(HF(k, \cdot))$
define $HF(k', x) = y$
return $y$
```

```
On oracle query $E(k, x)$:

if $k \in ukey$ then
   $bad_4 \leftarrow true$
if $HE(k, x)$ defined then
   return $HE(k, x)$
if $Game3$ and $HF(k, x)$ defined then
   $bad_5 \leftarrow true$
   return $HF(k, x)$
$y \leftarrow \overline{\mathrm{Range}}(HE(k, \cdot))$
if $Game3$ and $y \in \mathrm{Range}(HF(k, \cdot))$ then
   $bad_6 \leftarrow true$
   $y \leftarrow \overline{\mathrm{Range}}(HE(k, \cdot)) \cap \overline{\mathrm{Range}}(HF(k, \cdot))$
define $HE(k, x) = y$
return $y$


On oracle query $E^{-1}(k, y)$:

if $k \in ukey$ then
   $bad_7 \leftarrow true$
if $HE^{-1}(k, y)$ defined then
   return $HE^{-1}(k, y)$
if $Game3$ and $y \in \mathrm{Range}(HF(k, \cdot))$ then
   $bad_8 \leftarrow true$
   return $HF^{-1}(k, y)$
$x \leftarrow \overline{\mathrm{Dom}}(HE(k, \cdot))$
if $Game3$ and $x \in \mathrm{Dom}(HF(k, \cdot))$ then
   $bad_9 \leftarrow true$
   $x \leftarrow \overline{\mathrm{Dom}}(HE(k, \cdot)) \cap \overline{\mathrm{Dom}}(HF(k, \cdot))$
define $HE(k, x) = y$
return $x$
```

Figure 5: Game 3 (when $Game3 = true$) and Game 4 (otherwise).

specified algorithm with the flag *Game3* set to true, and we call "Game 4" the result of running the specified algorithm with the flag *Game3* set to false.

The idea of these games is to simulate one of two experiments —the exact two experiments used in the definition of Adv_A^2— and to structure these simulations so that they are "identical" until this can be maintained no longer. Game 3 will simulate the first experiment in the expression for Adv_A^2, that is, the experiment associated to

$$p_3 \stackrel{\mathrm{def}}{=} \Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : A^{E, E^{-1}, f} = 1\right]$$

Game 4 will simulate the second experiment in the expression for Adv_A^2, that is, the experiment associated to $p_4 \stackrel{\mathrm{def}}{=}$

$$\Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}; G \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa}; f \stackrel{\mathrm{def}}{=} (\mathsf{Fn}^d E)(k, \cdot) : A^{G, G^{-1}, f} = 1\right].$$

When Games 3 and 4 "diverge," a flag will be set. Bounding the probability that this flag gets set will serve to bound Adv_A^2.

Games 3 and 4 were designed to make the following two claims clear:

Claim 7.9 $\Pr_3\left[A^{E,E^{-1},F} = 1\right] = p_3$.

Claim 7.10 $\Pr_4\left[A^{E,E^{-1},F} = 1\right] = p_4$.

Combining these claims and Lemma 7.1, the advantage adversary A can achieve is bounded:

Claim 7.11 $|p_3 - p_4| \leq \sum_{i=1}^{9} \Pr_3[\,\mathsf{BAD}_i\,]$.

Therefore, instead of directly considering adversaries who try to maximize $|p_3 - p_4|$, we may consider adversaries whose goal it is to set the bad_i flags in Game 3. Claim 7.11 tells us that $|p_3 - p_4|$ is no larger than the maximum probability an adversary can achieve in setting the flags in Game 3. For the remainder of this section, we consider in turn the maximum probability an adversary D has in setting each of the bad_i flags. The overall bound we wish to prove, $|p_3 - p_4|$, is no larger than the sum of these maximum probabilities. We now bound the maximum probability that an adversary has in causing each event BAD_i in Game 3. Recall our convention that BAD_i is the event that bad_i is the *first* flag to get set.

BOUNDING BAD_1. The underlying keys are uniformly distributed and so we bound their collision probability with a birthday bound. So, $\Pr_3[\,\mathsf{BAD}_1\,] \leq j^2 2^{-\kappa-1}$.

