Counterexamples to Hardness Amplification Beyond Negligible

Yevgeniy Dodis[*] Abhishek Jain[†] Tal Moran[‡] Daniel Wichs[§]

January 19, 2012

Abstract

If we have a problem that is mildly hard, can we create a problem that is significantly harder? A natural approach to *hardness amplification* is the "direct product"; instead of asking an attacker to solve a single instance of a problem, we ask the attacker to solve several independently generated ones. Interestingly, proving that the direct product amplifies hardness is often highly non-trivial, and in some cases may be false. For example, it is known that the direct product (i.e. "parallel repetition") of general interactive games may not amplify hardness at all. On the other hand, positive results show that the direct product does amplify hardness for many basic primitives such as one-way functions/relations, weakly-verifiable puzzles, and signatures.

Even when positive direct product theorems are shown to hold for some primitive, the parameters are surprisingly weaker than what we may have expected. For example, if we start with a weak one-way function that no poly-time attacker can break with probability $> \frac{1}{2}$, then the direct product provably amplifies hardness to *some negligible* probability. Naturally, we would expect that we can amplify hardness exponentially, all the way to 2^{-n} probability, or at least to some fixed/known negligible such as $n^{-\log n}$ in the security parameter n, just by taking sufficiently many instances of the weak primitive. Although it is known that such parameters cannot be proven via black-box reductions, they may seem like reasonable conjectures, and, to the best of our knowledge, are widely believed to hold. In fact, a conjecture along these lines was introduced in a survey of Goldreich, Nisan and Wigderson (ECCC '95). In this work, we show that such conjectures are *false* by providing simple but surprising counterexamples. In particular, we construct weakly secure *signatures* and *one-way functions*, for which standard hardness amplification results are known to hold, but for which hardness does not amplify *beyond just negligible*. That is, for *any negligible* function $\varepsilon(n)$, we instantiate these primitives so that the direct product can always be broken with probability $\varepsilon(n)$, *no matter how many copies we take.*

1 Introduction

Hardness amplification is a fundamental cryptographic problem: given a "weakly secure" construction of some cryptographic primitive, can we use it to build a "strongly secure" construction? The first result in this domain is a classical conversion from weak one-way functions to strong one-way function by Yao [34] (see also [13]). This result starts with a function f which is assumed to be *weakly one-way*, meaning that it can be inverted on at most (say) a *half* of its inputs. It shows that the *direct-product* function $F(x_1, \ldots, x_k) =$

[*]NYU. E-mail: `dodis@cs.nyu.edu`

[†]UCLA. E-mail: `abhishek@cs.ucla.edu`. Research partially conducted while at IBM Research.

[‡]IDC Herzliya. E-mail: `talm@seas.harvard.edu`

[§]IBM Research T. J. Watson. E-mail: `wichs@cs.nyu.edu`

$(f(x_1), \ldots, f(x_k))$, for an appropriately chosen polynomial k, is one-way in the standard sense, meaning that it can be inverted on only a *negligible* fraction of its inputs. The above result is an example of what is called the *direct product theorem*, which, when true, roughly asserts that simultaneously solving many independent repetitions of a mildly hard task yields a much harder "combined task".[1] Since the result of Yao, such direct product theorems have been successfully used to argue security amplification of many other important cryptographic primitives, such as collision-resistant hash functions [8], encryption schemes [12], weakly verifiable puzzles [7, 22, 24], signatures schemes/MACs [11], commitment schemes [20, 9], pseudorandom functions/generators [11, 28], block ciphers [26, 29, 27, 32], and various classes of interactive protocols [5, 30, 21, 19].

Direct product theorems are surprisingly non-trivial to prove. In fact, in some settings, such as general interactive protocols [5, 31], they are simply false and hardness does not amplify at all, irrespective of the number of repetitions. Even for primitives such as one-way functions, for which we do have "direct product theorems", the parameters of these results are surprisingly weaker than what we may have expected. Let us say that a cryptographic construction is *weakly secure* if no poly-time attacker can break it with probability greater than $\frac{1}{2}$. Known theorems tell us that the direct product of $k = \Theta(n)$ independent instances of a weakly secure construction will become secure in the standard sense, meaning that no poly-time attacker can succeed in breaking security with better than *some negligible probability in the security parameter n*. However, we could naturally expect the direct product of k instances will amplify hardness exponentially, ensuring that no poly-time attacker can break security with more than 2^{-k} probability. Or, we would at least expect that a sufficiently large number of $k = \mathsf{poly}(n)$ repetitions can amplify hardness to some fixed/known negligible probability such as $\varepsilon(n) = 2^{-n^\delta}$ for some constant $\delta > 0$, or even less ambitiously, $\varepsilon(n) = n^{-\log n}$. We call such expected behavior *amplification beyond negligible.*

Limitation of Existing Proofs. One intuitive reason that the positive results are weaker than what we expect is the limitation of our reduction-based proof techniques. In particular, assume we wanted to show that the k-wise direct product amplifies hardness down to some very small probability ε. Then we would need an *efficient reduction* that uses an adversary $\mathcal{A}$ breaking the security of the k-wise direct product with probability ε, to break the security of a single instance with a much larger probability, say one half. Unfortunately, the reduction cannot get "anything useful" from the attacker $\mathcal{A}$ until it succeeds at least once. And since $\mathcal{A}$ only succeeds with small probability ε, the reduction is forced to run $\mathcal{A}$ at least (and usually much more than) $1/\varepsilon$ times, since otherwise $\mathcal{A}$ might never succeed. In other words, the reduction is only efficient as long as ε is an *inverse polynomial.* This may already be enough to show that the direct product amplifies hardness *to some negligible probability*, since the success probability of $\mathcal{A}$ must be smaller than *every* inverse polynomial ε. But it also tells us that black-box reductions cannot prove any stronger bounds *beyond negligible*, since the reduction would necessarily become inefficient.[2] For example, we cannot even prove that the k-wise direct product of a weak one-way function will amplify hardness to $n^{-\log n}$ security (where n is the security parameter), no matter how many repetitions k we take.

