Feistel Networks made Public, and Applications

Yevgeniy Dodis and Prashant Puniya

Department of Computer Science,
Courant Institute of Mathematical Sciences,
New-York University
{dodis,puniya}@cs.nyu.edu

Abstract. Feistel Network, consisting of a repeated application of the Feistel Transform, gives a very convenient and popular method for designing "cryptographically strong" permutations from corresponding "cryptographically strong" functions. Up to now, all usages of the Feistel Network, including the celebrated Luby-Rackoff's result, critically rely on (a) *the (pseudo)randomness of round functions*; and (b) *the secrecy of (at least some of) the intermediate round values* appearing during the Feistel computation. Moreover, a small constant number of Feistel rounds was typically sufficient to guarantee security under assumptions (a) and (b). In this work we consider several natural scenarios where at least one of the above assumptions does not hold, and show that a constant, or even logarithmic number of rounds is *provably insufficient* to handle such applications, implying that a new method of analysis is needed.

On a positive side, we develop a new combinatorial understanding of Feistel networks, which makes them applicable to situations when the round functions are merely *unpredictable* rather than (pseudo)random and/or when the intermediate round values may be leaked to the adversary (either through an attack or because the application *requires* it). In essence, our results show that in any such scenario a super-logarithmic number of Feistel rounds is *necessary and sufficient* to guarantee security.

Of independent interest, our technique yields a novel domain extension method for messages authentication codes and other related primitives, settling a question studied by An and Bellare in CRYPTO 1999.

Keywords: Feistel Network, Verifiable Random Functions/Permutations, PRFs, PRPs, MACs, Domain Extension.

1 Introduction

Feistel Networks are extremely popular tools in designing "cryptographically strong" permutations from corresponding "cryptographically strong" functions. Such networks consist of several iterative applications of a simple Feistel permutation $\Psi_f(x_L \parallel x_R) = x_R \parallel x_L \oplus f(x_R)$, with different (pseudo)independent round functions f used at each round. Among their applications, they are commonly used in the design of popular block ciphers, such as DES, as well as other constructs, such as popular padding schemes OAEP [2] or PSS-R [3]. In particular, the celebrated result of Luby and Rackoff [12] shows that three (resp.

four) rounds of the Feistel transform are sufficient to turn a pseudorandom function (PRF) family into a pseudorandom permutation (PRP) family (resp. strong PRP family). There has been a lot of subsequent work (e.g., [20, 23, 15, 22]) on improving various aspects of the Luby-Rackoff's result (referred to as "LR" from now on). However, all these results crucially relied on:

(a) *the (pseudo)randomness of round functions*; and
(b) *the secrecy of (at least some of) the intermediate round values appearing during the Feistel computation*

In this work we consider several natural scenarios where at least one of the above assumptions does not hold, and show that a fundamentally new analysis technique is needed for such applications. But first let us motivate our study.

IS UNPREDICTABILITY ENOUGH? We start with the assumption regarding pseudorandomness of round functions. This assumption is quite strong, since practical block ciphers certainly do not use PRFs as their round functions. Instead, they heuristically use considerably more than the three-six rounds predicted by the LR and all the subsequent "theoretical justifications". Thus, a large disconnect still remains to be bridged. Clearly, though, we need to assume some security property of the round function, but can a weaker property be enough to guarantee security?

In the context of domain extension of message authentication codes, An and Bellare [1] studied a natural question whether *unpredictability* — a much weaker property than pseudorandomness — can at least guarantee the unpredictability of the resulting Feistel permutation. Although not as strong as pseudorandomness, this will at least guarantee some minimal security of block ciphers (see [7]), is enough for basic message authentication, and anyway doubles the domain of the unpredictable function, which is useful (and non-trivial!) by itself. [1] gave a negative answer for the case of three rounds, and suggested that "even more rounds do not appear to help". This result indicates that previous "LR-type techniques" are insufficient to handle unpredictability (since in the case of PRFs three rounds are enough), and also leaves open the question whether more Feistel rounds will eventually be enough to preserve unpredictability. Our work will completely resolve this question. Along the way, it will prove that Feistel Networks could serve as domain extenders for message authentication codes.

IS IT SAFE TO LEAK INTERMEDIATE RESULTS? Another crucial reason for the validity of the LR result is the fact that all the intermediate round values are never leaked to the attacker. In fact, the *key* to the argument, and most of the subsequent results, is that the attacker effectively gets no information about most of these values in case a PRF is used for the round function, and simple attacks (which we later generalize to many more rounds) are possible to invalidate the LR result in case the intermediate values are leaked. Unfortunately, for many natural applications this assumption (or conclusion!) can not be enforced, and a totally new argument is needed. We give several examples.

Starting with the simplest (but also least interesting) example, intermediate values might be inadvertently leaked through an attack. For example, one might imagine a smartcard implementing a block cipher via the Feistel network using a secure chip implementing a PRF. In this case the attacker might be able to observe the communication between the smartcard and the chip, although it is unable to break the security of the chip. More realistically, when the round functions are not PRFs, the attacker might get a lot of information about the intermediate values anyway, even without extra attack capabilities. For example, in the case of unpredictable functions (UFs) mentioned above, we will construct provably secure UFs such that the output of the Feistel Network completely leaks *all* the intermediate round values. Although artificial, this example illustrates that weaker assumptions on the round functions can no longer guarantee the secrecy of intermediate values.

For yet another example, the round function might simply be public to begin with. This happens when one considers the question of implementing an ideal cipher from a random oracle, considered by the authors in TCC'06 [6]. In this case the round function is a publicly accessible random oracle, and is certainly freely available to the attacker. To see the difference with the usual block cipher setting where four round are enough, [6] showed that even five Feistel rounds are not sufficient to built an ideal cipher, although conjectured that a larger constant number of rounds is sufficient. The authors also showed a weaker positive implication in the so called "honest-but-curious model", although only for a super-logarithmic number of rounds (as they also showed, reducing the number of rounds in this model would imply the security in the usual, "malicious" model). As a final example (not considered in prior work), the attacker might get hold of the intermediate values because the *application requires to reveal such values*. This happens when one tries to add *verifiability* to PRFs and PRPs (or their unpredictable analogs), which we now describe in more detail.

Verifiable Random Functions and Permutations. We consider the problem of constructing *verifiable random permutations* (VRPs) from *verifiable random functions* (VRFs). VRFs and VRPs are verifiable analogs of PRFs and PRPs, respectively. Let us concentrate on VRFs first. Intuitively, regular PRFs have a limitation that one must trust the owner of the secret key that a given PRF value is correctly computed. And even when done so, a party receiving a correct PRF value cannot later convince some other party that the value is indeed correct (i.e., PRF values are "non-transferable"). In fact, since the function values are supposed to be (pseudo)random, it seems that such verifiability of outputs of a PRP would contradict its pseudorandomness. The way out of this contradiction was provided by Micali, Rabin and Vadhan [17], who introduced the notion of a VRF. Unlike PRFs, a VRF owner must be able to provide a short proof that any given VRF output is computed correctly. This implies that the VRF owner must publish a public key allowing others to verify the validity of such proofs. However, every "unopened" VRF value (i.e., one for which no proof was given yet) should still look indistinguishable from random, even if many other values were "opened" (by giving their proofs). Additionally, the public key should commit the owner of the VRF to all its function values in a unique way, even if the owner tries to select an "improper" public key. Micali et al. [17]

also gave a secure construction of a VRF based on the RSA assumption. Since then, several more efficient constructions of VRFs have been proposed based on various cryptographic assumptions; see [13, 5, 8].