BOUNDING $\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9$. Each common key group shares a single derived key. The elements of the common key group along with their associated derived key together define a contiguous set of entries in the HF-table which we call *distinguished boxes*. The locations of these distinguished boxes are fixed during initialization by the fixing of $HF(k_1^*, \cdot), \ldots, HF(k_j^*, \cdot)$. The adversary is allowed F queries to no more than $\hat{q}$ common key groups, so there will never be more than $\hat{q}$ distinguished boxes with entries in them. (Any distinguished boxes which have derived keys which coincide with any of the underlying keys in *ukey* will also have their entries filled, but we do not consider those here because BAD events associated with them are subsumed by BAD_4 and BAD_7.) Furthermore, the distribution on the location of the $\hat{q}$ occupied distinguished boxes is unaffected by the ordering or content of the adversary's queries. We therefore consider an adversary who asks her F queries first.

If we consider the the projection onto the HE-table of the $\hat{q}$ occupied distinguished boxes from the HF-table, then the ability of of the adversary to ask her E-oracle a query which intersects one of the projected distinguished boxes serves as a bound on the three events $\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9$. Let BAD^* be the event in Game 3 that such an intersection ocurrs. Event $\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9$ cannot occur without such an intersection. And so,

$$\begin{aligned}
\Pr_3[\,\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9\,] &= \Pr_3[\,\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9 | \mathsf{BAD}^*\,]\Pr_3[\,\mathsf{BAD}^*\,] + \\
& \Pr_3[\,\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9 | \overline{\mathsf{BAD}^*}\,]\Pr_3[\,\overline{\mathsf{BAD}^*}\,] \\
&\leq 1 \cdot \Pr_3[\,\mathsf{BAD}^*\,] + 0 \cdot \Pr_3[\,\overline{\mathsf{BAD}^*}\,] \\
&= \Pr_3[\,\mathsf{BAD}^*\,]
\end{aligned}$$

Event BAD^* is really the union of $\hat{q}$ events: That the adversary asks a query of E which intersects projected distinguished box $dbox_i$, for $1 \leq i \leq \hat{q}$. By the principle of *incusion-exclusion* we thus bound BAD^* by a sum, $\Pr_3[\,\mathsf{BAD}^*\,] = \Pr_3[\,\cup_{i=1}^{\hat{q}} \mathit{dbox}_i\,] \leq \sum_{i=1}^{\hat{q}} \Pr_3[\,\mathit{dbox}_i\,]$. But, notice that each column of the *HE*-table has at most one distinguished box and that its derived key is uniformly distributed, and so $\Pr_3[\,\mathit{dbox}_i\,] = q_i 2^{-\kappa}$, where q_i is the number of queries the adversary asks in the column where the i-th distinguished box is projected. Finally, we have $\sum_{i=1}^{\hat{q}} \Pr_3[\,\mathit{dbox}_i\,] = \sum_{i=1}^{\hat{q}} q_i 2^{-\kappa} = q2^{-\kappa}$.

Bounding $\mathsf{BAD}_3 \cup \mathsf{BAD}_6 \cup \mathsf{BAD}_8$. None of the events comprising $\mathsf{BAD}_3 \cup \mathsf{BAD}_6 \cup \mathsf{BAD}_8$ occur unless an entry in $\mathit{HF}(k,\cdot)$ is also in $\mathit{HE}(k,\cdot)$, for any k. We therefore bound $\mathsf{BAD}_3 \cup \mathsf{BAD}_6 \cup \mathsf{BAD}_8$ on the adversary's ability to cause such a collision in the same manner we did in the previous paragraphs. Let BAD^* be the event that such a collison occurs, then $\Pr_3[\,\mathsf{BAD}_2 \cup \mathsf{BAD}_5 \cup \mathsf{BAD}_9\,] \leq \Pr_3[\,\mathsf{BAD}^*\,]$.

As in the proof of Lemma 7.4, we assume that no 4 common key groups map to the same derived key (see that proof for details). Thus, we consider the case where no $\mathit{HF}(k,\cdot)$ has more than $4 \cdot 2^d$ defined elements, and they are all random and distinct by definition. (Again, those entries associated witht the elements of *ukey* are bounded separately.) So, given that q_i E-oracle queries are made of the form $E(i,\cdot)$, no matter what their distribution, the chance of colliding with at least one of the $4 \cdot 2^d$ random distinct values from $\mathit{HF}(i,\cdot)$ is $1-(1-(q_i/(2^\kappa - 4\cdot 2^d)))^{4\cdot 2^d}$. We sum over all q_i. The sum is maximized when $q_i = q$ for a single value of i. Adding the term which compensates for our assumption that no 4 common key groups map to the same derived key, and we arrive at our bound, $1-(1-(q/(2^\kappa - 4\cdot 2^d)))^{4\cdot 2^d} + \frac{\hat{q}^5}{5!}\cdot 2^{-4\kappa}$

Bounding $\mathsf{BAD}_4 \cup \mathsf{BAD}_7$. The underlying keys $k_1,\ldots,k_j$ are uniformly distributed on $\{0,1\}^\kappa$. Furthermore, The value of each underlying key is not significant, it is only a "name" for referring to one of the permutations. $j2^{-\kappa}$.