Our Question. The main goal of this work is to examine whether the limitations of current

[1] A related approach to amplifying the hardness of decisional problems is the "XOR Lemma" which roughly asserts the hardness of predicting an XOR of the challenge bits of many independent instances of a decisional problem will amplify. In this work, we will focus of "search" problems such as one-way functions and signatures and therefore only consider amplification via direct product.

[2] This "folklore" observation has been attributed to Steven Rudich in [15].

hardness amplification results are just an artifact our proof technique, or whether they reflect reality. Indeed, we may be tempted to ignore the lack of formal proofs and nevertheless make the following, seemingly believable conjecture that hardness does amplify *beyond negligible*:

Conjecture (Informal): *For all primitives for which standard direct product theorems hold (e.g., one-way functions, signatures etc.), the k-wise direct product of* any *weakly secure instantiation will amplify hardness all the way down to some fixed negligible bound $\varepsilon(n)$, such as $\varepsilon(n) = 2^{-\Omega(n)}$, or, less ambitiously, $\varepsilon(n) = n^{-\log n}$, when $k = \mathsf{poly}(n)$ is sufficiently large.*

To the best of our knowledge, such a conjecture is widely believed to hold. The survey of Goldreich et al. [15] explicitly introduced a variant of the above conjecture in the (slightly different) context of the XOR Lemma and termed it a *"dream version"* of hardness amplification which, although seemingly highly reasonable, happens to elude a formal proof.

Our Results. In this work, we show that, surprisingly, the above conjecture does *not* hold, and give strong counterexamples to the conjectured hardness amplification beyond negligible. We do so in the case of signature schemes and one-way functions for which we have standard direct-product theorems showing that hardness amplifies to negligible [34, 11]. Our result for the signature case, explained in Section 3, relies on techniques from the area of *stateless (resettably-secure) multiparty computation* [6, 3, 10, 18, 17]. On a high level, we manage to embed an execution of a stateless mutliparty protocol Π into the design of our signature scheme, where Π generates a random instance of a hard relation $\mathcal{R}$, and the signer will output its secret key if the message contains a witness for $\mathcal{R}$. The execution of Π can be driven via carefully designed signing queries. Since Π is secure and $\mathcal{R}$ is hard, the resulting signature scheme is still secure by itself. However, our embedding is done in a way so as to allow us to attack the direct product of *many independent* schemes by forcing them to execute a *single correlated* execution of Π resulting in a common instance of the hard relation $\mathcal{R}$. This allows us to break all of the schemes simultaneously by breaking a single instance of $\mathcal{R}$, and thus with some negligible probability $\varepsilon(n)$, which is independent of the number of copies k. Indeed, we can make $\varepsilon(n)$ an arbitrarily large negligible quantity (say, $n^{-\log n}$) by choosing the parameters for the relation $\mathcal{R}$ appropriately.

Result 1 (Informal). *Assuming the existence of a resettably secure MPC that generates a random instance of a hard relation and standard signature schemes, there exist signature schemes for which the direct product does not amplify security beyond negligible.*

One may wonder whether such counterexamples are particular to signature schemes. More specifically, our above counterexample seems to crucially rely on the fact that the security game for signatures is highly interactive (allowing us to embed an interactive MPC computation) and that the communication complexity between the challenger and attacker in the security game can be arbitrarily high (allowing us to embed data from all of the independent copies of the scheme into the attack on each individual one). Perhaps hardness still amplifies beyond negligible for simpler problems, such as one-way functions, where the security game is not interactive and has an a-priori bounded communication complexity. Our second result gives strong evidence that this too is unlikely, by giving a counterexmaple for one-way functions. The counterexample relies on a new assumption on a hash functions called *Extended Second Preimage Resistance* (ESPR), which we introduce in this paper. Essentially, this assumption says that given a random challenge x, it is hard to find a bounded-length *Merkle path* that starts at x, along with a collision on it. To break many independent copies of this problem, the attacker takes the independent challenges $x_1, \ldots, x_k$ and builds a *Merkle* *tree* with them as leaves. If it manages to find a *single* collision at the root of tree (which occurs with some probability independent of k), it will be able to find a witness (a Merkle path starting at x_i with a collision) for *each* of the challenges x_i. So far, this gives us an amplification counterexample for a *hard relation* based on the ESPR problem (which is already interesting), but, with a little more work, we can also convert it into a counterexample for a *one-way function* based on this problem. For the counterexample to go through, we need the ESPR assumption to hold for some fixed hash function (not a family), and so we cannot rely on collision resistance. Nevertheless, we argue that the ESPR assumption for a fixed hash function is quite reasonable and is likely satisfied by existing (fixed) cryptographic hash functions, by showing that it holds in a variant of the random oracle model introduced by Unruh [33], where an attacker gets arbitrary "oracle-dependent auxiliary input". As argued by [33], such model is useful for determining which security properties can be satisfies by a single hash function rather than a family.