The notion of a VRP, which we introduce in this paper, naturally adds verifiability to PRPs, in exactly the same natural way as VRFs do to PRFs. We will describe some applications of VRPs later (and more in [7]), but here let us concentrate on the relation between VRFs and VRPs. On the one hand, it is easy to see that a VRP (on a "non-trivial domain") is also a VRF, just like in the PRF/PRP case. On a first look, we might hope that the converse implication holds as well, by simply applying the Luby-Rackoff result to VRFs in place of PRFs. However, a moment of reflection shows that this is not the case. Indeed, the proof for the iterated Feistel construction *must include all the VRF values for the intermediate rounds*, together with their proofs. Thus, the attacker can legally obtain all the intermediate round values for every input/output that he queries, except for the one on which he is being "challenged". This rules out the LR-type proof for this application. More critically, even the recent proof of [6] (implementing the ideal cipher from a random oracle in the "honest-but-curious" model) appears to be "fundamentally inapplicable" as well. Indeed, that proof crucially used the fact that truly random functions (in fact, random oracles) are used in all the intermediate rounds: for example, to derive various birthday bounds used to argue that certain "undesirable" events are unlikely to happen. One might then hope that a similar argument might be carried out by replacing all the VRFs by truly random function as well. However, such "wishful replacement" is prevented by the fact that we are required to prove the correctness of each intermediate round value, and we (provably) *cannot provide such proofs when we use a totally random function in place of a VRF* (which is "committed" to by its public key). To put it differently, with a random function we have no hope of simulating the VRF proofs that are "legally expected" by an adversary attacking the VRP construction. Thus, again, a new technique is needed.

VERIFIABLE UNPREDICTABLE FUNCTIONS AND PERMUTATIONS. We also consider the natural combination of the scenarios we considered so far, exemplified by the task of constructing *verifiable unpredictable permutations* (VUPs) from *verifiable unpredictable functions* (VUFs) [17] (also called *unique signature schemes* [11, 13]). A VUF is defined in essentially the same way as VRFs, except that the pseudorandomness requirement for VRFs is replaced by a weaker unpredictability requirement. Similarly, VUPs, introduced in this paper, are either the permutation analogs of VUFs, or, alternatively, unpredictable analogs of VRPs. Of course, as a VRP is also a VUP, we could attempt to build a VUP by actually building a VRP via the Feistel construction applied to a VRF, as suggested in the previous paragraph. However, this seems quite wasteful since VUFs appear to be much easier to construct than VRFs. Indeed, although in theory VUFs are equivalent to VRFs [17], the "Goldreich-Levin-type" reduction from VUFs to VRFs in [17] is *extremely* inefficient (it loses exponential security and forces the authors to combine it with another inefficient tree construction). Moreover, several previous papers [17, 13] constructed *efficient* VUFs based on relatively standard *computational* assumptions, while all the *efficient* VRF constructions [5, 8] are based on very ad hoc *decisional* assumptions. Thus, it is natural to study the security of the Feistel network when applied to VUFs. In this case, not only the round functions cannot be assumed pseudorandom, but also all the intermediate values must be leaked together with their proofs of correctness, making this setting the most challenging to analyze.

Other Related Work. Several prior works tried to relax the security of some of the round functions. For example, Naor and Reingold showed that the first and the fourth round could use pairwise independent hash functions instead of PRFs. In a different vein, Maurer et al. [14] studied the case when the PRFs used are only non-adaptively secure. Already in this setting, the authors showed that it is unlikely that four Feistel rounds would yield a PRP (although this is true in the so called "information-theoretic" setting). However, in these results at least some of the round functions are still assumed random. In terms of leaking intermediate results, Reyzin and Ramzan showed that in a four-round construction it is safe to give the attacker *oracle access* to the second and third (but not first and fourth) round functions. This is incomparable to our setting: we leak intermediate results actually happening during the Feistel computation, and for *all* the rounds. Finally, we already mentioned the paper by the authors [6], which showed how to deal with public intermediate results when *truly random* round functions are used. As we argued, however, this technique is insufficient to deal with unpredictability, and cannot even be applied to the case of VRFs (because one cannot simulate the proofs of correctness for a truly random function).

1.1 Our Results

In this work we develop a new understanding of the Feistel Network which allows us to analyze the situations when the intermediate round values may be leaked to the adversary, and also handle cases when the round values are merely unpredictable rather than pseudorandom. In our modeling, a k-round Feistel Network is applied to k members $f_1 \dots f_k$ independently selected from some (not necessarily pseudorandom) function family C, resulting in a Feistel permutation π. Whenever an attacker makes a forward (resp. backward) query to π (resp. π^{-1}), we assume that it learns all the intermediate values (as we mentioned, this is either required by the application, or may anyway happen with unpredictable functions).

Negative Result. As our first result, we show a simple attack allowing an adversary to compute any value $\pi^{-1}(y)$ by making at most exponential in k number of *forward* queries to π. Since such an inversion should be unlikely (with polynomially many queries) even for an unpredictable permutation, this immediately means that at least a superlogarithmic number of Feistel rounds (in the security parameter λ) is *necessary* to guarantee security for *any* of the applications we consider. Aside from showing the *tightness of all our positive results* described below, this result partially explains *why practical block ciphers use significantly more than 3-6 rounds* predicted by all the previous "theoretical justifications" of the Feistel Network. Indeed, since all such ciphers heuristically use round functions which are not PRFs, and we just showed that even unpredictable round functions might leak a lot (or even *all*) of the intermediate results, the simple attack we present might have been quite applicable if a small constant number of rounds was used!

Matching Positive Result. On a positive side, we show a general combinatorial property of the Feistel Network which makes essentially no assumptions (such as pseudorandomness) about the round functions used in the Feistel construction, and allows us to apply it to a wide variety of situations described above, where the previous techniques (including that of [6]) failed. In essence, for any $s \leq k/2$, we show that if an attacker, making a sub-exponential in s number of (forward or backward) queries to the construction and always learning all the intermediate round values, can cause a non-trivial collision somewhere between rounds s and $k-s$, then the attacker can also find a simple (and non-trivial) XOR condition on a constant (up to six) number of the round values of the queries he has made. This means that if a function family C is such that it is provably hard for an efficient attacker to find such a non-trivial XOR condition, — and we call such families *5-XOR resistant* (see Section 4), — then it is very unlikely that the attacker can cause any collisions between rounds s and $k-s$ (as long as s, and thus k, are super-logarithmic in the security parameter λ). And once no such collisions are possible, we show that is possible to directly argue the security of the Feistel Network for our applications. In particular, as even mere unpredictability is enough to establish 5-XOR resistance, we conclude that super-logarithmic number of Feistel rounds is *necessary and sufficient* to yield

– a (strong) unpredictable permutation (UP) from any unpredictable function (UF).
– a strong PRP from any PRF, which remains secure even if all the round values are made public.
– a strong VUP from any VUF.
– a strong VRP from any VRF.

These results are in sharp contrast with the "LR-type" results where a constant number of rounds was sufficient, but also give the first theoretical justification regarding the usage of Feistel Networks not satisfying assumptions (a) or (b) mentioned earlier. For the case of block ciphers, our justification seems to match more closely the number of rounds heuristically used in practical constructions.