The summation of these terms completes the bound of Lemma 7.8. ▌

By the triangle inequality, $\mathrm{Adv}_A \leq \mathrm{Adv}_A^1 + \mathrm{Adv}_A^2$, which concludes the proof of Theorem 5.2. ▌

Remark 7.12 If $\kappa \geq n$ then we can improve our bound to:

$$\mathrm{Adv}_A \quad \leq \quad \hat{q}2^{2d-n-1} + j^2 2^{-\kappa} + j\hat{q}2^{-\kappa+1} + t\lceil \kappa/n \rceil 2^{-\kappa} + t2^{-\kappa} + t\hat{t}2^{d-n-\kappa}\ .$$

Proof: If $\kappa \geq n$, then $\Pr_1[\,\mathsf{BAD}_1\,] \leq \hat{q}2^{2d-n-1}$ and $\Pr_3[\,\mathsf{BAD}_4\,] + \Pr_3[\,\mathsf{BAD}_7\,] \leq t\hat{t}2^{d-n-\kappa}$. Each common key group will be mapped to a different derived key k'. When $\kappa \geq n$, the keys are generated by a function which is the concatenation of $j \geq 1$ permutations, ensuring than no two inputs map to the same output. ▌

8 Analysis of attacks

Here we prove the lower bounds, namely the results of Section 5.3.

8.1 Proof of Proposition 5.3

Adversary *CS* looks for collisions within common key groups in the output of f. The attacks are specified in Figure 6.

Proof: If $f(\cdot) = (\mathsf{Fn}^d E)(k,\cdot)$ then each common key group is answered by a single permutation, and so $\Pr[E \leftarrow \mathrm{BC}_{\kappa,n}; k \leftarrow \{0,1\}^{j\kappa} : \mathit{CS}^{\mathsf{Fn}^d E(k,\cdot)} = 1] = 0$. Thus, advantage $\mathrm{I}\textsc{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathit{CS})$ is exactly

```
function $CS^f(q,d)$
for $i = 0 \ldots \lfloor q2^{-d} \rfloor - 1$ do
  if $|\{f(i2^d), f(i2^d+1), \ldots,$
      $f(i2^d + 2^d - 1)\}| < 2^d$
    then return 1
return 0
```

```
function $KS^{f,E}(t,d,j,c)$
Choose $K \subseteq \{0,1\}^{j\kappa}$ where
    $|K| = \min\{\lfloor t/(c(j+1)) \rfloor, 2^{j\kappa}\}$
for each $k_1 \cdots k_j \in K$ do
    $k' \leftarrow [E(k_1,i)\| \cdots \|E(k_j,i)]_{1 \ldots \kappa}$
    if $f(i2^d) = E(k', i2^d)$ for all $0 \le i \le c-1$
        then return 1
return 0
```

Figure 6: *Naive attacks.* ***Left:*** *collision-search adversary.* ***Right:*** *key-search adversary.*

$\Pr[\rho \leftarrow \mathrm{Rand}_n : CS^{\rho(\cdot)} = 1]$. This is easily bounded using Lemma 7.3. Let $Q = \lfloor q2^{-d} \rfloor$.

$$\begin{aligned}
\mathrm{IA{\small DV}}^{\mathrm{prf}}_{\mathsf{F}}(CS) &= \Pr\left[\rho \leftarrow \mathrm{Rand}_n : CS^{\rho(\cdot)} = 1\right] \\
&= \Pr\left[\rho \leftarrow \mathrm{Rand}_n : \bigvee_{i=1}^{Q} \left[\exists l,j : (i-1)2^d \le l < j \le i2^d - 1 : \rho(l) = \rho(j)\right]\right] \\
&= 1 - \Pr\left[\rho \leftarrow \mathrm{Rand}_n : \bigwedge_{i=1}^{Q} \left[\forall l,j : (i-1)2^d \le l < j \le i2^d - 1 : \rho(l) \ne \rho(j)\right]\right] \\
&= 1 - \prod_{i=1}^{Q} \Pr\left[\rho \leftarrow \mathrm{Rand}_n : \left[\forall l,j : 0 \le l < j \le 2^d - 1 : \rho(l) \ne \rho(j)\right]\right] \\
&\ge 1 - e^{-Q(2^d-1)2^{d-n-1}}
\end{aligned}$$

▌

8.2 Proof of Proposition 5.4

Adversary KS makes a small number (c) of f-queries and a large number (t) of E-queries. (With typical values of κ,n, imagine $c = 2$ or $c = 3$.) The adversary simply guesses a key and then tries to confirm that guess. The attack is again specified in Figure 6.