Result 2 (Informal). *Assuming the existence of* extended second preimage resistant *(ESPR) hash functions, there exist one-way relations and one-way functions for which the direct product does not amplify security beyond negligible.*

Overall, our work gives strong indications that the limitations of our reductionist proofs for the direct product theorems might actually translate to real attacks for some schemes.

Related Work. Interestingly, a large area of related work comes from a seemingly different question of *leakage amplification* [2, 1, 25, 23]. These works ask the following: given a primitive P which is resilient to ℓ bits of leakage on its secret key, is it true that breaking k independent copies of P is resilient to almost $L = \ell k$ bits of leakage? At first sight this seems to be a completely unrelated question. However, there is a nice connection between hardness and leakage-resilience: if a primitive (such as a signature or one-way function) is hard to break with probability ε, then it is resilient to $\log(1/\varepsilon)$ bits of leakage. This means that if some counter-example shows that the leakage bound L does not amplify with k, then neither does the security. Therefore, although this observation was never made, the counterexamples to leakage amplification from [25, 23] seem to already imply some counterexample for hardness. Unfortunately, both works concentrate on a modified version of parallel repetition, where some *common public parameters* are reused by all of the instances and, thus, they are *not truly independent*. Indeed, although showing counterexamples for (the harder question of) leakage amplification is still interesting in this scenario, constructing ones for hardness amplification becomes trivial.[3] However, the work of [23] also proposed that a variant of their counterexample for leakage amplification may extend to the setting without common parameters under a highly non-standard assumption about computationally sound (CS) proofs. Indeed, this suggestion led us to re-examine our initial belief that such counterexamples should not exist, and eventually resulted in this work. We also notice that our counterexample for signature schemes (but not one-way functions) can be easily extended to give a counterexample for leakage amplification without common parameters.

2 Hardness Amplification Definitions and Conjectures

In this work, we will consider a non-uniform model of computation. We equate entities such as challengers and attackers with circuit families, or equivalently, Turing Machines with

[3]E.g., the hard problem could ask to break either the actual instance or the common parameter. While such an example does not necessarily contradict leakage amplification, it clearly violates hardness amplification.

advice. We let n denote the *security parameter.* We say that a function $\varepsilon(n)$ is *negligible* if $\varepsilon(n) = n^{-\omega(1)}$.

We begin by defining a general notion of (single prover) cryptographic games, which captures the security of the vast majority of cryptographic primitives, such as one-way functions, signatures, etc.

Definition 2.1 (Games)**.** A *game* is defined by a probabilistic interactive *challenger* $\mathcal{C}$. On security parameter n, the challenger $\mathcal{C}(1^n)$ interacts with some attacker $\mathcal{A}(1^n)$ and may output a special symbol win. If this occurs, we say that $\mathcal{A}(1^n)$ *wins* $\mathcal{C}(1^n)$.

We can also define a *class* $\mathbb{C}$ of cryptographic games $\mathcal{C} \in \mathbb{C}$. For example the factoring problem fixes a particular game with the challenger $\mathcal{C}_{FACTOR}$ that chooses two random n-bit primes p, q, sends $N = p \cdot q$ to $\mathcal{A}$, and outputs win iff it gets back p, q. On the other hand, *one-way functions* can be thought of as a class of games $\mathbb{C}_{OWF}$, where each candidate one-way function f defines a particular game $\mathcal{C}_f \in \mathbb{C}_{OWF}$ where the challenger samples a random x (from an appropriate domain), gives $f(x)$ to the attacker, and outputs win if it get back x' s.t. $f(x') = f(x)$. We assume that classes of games are only defined *syntactically* in terms of the structure of the challenger in the security game and so, when we talk about a class like one-way functions/relations or signatures (the only ones we will consider in this paper), it also includes all insecure candidates. In fact, the reader can think of the notion of a *cryptographic game in some class* as being interchangeable with the notion of a *candidate scheme for the corresponding primitive* – each candidate scheme defines a concrete challenger/game that captures its security. We now define what it means for a game to be *hard* (correspondingly, for a candidate scheme to be secure).

Definition 2.2 (Hardness)**.** We say that the game $\mathcal{C}$ is $(s(n), \varepsilon(n))$-hard if, for all sufficiently large $n \in \mathbb{N}$ and all $\mathcal{A}(1^n)$ of size $s(n)$ we have $\Pr[\mathcal{A}(1^n) \textit{ wins } \mathcal{C}(1^n)] < \varepsilon(n)$. We say that the game $\mathcal{C}$ is $(\mathsf{poly}, \varepsilon(n))$-hard if it is $(s(n), \varepsilon(n))$-hard for all polynomial $s(n)$. We say that the game $\mathcal{C}$ is $(\mathsf{poly}, \mathsf{negl})$-hard if it is $(s(n), 1/p(n))$-hard for all polynomials $s(n)$, $p(n)$.