Implications to Domain Extension. Since the Feistel Network doubles the length of its input, our results could also be viewed in relation to the question of domain extension of UFs, VUFs and VRFs. In practice, the question of domain extension is typically handled by a collision-resistant hash function (CRHF): it uses only one call the the underlying n-bit primitive f and does not require the secret key to grow. However, the existence of a CRHF is a theoretically strong assumption, which does not seem to follow from the mere existence of UFs, VRFs or VUFs. This is especially true for UFs, whose existence follows from the existence of mere one-way functions and, hence, can even be "blackbox separated" from CRHFs [24]. Thus, it makes sense to consider the question of domain extension *without introducing new assumptions*.

For PRFs, this question is easily solved by using (almost) universal hash functions (instead of CRHFs) to hash the message to n bits before applying the n-bit PRF. However, this technique fails for UFs, VUFs and VRFs: in the case of unpredictability because the output reveals information about the hash key, and for VRFs because it is unclear how to provide proofs of correctness without revealing the hash key. Another attempt (which works for digital signatures) is to use target collision-resistant hash functions [21] in place of CRHFs, but such functions have to be freshly chosen for each new input, which will break the unique provability of UFs, VUFs and VRFs. (Additionally, the hash key should also be authenticated, which further decreases the bandwidth.) In case the underlying n-bit primitive f is shrinking (say, to $n-a$ bits), one can use some variant of the cascade (or Merkle-Damgård) construction. Indeed, this was formally analyzed for MACs by [1, 16]. However, the cost of this method is one evaluation of f per a input bits. In particular, in case the output of f is also equal to n, which is natural if one wants to extend the domain of a UF given by a block cipher, this method is either inapplicable or very inefficient.[1]

In contrast, our method builds a UF/VUF/VRF from $2n$ to $2n$ bits from the one from n to n bits, by using $k = \omega(\log \lambda)$ evaluations of f, albeit also at the price of increasing the secret key by the same amount. This answers the question left open by An and Bellare [1] (who only showed that three rounds are insufficient): *Feistel Network is a good domain extender for MACs if and only if it uses super-logarithmic number of rounds*!

Moreover, in the context of UFs (and VUFs), where one wants to minimize the output length as well, we notice that the output length can be easily reduced from $2n$ to n. This is done by simply dropping the "left half" of the k-round Feistel network output! The justification for this optimization follows by noticing that in this case the attacker will only make forward queries to the Feistel construction. For such attackers, we can extend our main combinatorial lemma as follows. For any $s \leq k$, if a 5-XOR resistant family is used to implement the round functions and the attacker made less than exponential in s number of queries, then the attacker has a negligible chance to cause any collisions between rounds s and k (as opposed to $k-s$ we had when backward queries were allowed). From this, one can derive that $k = \omega(\log \lambda)$ Feistel rounds is enough to turn a UF (or VUF) from n to n bits into one from $2n$ to n bits. Moreover, in the case of UFs we expect that one would use a (possibly heuristic) pseudorandom generator to derive the k round keys (much like in the case of block ciphers), meaning that the only effective cost is k computations of the basic UF. Once the domain is doubled, however, one can use the cascade methods [1, 16] to increase it further without increasing the key or the output length.

[1] In principle, such length-preserving f can be "truncated" by a bits, but this loses an exponential factor in a in terms of exact security. Thus, to double the input length, one would have to evaluate f at least $\Omega(n/\log \lambda)$ times.

Other Applications. In the full version [7], we illustrate several applications of our results. We describe only a couple here due to the space constraints.

As a simple, but illustrative application, we notice that VRPs immediately yield *non-interactive, setup-free, perfectly-binding commitments schemes*. The sender chooses a random key pair (SK, PK) for a VRP π. To commit to m (in the domain of the VRP), the sender sends PK and the value $c = \pi_{SK}(m)$ to the receiver. To open m, the sender sends m and the proof that $c = \pi_{SK}(m)$, which the receiver can check using the public key PK. The hiding property of this construction trivially follows for the security of VRPs. As for binding, it follows from the fact that π is a permutation even for an *adversarial choice of* PK. As we can see, it is not clear how to achieve binding *directly* using plain VRFs. However, given our (non-trivial) equivalence between VRFs and VRPs, we get that VRFs are also sufficient for building non-interactive, perfectly binding commitment schemes without setup. Alternatively, to commit to a single bit b, one can use VUPs augmented with the Goldreich-Levin bit [10]. Here the sender would pick a random r and x, and send PK, r, $\pi_{SK}(x)$, and $(x \cdot r) \oplus b$, where $x \cdot r$ denotes the inner product modulo 2. Using our equivalence between VUPs and VUFs, we see that VUFs are sufficient as well.

We remark that the best general constructions of such commitments schemes was previously based on one-way permutations (using the hardcore bit) [4], since Naor's construction from one-way functions [19] is either interactive, or non-setup-free. Since the assumption of one-way permutations is incompatible with VUFs or VRFs, our new construction is not implied by prior work.

Micali and Rivest [18] suggested the following elegant way to perform *non-interactive lottery* (with the main application in micropayments). The merchant publishes a public key PK for a VRF f, the user chooses a ticket x, and wins if some predicate about $f(x)$ is true (for example, if $f(x)$ is less than some threshold t). Since f looks random to the user, the user cannot significantly bias his odds no matter what x he chooses. Similarly, since the merchant is committed to f by the public key PK, they merchant cannot lie about the value $f(x)$. However even in this case, nothing stops the merchant from publishing a "non-balanced" VRF (meaning choosing a specific f such that $f(x)$ is "far from random" even for *random* x). In the extreme case, a constant function $f(x) = c$, where c is selected so that the predicate does not hold. We need "balancedness" to ensure that the merchant not only cannot change the value of f after the commitment, but also that the user has a fair chance of winning when he chooses a *random* x, no matter which f the merchant selects. VRPs perfectly solve this problem.

Moreover, VRPs have an extra advantage that one can *precisely* know the number of possible winners: it is exactly equal to the number of strings y satisfying the given predicate. Thus, one can always allocate a given number of prizes and never worry that with some small probability there will be more winners than prizes.

We briefly mention some other applications described in [7]. For example, UPs are enough to argue weaker "fall-back" security properties for some applications of block ciphers, which is nice in case the PRP assumption on the block cipher turns out incorrect. VRPs, or sometimes even VUPs, can be useful in several applications where plain VRFs are insufficient. For example, to implement so called "invariant signatures" needed by Goldwasser and Ostrovsky [11] in constructing non-interactive zero-knowledge proofs. Additionally, VRPs could be useful for adding verifiability to some application of PRPs (where, again, PRFs are not sufficient). For example, to construct verifiable CBC encryption or decryption, or to "truthfully", yet efficiently, sample certain verifiable huge (pseudo)random objects [9], such as random constant-degree expanders. Finally, our construction of VRPs from VRFs could lead to a "proof-transferable" implementation of the Ideal Cipher Model using a semi-trusted third party. We refer to [7] for more details, and hope that more applications of our constructs and techniques will be found.

2 Definitions and Preliminaries

Let λ denote the security parameter. We use $negl(\lambda)$ to denote a negligible function of λ. $\mathsf{Fibonacci}(k)$ denotes the k^{th} Fibonacci number, and thus $\mathsf{Fibonacci}(k) = \mathcal{O}(1.618^k)$.