Proof: If f is a random function then there is a small chance that KS will incorrectly identify it as an instance of $\mathsf{Fn}^d E$. For this to happen some $k_1, \ldots, k_j$ must collide with f's output. This occurs with chance only 2^{-nc} for each of the t queries, and so $\Pr\left[E \leftarrow \mathrm{BC}_{\kappa,n}; f \leftarrow \mathrm{Rand}_n : KS^{f,E}(t,d,j,c) = 1\right]$ is no more than $t2^{-nc}$.

If f is an instance of $\mathsf{Fn}^d E$, then the chance that the algorithm outputs 1 is at least as much as the probability that the algorithm guesses the random key set correctly. We try $\min\{\lfloor t/(c(j+1)) \rfloor, 2^{j\kappa}\}$ out of a total $2^{j\kappa}$ possible keys, from which the result now follows. ▌

References

[1] W. Aiello and R. Vanketesan, "Foiling birthday attacks in output-doubling transformations." *Advances in Cryptology – Eurocrypt 96 Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.

[2] M. Bellare, O. Goldreich and H. Krawczyk, personal communications, 1995.

[3] M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security." *Proceedings of the* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996.

[4] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption." *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.

[5] M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using a finite pseudorandom function." *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

[6] M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining." *Advances in Cryptology – Crypto* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[7] S. Even and Y. Mansour, "A construction of a cipher from a single pseudorandom permutation." *Advances in Cryptology – ASIACRYPT* 91 *Proceedings*, Lecture Notes in Computer Science Vol. 739, H. Imai, R. Rivest and T. Matsumoto ed., Springer-Verlag, 1991.

[8] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM,* Vol. 33, No. 4, 1986, pp. 210–217.

[9] S. Goldwasser and S. Micali, "Probabilistic encryption." *J. of Computer and System Sciences*, Vol. 28, April 1984, pp. 270–299.

[10] J. Kilian and P. Rogaway, "How to protect DES against exhaustive key search." *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[11] M. Luby, *Pseudorandomness and Crpyptographic Applications.* Princeton University Press, 1996.

[12] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions." *SIAM J. Comput,* Vol. 17, No. 2, April 1988.

[13] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard." *Advances in Cryptology – Crypto* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994, pp. 1–11.

[14] U. Maurer, "A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators." *Advances in Cryptology – Eurocrypt* 92 *Proceedings*, Lecture Notes in Computer Science Vol. 658, R. Rueppel ed., Springer-Verlag, 1992, pp. 239–255.

[15] M. Naor and O. Reingold, "On the construction of pseudo-random permutations: Luby-Rackoff revisited." *Proceedings of the* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[16] J. Patarin, "Improved security bounds for pseudorandom permutations." Fourth ACM Conference on Computer and Communications Security, 1997.

[17] J. Patarin, “About Feistel schemes with six (or more) rounds.” To appear in *Fast Software Encryption* (FSE5), March 1998.

[18] J. Pieprzyk, “How to construct pseudorandom permutations from single pseudorandom functions.” *Advances in Cryptology – Eurocrypt* 90 *Proceedings*, Lecture Notes in Computer Science Vol. 473, I. Damgård ed., Springer-Verlag, 1990 pp. 140–150.

[19] C. Shannon, “Communication theory of secrecy systems.” Bell Systems Technical Journal, 28(4), 656–715 (1949).

[20] V. Shoup, “On fast and provably secure message authentication based on universal hashing.” *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[21] M. Wegman and L. Carter, “New hash functions and their use in authentication and set equality.” *J. of Computer and System Sciences* 22, 265–279 (1981).

[22] Y. Zheng, T. Matsumoto and H. Imai, “Impossibility and optimality results on constructing pseudorandom permutations.” *Advances in Cryptology – Crypto* 90 *Proceedings*, Lecture Notes in Computer Science Vol. 537, A. J. Menezes and S. Vanstone ed., Springer-Verlag, 1990, pp. 412-422.