Definition 2.3. [Direct Product] For a cryptographic game $\mathcal{C}$ we define the *k-wise direct-product game $\mathcal{C}^k$*, which initializes k independent copies $\mathcal{C}_1, \ldots, \mathcal{C}_k$ of the challenger $\mathcal{C}$. It proceeds in rounds where, in each round, it receives input $\bar{x} = (x_1, \ldots, x_k)$ from the attacker, gives each x_i to the corresponding $\mathcal{C}_i$ receiving the reply y_i, and finally replies with $\bar{y} = (y_1, \ldots, y_k)$ to the attacker. It outputs the win symbol if and only if each of the k copies $\mathcal{C}_i$ individually output win.

We note that specialized variants of the direct product also make sense in some applications, and will discuss one such variant in the case of signatures. Finally, we are ready to formally define what we mean by hardness amplification. Since we focus on negative results, we will distinguish between several broad levels of hardness amplification and ignore exact parameters. For example, we do not pay attention to the number of repetitions k needed to reach a certain level of hardness (an important parameter for positive results), but are more concerned with which levels of hardness are or are not reachable altogether.

Definition 2.4 (Hardness Amplification)**.** For a fixed game $\mathcal{C}$, we say that *hardness amplifies to $\varepsilon = \varepsilon(n)$* if there exists some polynomial $k = k(n)$ such that $\mathcal{C}^k$ is $(\mathsf{poly}, \varepsilon)$-hard. We say that hardness amplifies *to negligible* if there exists some polynomial $k = k(n)$ such that $\mathcal{C}^k$ is $(\mathsf{poly}, \mathsf{negl})$-hard. For a *class* $\mathbb{C}$ of games, we say that:

1. The hardness of a class $\mathbb{C}$ amplifies *to negligible* if, for every game $\mathcal{C} \in \mathbb{C}$ which is $(\mathsf{poly}, \frac{1}{2})$-hard, the hardness of $\mathcal{C}$ amplifies to negligible.
2. The hardness of a class $\mathbb{C}$ amplifies to $\varepsilon(n)$ if, for every game $\mathcal{C} \in \mathbb{C}$ which is $(\mathsf{poly}, \frac{1}{2})$-hard, the hardness of $\mathcal{C}$ amplifies to $\varepsilon(n)$.
3. The hardness of a class $\mathbb{C}$ amplifies *beyond negligible* if there exists some negligible function $\varepsilon(n)$ for the entire class, such that the hardness of $\mathbb{C}$ amplifies to $\varepsilon(n)$.

Remarks on Definition. The standard "direct product theorems" for classes such as one-way functions/relations and signatures show that the hardness of the corresponding class amplifies *to negligible* (bullet 1). For example, if we take *any* $(\mathsf{poly}, 1/2)$-hard function f, then a sufficiently large direct product f^k will be $(\mathsf{poly}, \mathsf{negl})$-hard.[4] However, what "negligible" security can we actually get? The result does not say and it may depend on the function f that we start with.[5] One could conjecture that there is *some* fixed negligible $\varepsilon(n)$ such that a sufficiently large direct product of *any* weak instantiation will amplify its hardness to $\varepsilon(n)$. This is amplification *beyond negligible* (bullet 3). More ambitiously, we could expect that this negligible $\varepsilon(n)$ is very small such as $\varepsilon(n) = 2^{-n^{\Omega(1)}}$ or even $2^{-\Omega(n)}$. We explicitly state these conjectures below.

Dream Conjecture (Weaker): For any class of cryptographic games $\mathbb{C}$ for which hardness amplifies *to negligible*, the hardness of $\mathbb{C}$ also amplifies *beyond negligible*.

Dream Conjecture (Stronger): For any class of cryptographic games $\mathbb{C}$ for which hardness amplifies *to negligible*, the hardness of $\mathbb{C}$ also amplifies to some $\varepsilon(n) = 2^{-n^{\Omega(1)}}$.

Our work gives counterexamples to both conjectures. We give two very different types counterexamples: one for the classes of *signature schemes* (Section 3) and one for the class of *one-way functions/relations* (Section 4). Our counterexamples naturally require that some hard instantiations of these primitives exist to begin with, and our counterexamples for the weaker versions of the dream conjecture will actually require the existence of *exponentially hard* versions of these primitives. In particular, under strong enough assumptions, we will show that for *every* negligible function $\varepsilon(n)$ there is stand-alone scheme which is *already* $(\mathsf{poly}, \mathsf{negl})$-hard, but whose k-wise direct product is *not* $(\mathsf{poly}, \varepsilon(n))$-hard, no matter how large k is. In fact, we will rely on the following theorem that makes it easier to construct such dramatic counterexamples to the dream conjectures. It tells us that it is sufficient to just find a weakly secure scheme whose k-wise direct product can always be broken with some fixed probability $2^{-c \cdot n}$ for some constant $c \geq 0$, no matter how many copies k we take. We can then use it to also get weakly secure schemes whose hardness does not amplify beyond *any* negligible .