Now we give informal definitions of the various primitives that we use in this paper. For formal definitions, see full version [7]. We start by defining the notion of *pseudorandom functions* (PRFs). We use a slightly non-standard definition of PRFs that is convenient to prove our results. However, this definition is equivalent to the usual definition.

In the new PRF attack game, the attacker A_f runs in three stages: (1) In the experimentation phase, it is allowed to query a PRF sampled from the PRF family. (2) In the challenge phase, it sends an unqueried PRF query and in response the challenger sends either the PRF output or a random output with equal probability. (3) In the analysis phase, the attacker again gets oracle access to the PRF, but cannot query it on the challenge query. At the end of the attack, A_f has to guess if the challenge response was random or pseudorandom. The attacker A_f wins if it guesses correctly. Similar to the notion of PRFs, we can define the notion of *(strong) pseudorandom permutations* (PRPs). Here the attacker has oracle access to both the forward as well as inverse PRP, but the attack game is otherwise similar to that for PRFs.

A slightly weaker notion than PRFs is that of *Unpredictable Functions* (UFs). Unpredictable functions are also popularly known as (deterministic) *Message Authentication Codes* (MACs). In this case, the UF attacker is allowed to query an unpredictable function from the UF family, and it needs to predict the output of the UF on an unqueried input at the end of the interaction. The advantage of the UF adversary is the maximum probability with which it predicts correctly. In an analogous fashion, we can also define the notion of *Unpredictable Permutations* (UPs), where the attacker has oracle access to both the forward and inverse permutation and has to predict an unqueried input/output pair.

We can define verifiable analogs of each of the above primitives. Thus, we get *verifiable random functions*, *verifiable random permutations*, *verifiable un-*

predictable functions and *verifiable unpredictable permutations*. In each case, the primitive takes a public/private key pair, and consists of three algorithms (Gen, Prove, Verify). The Gen algorithm outputs a public/private key pair. The Prove algorithm allows the private key owner to compute the function/permutation output as well as give a proof of correctness. Finally, the Verify algorithm allows anyone who knows the public key to verify the correctness of an input/output pair by observing the corresponding proof.

Each of these primitives satisfies two properties: (1) *Correctness*, i.e. one can verify correct input/output pairs, and (2) *Soundness*, i.e. one cannot prove two distinct outputs for the same input, even for an *adversarially chosen public key*. Additionally, these primitives satisfy the natural analogs of the pseudorandomness/unpredictability definition of the corresponding non-verifiable primitive (except the attacker also gets the proofs for all the values except for the challenge).

The *Feistel transformation* using $f : \{0,1\}^n \to \{0,1\}^n$ is a permutation Ψ_f on $2n$ bits defined as, $\Psi_f(x) \stackrel{def}{=} x_R \parallel x_L \oplus f(x_R)$. The symbols x_L and x_R denote the left and right halves of $2n$ bit string x. We will often call the construction based on k iterated applications of the Feistel transformation, a k-round LR construction, and denote it by $\Psi_{f_1 \dots f_k}$ (or Ψ_k when $f_1 \dots f_k$ are clear from context) where $f_1 \dots f_k$ are the *round functions* used. If the input to Ψ_k is $x = R_0 || R_1$, for $R_0, R_1 \in \{0,1\}^n$, then the k-round LR construction Ψ_k generates k more n-bit values $R_2 \dots R_{k+1}$ (one after each application of a round function, i.e. $R_i = f_{i-1}(R_{i-1}) \oplus R_{i-2}$ for $i = 2 \dots (k+1)$). We will refer to the n-bit values $R_0, R_1 \dots R_k, R_{k+1}$ as the *round values* of the LR construction.

3 Insecurity of $\mathcal{O}(\log \lambda)$-round Feistel

We will demonstrate here that upto a logarithmic number of Feistel rounds do not suffice for any of our results. In order to make our proof precise, we show a simple adversary that is able to find the input corresponding to any permutation output $y \in \{0,1\}^{2n}$ by making polynomially many *forward* queries and observing the intermediate round values.

Theorem 1. *For the k round Feistel construction Ψ_k that uses $k = \mathcal{O}(\log \lambda)$ round functions, there exists a probabilistic polynomial time adversary A_π that takes oracle access to Ψ_k (while also gets access to the* intermediate round values *of Ψ_k). The adversary A_π makes $\mathcal{O}(\mathsf{Fibonacci}(k)) = poly(\lambda)$ forward queries to Ψ_k and with high probability finds the input corresponding to an output y without actually making that query.*

Proof: The adversary A_π starts by choosing a permutation output y, that it will try to invert Ψ_k on. For concreteness, we assume that $y = 0^{2n}$ (anything else works just as well). We will describe the recursive subroutine that the attacker A_π is based on. Say the round functions of Ψ_k are $f_1 \dots f_k$. The recursive function that we describe is $E(j, Y)$, where j is the number of rounds in the Feistel construction and Y is a $2n$ bit value, and the task of $E(j, Y)$ is to find the input such that the j^{th} and $(j+1)^{th}$ round values are Y_L and Y_R (the left and right halves of Y), respectively.

- $\mathbf{E(1, Y)}$: Choose a random $R'_0 \leftarrow \{0,1\}^n$. Make the forward query $R'_0 \parallel Y_L$ to Ψ_1, where the 2^{nd} round value is R'_2. Now the 1^{st} and 2^{nd} round values for the input $R'_2 \oplus R'_0 \oplus Y_R \parallel Y_L$ are Y_L and Y_R.
- $\mathbf{E(j, Y)}$, $\mathbf{j > 1}$: Perform the following steps,
  - Make a random query $R_0 \parallel R_1 \leftarrow \{0,1\}^{2n}$, and say the $2n$ bit value at the j^{th} round is is $R_j \parallel R_{j+1}$. Then, $f_j(R_j) = (R_{j-1} \oplus R_{j+1})$.
  - Run $E(j-2, (f_{j-1}(R_{j-1}) \oplus Y_L) \parallel R_{j-1})$ and the $2n$ bit value at the $(j-1)^{th}$ round is $R_{j-1} \parallel Y_L$. Hence $f_j(Y_L) = R_{j-1} \oplus R_{j+1}$.
  - Run $E((j-1), (f_j(Y_L) \oplus Y_R) \parallel Y_L)$, and the j^{th} and $(j+1)^{th}$ round values are Y_L and Y_R, respectively.

The adversary A_π essentially runs the algorithm $E(k, 0^{2n})$. Now we need to make sure that the adversary A_π does not query on the input corresponding to the output 0^{2n}. But since all the queries made in the recursive algorithm are essentially chosen at random, we know that the probability of this happening is $\frac{q}{2^{2n}}$. Hence, the probability that A_π succeeds is at least $\left(1 - \frac{q}{2^{2n}}\right)$. ⊓⊔

We note that the above attacker works in a scenario where it can only make forward queries to the Feistel construction Ψ_k. In case it can make inverse queries as well, it is possible to design a similar attacker that succeeds in $\mathcal{O}(\mathsf{Fibonacci}(k/2))$ queries. If the number of rounds $k = \mathcal{O}(\log \lambda)$, then the number of queries needed by either of these attackers is polynomial in the security parameter λ.

It is easy to see how such an attacker can be utilized in three of the four scenarios, if we use the Feistel construction for each of these cases.

- *PRP construction with public round values*: By definition, for a PRP we should not be able to invert an output without actually querying the construction on it.
- *VRP (VUP) construction using VRFs (VUFs)*: In order to provide the proofs for the VRP (VUP), the VRP (VUP) construction will need to reveal all intermediate VRF (VUF) inputs/outputs and the corresponding proofs.

On the first look, it seems that when we use a Feistel construction with *unpredictable functions* in each round to construct an *unpredictable permutation* (UP), the UP adversary cannot make use of the above attacker since it does not have access to all the intermediate round values. However, we will show that if certain pathological (but secure) unpredictable functions are used as round functions, then the UP adversary can infer *all* the round values simply by observing the output of the Feistel construction!

Lemma 1. For any $k \leq \frac{n}{\omega(\log \lambda)}$ (in particular, if $k = \mathcal{O}(\log \lambda)$), there exist k secure unpredictable functions $f_1 \dots f_k$, such that by querying the k-round Feistel construction $\Psi_{f_1 \dots f_k}$ on any input, an attacker can always efficiently learn all the intermediate round values (even when it does not have access to the intermediate round values).

Proof: Let $\{g_i : \{0,1\}^n \rightarrow \{0,1\}^{n/k}\}_{i\in\{1\ldots k\}}$ be k secure unpredictable functions. For $i \in \{1,k\}$, we will define the functions $f_i : \{0,1\}^n \rightarrow \{0,1\}^n$ as $f_i(x) = 0^{(i-2)\cdot(n/k)} \parallel x_{i-1} \parallel g_i(x) \parallel 0^{(k-i)\cdot(n/k)}$, where x_{i-1} denotes the $(i-1)^{th}$ (n/k) bit block in the input x. Each of the functions f_i is a secure unpredictable function if the corresponding function g_i is a secure UF.

Consider a query $(R_0 \parallel R_1) \in \{0,1\}^{2n}$ made to the Feistel construction $\Psi_{f_1\ldots f_k}$. Now we will consider both R_0 and R_1 as consisting of k blocks of length (n/k) each, which we will denote by $R_0 = R_0^1 \parallel \ldots \parallel R_0^k$ and $R_1 = R_1^1 \parallel \ldots \parallel R_1^k$. Denote the round values generated in computing the output of this construction as $(R_0, R_1)\ldots(R_k, R_{k+1})$, where $R_k \parallel R_{k+1}$ is the output of this construction. If the number of rounds k in the Feistel construction is even, then we note that the output of the construction is:

$$\begin{aligned} R_k &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_{k-1}(R_{k-1}) \oplus R_0^{k-1}) \parallel R_0^k \\ R_{k+1} &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_k(R_k) \oplus R_1^k) \end{aligned}$$

If number of rounds k is odd, then the output of the Feistel construction is,

$$\begin{aligned} R_k &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_{k-1}(R_{k-1}) \oplus R_1^{k-1}) \parallel R_1^k \\ R_{k+1} &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_k(R_k) \oplus R_0^k) \end{aligned}$$

Now it is easy to find each of the round function outputs (and hence the intermediate round values) by simply observing the right half of the output of the Feistel construction. ⊔⊓

Thus, we see that if the number of rounds in the Feistel construction (using UFs) used to construct *unpredictable permutations* is $k = \mathcal{O}(\log\lambda)$, then the resulting construction is insecure (since all the intermediate round values may be visible and we can apply theorem 1). Even if we attempt to shrink the output length of this MAC construction by chopping the left half of the output, it would be possible to retrieve all intermediate round values by simply observing the MAC output. In fact, even for $k = \omega(\log\lambda)$ (but less than $n/\omega(\log\lambda)$) rounds it might be possible to retrieve all intermediate round values, and hence a new proof technique is needed.

4 A Combinatorial Property of the Feistel Construction

In this section, we will prove a general combinatorial lemma about the k round LR-construction Ψ_k, that uses arbitrary round functions $f_1 \ldots f_k$. We will see in the following section that this lemma is crucial in deriving each of our results using the Feistel construction.

Consider an arbitrary ordered sequence of q forward/inverse permutation queries made to the construction Ψ_k, each of which is a $2n$ bit string. Denote the $(k+2)$ n-bit round values associated with the i^{th} query as $R_0^i, R_1^i \ldots R_k^i, R_{k+1}^i$, where $R_0^i \parallel R_1^i$ (resp. $R_k^i \parallel R_{k+1}^i$) is the input if this is a forward (resp. inverse) query. We say that such a sequence of queries produces an s^{th} round value collision, if the s^{th} round value collides for two different permutation queries from this query sequence. That is, we have that $R_s^i = R_s^j$ for $i, j \in \{1 \dots q\}$ and $R_0^i \parallel R_1^i \neq R_0^j \parallel R_1^j$.

We essentially show that if any such sequence of q queries produces a r^{th} round value collision for any $r \in \{s \dots (k-s)\}$ (where $s \leq (k/2)$), then one of the following must hold:

1. The number of queries q is exponential in s.
2. For this sequence of queries, there is at least one new round function evaluation such that the new round value generated can be represented as a bit-by-bit XOR of upto 5 previously existing round values.

We refer to the second condition above as the 5-*XOR condition*. We label the queries in the order they are made, i.e. query i is made before query $i+1$ for $i = 1 \dots q-1$. By a "new round function evaluation", we mean when a round function is evaluated on an input (i.e. the corresponding round value) to which it was not applied in an earlier query. If the i^{th} query is a forward (inverse) query and the round function evaluation $f_j(R_j^i)$ is a new one, then the new round value generated as a result is R_{j+1}^i (resp. R_{j-1}^i). The 5-*XOR condition* essentially states that for at least one such new round function evaluation, the new round value generated can be represented as the bit-by-bit XOR of upto 5 previously existing round values. Here previously existing round values include round values from previous queries and round values in the same query that were generated earlier (depending on whether this is a forward/inverse query). Our combinatorial result is formalized in the main lemma below (where, for future convenience, we denote R_j^i by $R[i, j]$).

Lemma 2. Let Ψ_k be a k round LR construction that uses fixed and arbitrary round functions $f_1 \dots f_k$. For any $s \leq \frac{k}{2}$, and any ordered sequence of $q = o(1.3803^{\frac{s}{2}})$ forward/inverse queries, with associated round values $R[i, 0], \dots, R[i, k+1]$ for $i = 1 \dots q$, if the 5-XOR condition does not hold for this sequence of queries then there is no r^{th} round value collision for these queries, for all $r \in \{s \dots (k-s)\}$.

Note that lemma 2 simply states a structural property of the k-round LR construction that holds irrespective of the round functions used in the construction. The proof of this lemma is quite technical and we omit it here due to space constraints (see [7]).

Next, we state a more restricted version of the combinatorial lemma, when the adversary only makes forward queries to the Feistel construction. This lemma (whose proof can also be found in [7]) will be useful when we attempt *domain extension of MACs* in the next section. We give an intuition of the proof of this lemma, which is quite similar to the proof of Lemma 2 (though slightly simpler).

Lemma 3. Let Ψ_k be a k-round LR construction that uses fixed and arbitrary round functions $f_1 \dots f_k$. For any round number s, and any ordered sequence of $q = o(1.3803^{\frac{s}{2}})$ forward queries, with associated round values $R[i, 0], \dots, R[i, k+1]$ for $i = 1 \dots q$, if the 5-XOR condition does not hold for this sequence of forward queries then there is no r^{th} round value collision for these queries, for all $r \geq s$.