Theorem 2.5. *Assume that a class $\mathbb{C}$ contains some game $\mathcal{C}$ which is $(\mathsf{poly}, \frac{1}{2})$-hard but whose k-wise direct-product $\mathcal{C}^k$ has an attack of size $\mathsf{poly}(k, n)$ with success probability $2^{-c \cdot n}$, for some constant $c \geq 0$ independent of k. Then the hardness of the class $\mathbb{C}$ does not amplify to any $\varepsilon(n) = 2^{-n^{\Omega(1)}}$. If, in addition, $\mathcal{C}$ is $(s(n), \frac{1}{2})$-hard for some $s(n) = 2^{\Omega(n)}$, then the hardness of $\mathbb{C}$ does not amplify beyond negligible.*

[4] The choice of 1/2 is arbitrary and can be replaced with any constant or even any function bounded-away-from 1. We stick with 1/2 for concreteness and simplicity.

[5] It also seemingly depends on the exact polynomial size $s(n)$ of the attackers we are trying to protect against. However, using a result of Bellare [4], the dependence on $s(n)$ can always be removed.

Proof. The main idea is to take the original game with challenger $\mathcal{C}(1^n)$ and convert it into a modified challenger $\mathcal{C}'_{m(\cdot)}(1^n)$ which executes $\mathcal{C}(1^m)$, where $m = m(n)$ is some carefully chosen value smaller than n. In particular, for a given function $\varepsilon(n) \geq 2^{-n}$, set $m(n) = \frac{1}{c}\log(1/\varepsilon(n))$. Then the k-wise direct product of $\mathcal{C}'_{m(\cdot)}$ can be broken by a $\mathsf{poly}(k, m) = \mathsf{poly}(n)$ attack with success probability $\geq 2^{-c \cdot m(n)} \geq \varepsilon(n)$.

For the first part of the theorem, we can take *any* $\varepsilon(n) = 2^{-n^{\delta}}$ for *any* constant $\delta \in (0, 1]$, and get the game $\mathcal{C}'_{m(\cdot)}$ where $m(n) = \frac{1}{c}\log(1/\varepsilon(n)) = n^{\delta}/c$. This game remains $(\mathsf{poly}, \frac{1}{2})$-hard since $\mathsf{poly}(n) = \mathsf{poly}(m)$. Therefore for each such $\varepsilon(n)$, there is a corresponding game which is $(\mathsf{poly}, \frac{1}{2})$-hard but whose k-wise direct product does not amplify to $\varepsilon(n)$, no matter how large k is.

For the second part of the theorem, we can take *any* negligible $\varepsilon(n) = n^{-\omega(1)} \geq 2^{-n}$ and get the game $\mathcal{C}'_{m(\cdot)}$ where $m(n) = \frac{1}{c}\log(1/\varepsilon(n))$. Now since, $\mathcal{C}$ is $(s(n), \frac{1}{2})$-hard for $s(n) = 2^{\Omega(n)}$ it means that $\mathcal{C}'_{m(\cdot)}$ is $(s'(n), \frac{1}{2})$-hard where $s'(n) = s(m(n)) = 2^{\Omega(\log(1/\varepsilon(n)))} = n^{\omega(1)}$. So $\mathcal{C}'_{m(\cdot)}$ is also $(\mathsf{poly}, \frac{1}{2})$-hard. Therefore for each such negligible $\varepsilon(n)$, there is a corresponding game which is $(\mathsf{poly}, \frac{1}{2})$-hard but whose k-wise direct product does not amplify to $\varepsilon(n)$, no matter how large k is.

(*Remark:* In the proof, we implicitly assume the class $\mathbb{C}$ is robust so that, when $\mathcal{C} \in \mathbb{C}$ then the corresponding game $\mathcal{C}'_{m(\cdot)} \in \mathbb{C}$. This is true for syntactic definitions of classes such as candidate one-way functions or signatures. However, for the second part of the theorem, the function $m(n)$ may not be efficiently computable if $\log(1/\varepsilon(n))$ is not efficiently computable in $\mathsf{poly}(n)$ time. In that case, the challenger $\mathcal{C}'_{m(\cdot)}$ in the counterexample may be not be uniformly efficient even when $\mathcal{C}$ is (it is always non-uniformly efficient since $m(n)$ can be hardcoded). We can either live with having non-uniform counterexamples or we can restrict ourselves to showing a counterexample for every negligible function $\varepsilon(n)$ for which $\log(1/\varepsilon(n))$ is efficiently computable in $\mathsf{poly}(n)$ time.)

□

2.1 Hard and One-Way Relations

As a component of both counterexamples, we will rely on the following definition of hard relations phrased in the framework of cryptographic games:

Definition 2.6 (Hard Relations)**.** Let p be some polynomial and $R \subseteq \bigcup_{n \in \mathbb{N}} \{0,1\}^n \times \{0,1\}^{p(n)}$ be an NP relation consisting of pairs (y, w) with *instances* y and *wintesses* w. Let $L = \{\ y\ :\ \exists\ w \text{ s.t. } (y, w) \in R\}$ be the corresponding NP language. Let $y \leftarrow \textsc{SamL}(1^n)$ be a PPT algorithm that samples values $y \in L$. For a relation $\mathcal{R} = (R, \textsc{SamL})$, we define the corresponding security game where the challenger $\mathcal{C}(1^n)$ samples $y \leftarrow \textsc{SamL}(1^n)$ and the adversary wins if it outputs w s.t. $(y, w) \in R$. As with other cryptographic games, we can talk about $(s(n), \varepsilon(n))$-hard relations, as well as $(\mathsf{poly}, \mathsf{negl})$-hard relations.