Proof Intuition: Consider a sequence of q queries for which the r^{th} round values of two queries collide, while the 5-XOR condition does not hold. Without loss of generality, we can assume that one of queries involved in the r^{th} round value collision is the last one (i.e. the q^{th} query) [2]. We will label the queries $1 \cdots q$, in the order in which they were made. Thus for the round value $R[i,j]$, all the round values $R[i',j']$, with $(i' < i)$ or $(i' = i) \wedge (j' < j)$, were generated before $R[i,j]$. We denote by $\mathsf{p}(i,j)$, the least query number such that $R[\mathsf{p}(i,j),j] = R[i,j]$.

Our main argument consists of four steps which all rely on the fact that the 5-XOR condition does not hold. We start by showing that if the round value $R[q,r]$ collides with the r^{th} round value in an earlier query, then all the round values $R[q,1] \ldots R[q,(r-1)]$ collide with corresponding round values in earlier queries as well. That is,

$$(\mathsf{p}(q,r) < q) \Rightarrow (\mathsf{p}(q,1) < q) \wedge \ldots \wedge (\mathsf{p}(q,(r-1)),(r-1))$$

In order to see this, consider the round value $R[q,(r-1)]$. We know that $(f_{r-1}(R[q,(r-1)]) = R[q,(r-2)] \oplus R[q,r])$. Now since both the round values $R[q,(r-2)]$ and $R[q,r]$ were generated before $R[q,(r-1)]$ (the former because this is a forward query and the latter because $\mathsf{p}(q,r) < q$), it must be the case that $\mathsf{p}(q,(r-1)) < q$ since otherwise the 5-XOR condition will be satisfied. Now we can apply the same argument to the round values $R[q,(r-2)]$ down to $R[q,1]$ to get the desired result. Moreover, we can also show that these queries $\mathsf{p}(q,1) \ldots \mathsf{p}(q,r)$ could only have been made in certain restricted orders. In particular, we show that there is a $j \in \{1 \ldots r\}$ such that

$$\mathsf{p}(q,1) > \ldots > \mathsf{p}(q,j) < \ldots < \mathsf{p}(q,r)$$

In order to see this consider any three consecutive round values $R[q,(i-1)]$, $R[q,i]$ and $R[q,(i+1)]$, corresponding to queries $\mathsf{p}(q,(i-1)), \mathsf{p}(q,i)$ and $\mathsf{p}(q,(i+1))$. We know that $f_i(R[\mathsf{p}(q,i),i]) = R[\mathsf{p}(q,(i-1)),(i-1)] \oplus R[\mathsf{p}(q,(i+1)),(i+1)]$. If it were the case that $\mathsf{p}(q,(i-1)) < \mathsf{p}(q,i)$ and $\mathsf{p}(q,(i+1)) < \mathsf{p}(q,i)$, then this would imply a 5-XOR condition. The only orders that do not have such a query triple are the ones specified above.

Now we can deduce that at least one of these two strictly descending/ascending query sequence, i.e. $\mathsf{p}(q,1) > \ldots > \mathsf{p}(q,j)$ or $\mathsf{p}(q,j) < \ldots < \mathsf{p}(q,r)$, consists of at least $(r/2)$ queries. Without loss of generality, as the longer sequence of queries is $\mathsf{p}(q,1) > \ldots > \mathsf{p}(q,j)$. We consider any of the queries $\mathsf{p}(q,\ell)$ for $\ell \in \{1 \ldots (j-2)\}$, and show that each of the round values $R[\mathsf{p}(q,\ell),1] \ldots R[\mathsf{p}(q,\ell),(\ell-1)]$ collide with the corresponding round value in an earlier query. The first step of this argument, i.e. showing that $R[\mathsf{p}(q,\ell),(\ell-1)]$ collides with the corresponding round value in an earlier query, is the tricky step in this part, beyond which the argument is similar to the first step. Thus, we show that

$$(\mathsf{p}(\mathsf{p}(q,\ell),1) < \mathsf{p}(q,\ell)) \wedge \ldots \wedge (\mathsf{p}(\mathsf{p}(q,\ell),(\ell-1)) < \mathsf{p}(q,\ell))$$

[2] In addition, we assume that the query sequence does not consist of any duplicate queries.

Next, we show that the queries $\mathsf{p}(\mathsf{p}(q,\ell),1)\ldots\mathsf{p}(\mathsf{p}(q,\ell),(\ell-1))$ occur only in a *strictly descending* order. Additionally, we also show that the first $(\ell-2)$ of these queries were made strictly in between the queries $\mathsf{p}(q,(\ell+1))$ and $\mathsf{p}(q,\ell)$. That is, we show that

$$\mathsf{p}(q,(\ell+1)) < \mathsf{p}(\mathsf{p}(q,\ell),(\ell-2)) < \ldots < \mathsf{p}(\mathsf{p}(q,\ell),1) < \mathsf{p}(q,\ell)$$

Note that this is the really crucial step of the argument since we have essentially shown that each of the queries $\mathsf{p}(\mathsf{p}(q,\ell),1)\ldots\mathsf{p}(\mathsf{p}(q,\ell),(\ell-2))$ is distinct from any of the queries $\mathsf{p}(q,1)\ldots\mathsf{p}(q,j)$ (since they occur strictly in between two consecutive queries in the latter sequence). In addition, we are also able to prove that these queries are in strict descending order (unlike the queries $\mathsf{p}(q,1)\ldots\mathsf{p}(q,r)$).

We notice that the above technique can again be applied to the strictly descending sequence of queries, $\mathsf{p}(\mathsf{p}(q,\ell),1)\ldots\mathsf{p}(\mathsf{p}(q,\ell),(\ell-2))$. In this manner, we can continue this argument recursively and derive a recurrence equation to count the number of queries whose existence we prove (which can all shown to be different using the technique above) as follows:

$$q \geq \mathcal{Q}(r/2), \text{ where } \mathcal{Q}(i) = i + \sum_{\ell=2}^{i-2} \mathcal{Q}(\ell-2)$$

Upon solving this recurrence, we get that $q = \omega(1.3803^{r/2})$. ⊒

In our applications, we will be interested in using the LR construction with round functions that resist the 5-XOR condition, when any adaptive adversary makes a polynomial number of queries to the construction while having access to all the intermediate round values. We will specify this as a property of families of functions from which the round functions are independently derived. Hence, let us begin by describing a *function family.* A *function family* C is a set of functions along with a distribution defined on this set. For such a family, $f \leftarrow C$ denotes sampling a function according to the distribution specified by C. A function family is called a *5-XOR resistant function family* if the LR construction using independently sampled functions from this family resists the 5-XOR condition when queried a polynomial number of times by any adaptive adversary.

Definition 1 (5-XOR resistant function family). *A function family $C_{(k,n)}$, that consists of length preserving functions on n bits, is a 5-XOR resistant function family if for any adversary A,*

$$\Pr\left[\begin{array}{c} A \text{ 5-XOR condition} \\ \text{holds in } (A \longleftrightarrow \Psi_{f_1\ldots f_k}) \end{array} \middle|\ f_1\ldots f_k \longleftarrow C_{(k,n)}\right] \leq \epsilon_{xor} = negl(\lambda)$$

Here the advantage ϵ_{xor} of the adversary A depends on the running time of A and the security parameter λ. The running time of A, the input length n and number of Feistel rounds k are all polynomial functions of λ.

By applying Lemma 2 to a LR construction using round functions independently sampled from a 5-XOR resistant function family, we can derive the following corollary.