Note that, for hard relations, we only require that there is an efficient algorithm for sampling hard instances y. Often in cryptography we care about a sub-class of hard relations, which we call *one-way relations*, where is it also feasible to efficiently sample a hard instance y *along with* a witness w. We define this below.

Definition 2.7 (One-Way Relation)**.** Let R be an NP relation and L be the corresponding language. Let $(y, w) \leftarrow \textsc{SamR}(1^n)$ be a PPT algorithm that samples values $(y, w) \in R$, and define $y \leftarrow \textsc{SamL}(1^n)$ to be a restriction of $\textsc{SamR}$ to its first output. We say that $(R, \textsc{SamR})$ is a *one-way relation* if $(R, \textsc{SamL})$ is a hard relation.

3 Counterexample for Signature Schemes

3.1 Overview

We use the standard notions of signature security, meaning existential unforgeability against chosen message attack [16]. The work of [11] shows that the direct product of any *stateless* weak signature schemes amplifies hardness *to negligible*. We now show that it does not (in general) amplify hardness beyond negligible. For signatures, it is natural to consider a variant of the direct product, which we call the "*same-message direct product*" where, instead of asking the attacker to break k independent schemes by interacting with each challenger separately, we also restrict the attacker to always querying the *same* message for all k schemes in its signing queries and forging a signature on the *same* message for all k schemes. This corresponds to the natural way of using many weak signature schemes to construct a single strong signature by signing each message under each scheme separately. Since the "same-message direct product" just restricts the attacker further, any counterexample of this form will also imply a counterexample for the standard direct product (Definition 2.3). Therefore, in the rest of the section we will consider the same-message direct product which gives us the strongest and most meaningful counterexample result.

We will give a transformation from any standard signature scheme (Gen, Sign, Verify) into a new signature scheme (Gen, Sign, Verify) whose hardness does not amplify (via a direct product) beyond negligible. We start by giving an informal description of the transformation to illustrate our main ideas. In order to convey the intuition clearly, we will first consider a simplified case where the signing algorithm Sign of the (modified) scheme (Gen, Sign, Verify) is *stateful*, and will then discuss how to convert the stateful signing algorithm into one that is *stateless*.[6]

Embedding MPC in Signatures. Let (Gen, Sign, Verify) be any standard signature scheme. Let $\mathcal{F} = \{\mathcal{F}_k\}_{k\in\mathbb{N}}$ be a randomized k-party "ideal functionality" that takes no inputs and generates a random instance y of a hard relation $\mathcal{R} = (R, \textsc{SamL})$ according to the distribution SamL. Further, let $\Pi = \{\Pi_k\}_{k\in\mathbb{N}}$ be a multi-party computation protocol that securely realizes the functionality $\mathcal{F}$ for any number of parties k. Then, the new signature scheme (Gen, Sign, Verify) works as follows.

Algorithms Gen and Verify are identical to Gen and Verify respectively. The signing algorithm Sign is essentially the same as Sign, except that, on receiving a signing queries of a "special form", Sign interprets these as "protocol messages" for Π_k and (in addition to generating a signature of them under Sign) also executes the *next message function* of the protocol and outputs its response as part of the new signature. A special initialization query specifies the number of parties k involved in the protocol and the role P_i in which the signing algorithm should act. The signing algorithm then always acts as the honest party P_i while the user submitting signing queries can essentially play the role of the remaining $k-1$ parties. When Π_k is completed yielding some output y (interpreted as the instance of a hard relation $\mathcal{R}$) the signing algorithm Sign will look for a signing query that contains a corresponding witness w, and, if it receives one, will respond to it by simply outputting its entire *secret key* in the signature. The security of the transformed signature (Gen, Sign,

[6]We note that in the setting of stateful signatures, hardness fails to amplify even *to* negligible since we can embed the counterexamples of [5, 31] into the signature scheme. Nevertheless our initial description of our counterexample for the stateful setting will clarify the main new result, which is a counterexample for the stateless setting.

VERIFY) immediately follows from the security of the MPC protocol Π_k against all-but-one corruptions, the hardness of the relation $\mathcal{R}$ and the security of the original signature scheme.

Attacking the Direct-Product. Let us now briefly demonstrate an adversary $\mathcal{A}$ for the k-wise direct product. Very roughly, $\mathcal{A}$ carefully chooses his signing queries so as to force $\text{SIGN}_1, \ldots, \text{SIGN}_k$ to engage in a *single* execution of the protocol Π_k, where each SIGN_i plays the role of a different party P_i, while $\mathcal{A}$ simply acts as the "communication link" between them. This results in all component schemes SIGN_i generating a common instance y of the hard relation. Finally, $\mathcal{A}$ simply "guesses" a witness w for y at random and, if it succeeds, submits w as a signing query, thereby learns the secret key of *each* component signature scheme thereby breaking all k of them! Note that the probability of guessing w is bounded by some negligible function in n and is *independent* of the number of parallel repetitions k.

Stateful to Stateless. While the above gives us a counterexample for the case where SIGN is a *stateful* algorithm, (as stated above) we are mainly interested in the (standard) case where SIGN is *stateless*. In order to make SIGN a stateless algorithm, we can consider a natural approach where we use a modified version Π'_k of protocol Π_k: each party P_i in Π'_k computes an outgoing message in essentially the same manner as in Π_k, except that it also attaches an *authenticated encryption* of its current protocol state, as well as the previous protocol message. This allows each (stateless) party P_i to "recover" its state from the previous round to compute its protocol message in the next round. Unfortunately, this approach is insufficient, and in fact insecure, since an adversarial user can *reset* the (stateless) signing algorithm at any point and achieve the effect of *rewinding* the honest party (played by the signing algorithm) during the protocol Π_k. To overcome this problem, we leverage techniques from the notion of resettably-secure computation. Specifically, instead of using a standard MPC protocol in the above construction, we use a recent result of Goyal and Maji [17] which constructs an MPC protocol that is secure against reset attacks and works for stateless parties for a large class of functionalities, including "inputless" randomized functionalities (that we will use in this paper).

The above intuitive description hides many details of how the user can actually "drive" the MPC execution between the k signers within the direct-product game where all signers respond to a single common message. We proceed to make this formal in the following section.

3.2 Our Signature Scheme

We now give our transformation from any standard signature scheme into one whose hardness does not amplify beyond negligible. We first establish some notation.

Notation. Let n be the security parameter. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be any standard signature scheme. Further, let $(\mathcal{R}, \text{SAML})$ be a hard relation as per Definition 2.6. Let $\{PRF_K : \{0,1\}^{\mathsf{poly}(n)} \rightarrow \{0,1\}^{\mathsf{poly}(n)}\}_{K \in \{0,1\}^n}\}$ be a pseudo-random function family.

Stateless MPC. We consider a randomized k-party functionality $\mathcal{F} = \{\mathcal{F}\}_{k \in \mathbb{N}}$ that does not take any inputs; $\mathcal{F}$ simply samples a random pair $y \leftarrow \text{SAML}(1^n)$ and outputs y to all parties. Let $\{\Pi_k\}_{k \in \mathsf{poly}(n)}$ be a family of protocols, where each $\Pi_k = \{P_1, \ldots, P_k\}$ is a k-party MPC protocol for computing the functionality $\mathcal{F}$ in the *public state model.* This model is described formally in Appendix A, and we only give a quick overview here. Each party P_i

is completely described by the next message function NM_i, which takes the following four values as input: (a) a string π_{j-1} that consists of all the messages sent in any round $j-1$ of the protocol, (b) the *public state* state_i of party P_i, and (c) the secret randomness r_i. On receiving an input of the form $\pi_{j-1}\|\mathsf{state}_i\|r_i$, NM_i outputs P_i's message in round j along with the updated value of state_i. We assume that an attacker corrupts (exactly) $k-1$ of the parties. In the real-world execution, the attacker can arbitrarily call the next-message function NM_i of the honest party P_i with arbitrarily chosen values of the public state state_i and arbitrary message π_{j-1} (but with an honestly chosen and secret randomness r_i). Nevertheless, the final output of P_i and the view of the attacker can be simulated in the ideal world where the simulator can "reset" the ideal functionality. In our case, that means that the attacker can adaptively choose one of polynomially many honestly chosen instances $y_1, \ldots, y_q$ of the hard relation which P_i will then accept as output.

The Construction. We now formally describe our signature scheme ($\textsc{Gen}, \textsc{Sign}, \textsc{Verify}$).

$\textsc{Gen}(1^n)$: Compute $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$. Also, sample a random tape $K \leftarrow \{0,1\}^{\mathsf{poly}(n)}$ and a random identity $\mathsf{id} \in \{0,1\}^n$. Output $PK = (pk, \mathsf{id})$ and $SK = (sk, K, \mathsf{id})$.

$\textsc{Sign}(SK, m)$: To sign a message m using secret key $SK = (sk, K, \mathsf{id})$, the signer outputs a signature $\sigma = (\sigma^1, \sigma^2)$ where $\sigma^1 \leftarrow \mathsf{Sign}(sk, m)$. Next, if m does not contain the prefix "$\texttt{prot}$", then simply set $\sigma^2 = \{0\}$. Otherwise, parse $m = (\text{"}\texttt{prot}\text{"}\|\mathsf{IM}\|\pi_j\|\mathsf{state}\|w)$, where $\mathsf{IM} = k\|\mathsf{id}_1\|\ldots\|\mathsf{id}_k$ such that $\mathsf{state} = \mathsf{state}_1\|\ldots\|\mathsf{state}_k$, then do the following:

- Let $i \in [k]$ be such that $\mathsf{id} = \mathsf{id}_i$. Compute $r_i = PRF_K(\mathsf{IM})$. Then, apply the next message function NM_i of (stateless) party P_i in protocol Π_k over the string $\pi_j\|\mathsf{state}_i\|r_i$ and set σ^2 to the output value.[7]
- Now, if σ^2 contains the output y of protocol Π_k,[8] then further check whether $(y, w) \in \mathcal{R}$. If the check succeeds, set $\sigma^2 = SK$.

$\textsc{Verify}(PK, m, \sigma)$: Given a signature $\sigma = (\sigma^1, \sigma^2)$ on message m with respect to the public key $PK = (pk, \mathsf{id})$, output 1 iff $\textsc{Verify}(pk, m, \sigma^1) = 1$.

This completes the description of our signature scheme. We prove the following theorem showing that the signature scheme satisfies basic signature security in Appendix B.