Corollary 1. *Let Ψ_k be a k-round LR construction that uses round functions that are independently sampled from a 5-XOR resistant function family consisting of functions on n bits. For any adversary A that adaptively makes permutation queries to Ψ_k, while observing the intermediate round values, it holds that*

- *if A makes both forward/inverse queries, then for any round number $s \leq (k/2)$ with $s = \omega(\log \lambda)$,*

$$\Pr\left[\begin{array}{c}\exists\ r^{th} \text{ round value collision during } A \leftrightarrow \Psi_k \\ \text{for some } r \in \{s \ldots (k-s)\}\end{array}\right] \leq \epsilon_{xor}$$

- *if A makes only forward queries, then for any round number $s = \omega(\log \lambda)$,*

$$\Pr\left[\begin{array}{c}\exists\ r^{th} \text{ round value collision during } A \leftrightarrow \Psi_k \\ \text{for some } r \in \{s \ldots k\}\end{array}\right] \leq \epsilon_{xor}$$

Here the bound ϵ_{xor} denotes the maximum advantage of the XOR finding adversary that runs in time $\mathcal{O}(t_A + (q_A k)^5)$, where t_A is the running time of the adversary A and q_A denotes the number of queries made by it. Also, t_A, q_A and the input length n are all polynomial in λ.

This corollary is easily proved since the 5-XOR finding adversary simply runs the collision finding adversary, and performs a brute-force search for a 5-XOR condition when it finds a round value collision. From Lemma 2, such a 5-XOR condition is guaranteed to exist. In fact, we will make use of this corollary in each of the results that we present in the next section, since each of these function families will turn out to be 5-XOR resistant (the proof of this result can also be found in [7]; here we just state the result, although briefly sketching the case of UFs inside the proof of Theorem 3).

Theorem 2. *For each of the primitives: (1)* unpredictable functions*, (2)* pseudorandom functions*, (2)* verifiable unpredictable functions*, and (4)* verifiable random functions*; a function family that yields an independent random sample of the appropriate primitive is a 5-XOR resistant function family.*

5 Implications

All the cryptographic applications of the Feistel construction until recently have relied on all or some of the round functions not being visible to the adversary. In the previous section, we proved a combinatorial property of the Feistel construction where the internal round function values were visible to the adversary. Now we will describe how this property can be applied to a variety of scenarios to yield new or improved cryptographic constructions than before.

We get the following constructions using this new technique: (1) secure construction of *unpredictable permutations* from *unpredictable functions*, (2) more resilient construction of *pseudorandom permutations* from *pseudorandom functions*, (3) construction of *verifiable unpredictable permutations* from *verifiable unpredictable functions*, and (4) construction of *verifiable random permutations* from *verifiable random functions*.

In each case, the proof consists of three parts: (1) showing that the function family under consideration is a 5-XOR function family (see Theorem 2); (2) using Corollary 1 to show that the corresponding permutation construction is unlikely to have collisions at "advanced" rounds; and (3) show that the lack of such collisions implies that the construction is secure. All the proofs are given in [7].

5.1 Unpredictable Permutations and more resilient PRPs

As a first implication of our combinatorial result, we can see that an $\omega(\log(\lambda))$-round LR construction with independent PRFs in each round gives a more resilient construction of PRPs that remain secure even if the intermediate round values are visible to the attacker. We defer further details of this application to the full version [7].

We saw in Section 3 that observing the output of a $k = n/\omega(\log \lambda)$ round Feistel construction with unpredictable round functions may leak all the intermediate round values. Even for realistic UFs, some partial information about the intermediate round values may be leaked through the output. As we discussed earlier, in such a case none of the previous proof techniques are applicable. We will prove a much stronger result here, by showing that if we use a super-logarithmic number of rounds in the Feistel construction, then the resulting UP construction is secure even if the adversary gets all the intermediate round values along with the permutation output.

The UP construction $\Psi_{U,k}$ that we propose consists of $k = \omega(\log \lambda)$ rounds of the Feistel construction using independent *unpredictable functions* $f_1 \dots f_k \leftarrow F$. The following theorem essentially states that this construction is a secure UP construction. Due to space constraints, we omit the formal proof of this theorem (see [7]) and give a short proof intuition here.

Theorem 3. *Given a UP adversary A_π (with advantage ϵ_π) in the unpredictability game against the UP construction $\Psi_{U,k}$ (using round functions from UF family $\mathcal{F}$), one can build a UF adversary A_f that has comparable advantage (to ϵ_π) in the UF attack game against a UF sampled from $\mathcal{F}$. Quantitatively, we show $\epsilon_\pi = \mathcal{O}\left(\epsilon_f \cdot (qk)^6\right)$, where ϵ_f denotes the maximum advantage of a UF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a UF sampled from $\mathcal{F}$. Here t, q are the running time and the number of queries made by A_π, respectively.*

Proof Intuition: Consider the UP adversary A_π that has advantage ϵ_π against the construction $\Psi_{U,k}$, based on the k-round LR construction with independently sampled UFs from the family F in each round. We can consider two cases.

Case 1: the sequence of queries made by A_π satisfies the 5-XOR condition with probability at least $\epsilon_\pi/2$. However, this means that our UF family in not 5-XOR resistant, contradicting Theorem 2. To get the exact security bound (alternatively, to sketch the proof Theorem 2 for the case of UFs), we can construct an attacker A_f for the UF as follows. A_f proceeds by plugging in its challenge UF randomly as any one of the round functions $f_1 \dots f_k$, say f_i. It then chooses at random a query made by A_π as the query in which it will try to predict the output of f_i. It honestly computes the LR construction for this query until the round function f_i, getting round value R_i. At this point, it chooses a random XOR representation from all round values that already exist and outputs this as its prediction for $f_i(R_i)$. Since the UP attacker A_π forces a 5-XOR condition with non-negligible probability, then A_f also succeeds with non-negligible probability (precisely, the advantage ϵ_f of A_f is $\Omega(\epsilon_\pi/(qk)^6)$).

Case 2: alternatively, A_π wins the UP attack game with advantage at least $\epsilon_\pi/2$ without its queries satisfying the 5-XOR condition. In this case we construct A_f as follows. It will attempt to predict a fresh UF value for the middle round function $f_{k/2}$ (and will choose the remaining functions by itself). It will simulate A_π in the obvious manner, using its oracle to find out the middle values $f_{k/2}$. When the adversary A_π outputs a prediction (X, Y) for some fresh UP input/output, A_f computes the LR construction "forward" honestly to get $R_{k/2}$ from X, and "backward" honestly to get $R_{k/2+1}$ from Y. It then outputs $R_{k/2-1} \oplus R_{k/2+1}$ as its prediction for $f_{k/2}(R_{k/2})$, winning if A_π won *and the value* $R_{k/2}$ *is "fresh"*. Thus, A_f could only fail if it already made the query $f_{k/2}(R_{k/2})$ in order to respond to one of the queries of A_π. However, this would imply a $(k/2)^{th}$ round collision, and we can use the combinatorial Lemma 2 to show that the 5-XOR condition must have been true, contradicting our assumption on A_π that the 5-XOR condition was false. ⊒

Domain Extension of MACs. The above result can also be viewed as a construction of MACs from $2n$ to $2n$ bits using MACs from n to n bits. We observe that it is possible to reduce the output length in the above construction to n by simply dropping the left half of the output. Using this technique, we get a MAC construction from $2n$ to n bits. To briefly justify it, in the usual MAC attack game the attacker can only make forward queries. From corollary 1, we get that for any $s = \omega(\log \lambda)$ no efficient attacker can cause a collision on any round value $r \in \{s \dots k\}$ with non-negligible probability. Thus, a proof of security for this MAC will proceed by plugging in the target n- to n-bit MAC in the last round function of the Feistel construction, and arguing that the attacker predicting the $2n$- to n-bit constructed MAC must also forge this last-round n- to n-bit MAC. This is done using a similar proof technique to that for Theorem 3 (albeit using second part of Corollary 1 to argue that no collision occurs at the last round).

More Resilient PRPs. Similarly to the above, we show that $\omega(\log \lambda)$ Feistel rounds yeilds a construction of PRPs from PRFs that remains secure even if the PRF input/output pairs used in the intermediate rounds are visible to an attacker. We denote the corresponding k-round construction by $\Psi_{R,k}$, and show the following quantitative result in [7].

Theorem 4. *Given a PRP adversary A_π with advantage ϵ_π in the "extended PRP" attack game against $\Psi_{R,k}$ (using round functions from PRF family $\mathcal{F}$), one* *can build a PRF adversary A_f having comparable advantage (to ϵ_π in the PRF attack game against a PRF sampled from the PRF family $\mathcal{F}$. Quantitatively, we show that $\epsilon_\pi = \mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)$, where ϵ_f denotes the maximum advantage of a PRF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a PRF sampled from $\mathcal{F}$. Here t, q are the running time and number of queries made by A_π, respectively.*

5.2 Verifiable Unpredictable/Pseudorandom Permutations

The VRP and VUP constructions that we propose are essentially the same, except that we use VRFs as round function in one case and VUFs in the other. Our VRP (resp. VUP) construction $\Psi_{VR,k}$ (resp. $\Psi_{VU,k}$) uses a k-round Feistel construction using independent VRFs (resp. VUFs) $f_1 \dots f_k \leftarrow \mathcal{F}$ as round functions. The public/private keys of $\Psi_{VR,k}$ (resp. $\Psi_{VU,k}$) are simply the concatenation of the public/private keys of the k VRFs (resp. VUFs). The Prove functionality for $\Psi_{VR,k}$ (resp. $\Psi_{VU,k}$) simply gives the permutation output, and gives all intermediate round values along with the VRF (resp. VUF) proofs as its proof. The Verify functionality simply checks if all intermediate VRF (resp. VUF) proofs verify correctly.

We then prove the three properties of the VRP (resp. VUP) construction: *Completeness*, *Soundness* (or unique proofs) and *Pseudorandomness* (resp. *Unpredictability*). The Completeness and Soundness properties in each case are a direct consequence of the corresponding VRF (resp. VUF) properties. Here the *Pseudorandomness* (resp. *Unpredictability*) property are proven in much the same way as Theorem 4 (resp. Theorem 3); see [7] for formal proofs.

Theorem 5. *Given a VRP (resp. VUP) adversary A_π with advantage ϵ_π in the pseudorandomness (resp. unpredictability) game against the VRP (resp. VUP) construction $\Psi_{VR,k}$ (resp. $\Psi_{VU,k}$) using VRFs (resp. VUFs) sampled from the VRF (resp. VUF) family $\mathcal{F}$ as round functions, one can build a VRF (resp. VUF) adversary A_f that has comparable advantage (to ϵ_π) in the pseudorandomness (resp. unpredictability) game against a VRF (resp. VUF) sampled from $\mathcal{F}$. Quantitatively, we show $\epsilon_\pi = \mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)$ (resp. $\mathcal{O}(q^6k^7 \cdot \epsilon_f)$), where ϵ_f denotes the maximum advantage of a VRF (resp. VUF) adversary running in time $\mathcal{O}(t + (qk)^5)$ (resp. $\mathcal{O}(t + (qk)^5)$) against a VRF (resp. VUF) sampled from $\mathcal{F}$. Here t, q are the running time and number of queries made by A_π, respectively.*

Acknowledgments: We would like to thank Rafail Ostrovsky and Shabsi Walfish for several helpful discussions.

References

1. Jee Hea An and Mihir Bellare, *Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions*, CRYPTO 1999: 252-269.
2. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92–111.

3. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
4. Manuel Blum, *Coin Flipping by Telephone - A Protocol for Solving Impossible Problems*, COMPCON 1982: 133-137.
5. Y. Dodis, *Efficient construction of (distributed) verifiable random functions*, In *Proceedings of 6th International Workshop on Theory and Practice in Public Key Cryptography*, pp 1 -17, 2003.
6. Y. Dodis and P. Puniya, *On the relation between Ideal Cipher and Random Oracle Models*, In *Theory of Cryptography Conference 2006.*
7. Y. Dodis and P. Puniya, *Feistel Networks made Public, and Applications*, Full Version, available from IACR EPrint Archive.
8. Y. Dodis and A. Yampolskiy, *A Verifiable Random Function With Short Proofs and Keys*, In *Workshop on Public Key Cryptography (PKC)*, January 2005.
9. Oded Goldreich, Shafi Goldwasser and Asaf Nussboim, *On the Implementation of Huge Random Objects*, FOCS 2003: 68-79.
10. Oded Goldreich and Leonid A. Levin, *A Hard-Core Predicate for all One-Way Functions*, STOC 1989: 25-32.
11. Shafi Goldwasser and Rafail Ostrovsky, *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent (Extended Abstract)*, in *CRYPTO 1992*: 228-245.
12. M. Luby and C. Rackoff, *How to construct pseudo-random permutations from pseudo-random functions*, in *SIAM Journal on Computing*, Vol. 17, No. 2, April 1988.
13. A. Lysyanskaya, *Unique Signatures and verifiable random functions from DH-DDH assumption*, in *Proceedings of the 22nd Annual International Conference on Advances in Cryptography (CRYPTO)*, pp. 597 612, 2002.
14. Ueli M. Maurer, Yvonne Anne Oswald, Krzysztof Pietrzak and Johan Sj?din, *Luby-Rackoff Ciphers from Weak Round Functions?*, EUROCRYPT 2006: 391-408.
15. Ueli M. Maurer and Krzysztof Pietrzak, *The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations*, in *EUROCRYPT 2003*, 544-561.
16. Ueli M. Maurer and Johan Sj?din, *Single-Key AIL-MACs from Any FIL-MAC*, ICALP 2005: 472-484.
17. S. Micali, M. Rabin and S. Vadhan, *Verifiable Random functions*, In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pp. 120 -130, 1999.
18. Silvio Micali and Ronald L. Rivest, *Micropayments Revisited*, CT-RSA 2002, 149-163.
19. Moni Naor, *Bit Commitment Using Pseudo-Randomness*, CRYPTO 1989: 128-136.
20. Moni Naor and Omer Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, in *Journal of Cryptology*, vol 12, 1999, pp. 29-66.
21. Moni Naor and Moti Yung, *Universal One-Way Hash Functions and their Cryptographic Applications*, STOC 1989: 33-43.
22. Jacques Patarin, *Security of Random Feistel Schemes with 5 or More Rounds*, in *CRYPTO 2004*, 106-122.
23. Z. Ramzan and L. Reyzin, *On the Round Security of Symmetric-Key Cryptographic Primitives*, in Advances in Cryptography - Crypto, LNCS vol. 1880, Springer-Verlag, 2000.
24. Daniel R. Simon, *Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?*, EUROCRYPT 1998: 334-345.