Theorem 3.1. *If (Gen, Sign, Verify) is a secure signature scheme, $\{PRF_K\}$ is a PRF family, $\mathcal{R}$ is a hard relation, and Π_k is a stateless MPC protocol for functionality $\mathcal{F}$, then the proposed scheme ($\textsc{Gen}, \textsc{Sign}, \textsc{Verify}$) is a secure signature scheme.*

3.3 Attack on the Direct Product

Theorem 3.2. *Let ($\textsc{Gen}, \textsc{Sign}, \textsc{Verify}$) be the described signature scheme and let $\mathcal{R} = (\textsc{SamL}, R)$ be the hard relation used in the construction. Assume that for any $y \xleftarrow{\$} \textsc{SamL}(1^n)$, the size of the corresponding witness w is bounded by $|w| = p(n)$ for some polynomial $p(\cdot)$.*

[7] Note that here σ^2 consists of party P_i's protocol message in round $j+1$, and its updated public state state_i. See Section A for details.

[8] Note that this is the case when j is the final round in Π_k. Here we use the property that the last round of Π_k is the *output delivery round*, and that when NM_i is computed over the protocol messages of this round, it outputs the protocol output.

Then, for any polynomial $k = k(n)$, there is an attack against the k-wise direct product running in time $\mathsf{poly}(k, n)$ *with success probability $\varepsilon(n) = 2^{-p(n)}$.*

We will prove Theorem 3.2 by constructing an adversary $\mathcal{A}$ that mounts a key-recovery attack on any k-wise direct product of the signature scheme (Gen, Sign, Verify).

k-wise Direct Product. Let (**Gen**, **Sign**, **Verify**) denote the k-wise direct product of the signature scheme (Gen, Sign, Verify), described as follows. Algorithm **Gen** runs Gen k-times to generate k key pairs (PK_1, SK_1),...,(PK_k, SK_k). To sign a message m, **Sign** computes $\sigma_i \leftarrow \textsc{Sign}(SK_i, m)$ for every $i \in k$ and outputs $\sigma = (\sigma_1, \ldots, \sigma_k)$. Finally, on input a signature $\sigma = (\sigma_1, \ldots, \sigma_k)$ on message m, **Verify** outputs 1 iff $\forall i \in k$, $\textsc{Verify}(PK_i, m, \sigma_i) = 1$.

Description of $\mathcal{A}$. We now describe the adversary $\mathcal{A}$ for (**Gen**, **Sign**, **Verify**). Let $(PK_1, \ldots, PK_k)$ denote the public key that $\mathcal{A}$ receives from the challenger of the signature scheme (**Gen**, **Sign**, **Verify**), where each $PK_i = (pk_i, \mathsf{id}_i)$. The adversary $\mathcal{A}$ first sends a signing query m_0 of the form "prot"$\|\mathsf{IM}\|\pi_0\|\mathsf{state}\|w$, where $\mathsf{IM} = k\|\mathsf{id}_1\|\ldots\|\mathsf{id}_k$, and $\pi_0 = \mathsf{state} = w = \{0\}$. Let $\sigma = (\sigma_1, \ldots, \sigma_k)$ be the response it receives, where each $\sigma_i = \sigma_i^1, \sigma_i^2$. $\mathcal{A}$ now parses each σ_i^2 as a first round protocol message π_1^i from party P_i followed by the public state state_i of P_i (at the end of the first round) in protocol Π_k.

$\mathcal{A}$ now prepares a new signing query m_1 of the form "prot"$\|\mathsf{IM}\|\pi_1\|\mathsf{state}\|w$, where IM and w are the same as before, but $\pi_1 = \pi_1^1\|\ldots\|\pi_1^k$, and $\mathsf{state} = \mathsf{state}_1\|\ldots\|\mathsf{state}_k$. On receiving the response, $\mathcal{A}$ repeats the same process as above to produce signing queries $m_2, \ldots, m_{t-1}$, where t is the total number of rounds in protocol Π_k. (That is, each signing query $m_2, \ldots, m_{t-1}$ is prepared in the same manner as m_1.)

Finally, let $\sigma = (\sigma_1, \ldots, \sigma_k)$ be the response to the signing query m_{t-1}. $\mathcal{A}$ now parses each σ_i^2 as the round t protocol message π_t^i from party P_i followed by the state state_i of P_i. Now, since the final round (i.e., round t) of protocol Π_k is the *output delivery round*, and further, Π_k satisfies the *publicly computable output* property, $\mathcal{A}$ simply computes the protocol output y from the messages $\pi_t^1, \ldots, \pi_t^k$. Now, $\mathcal{A}$ guesses a $p(n)$-sized witness $w^* \xleftarrow{\$} \{0,1\}^{p(n)}$ at random and, if $(y, w^*) \in \mathcal{R}(x)$, it now sends the final signing query $m_t =$"prot"$\|\mathsf{IM}\|\pi_t\|\mathsf{state}\|w$, where IM is the same as before, $\pi_t = \pi_t^1\|\ldots\|\pi_t^k$, $\mathsf{state} = \mathsf{state}_1\|\ldots\|\mathsf{state}_k$, and $w = w^*$. Thus, $\mathcal{A}$ obtains $SK_1, \ldots, SK_k$ from the challenger and can forge arbitrary signatures for the direct product scheme. It's clear that its success probability is at least $2^{-p(n)}$.

Corollary 3.3. *Assuming the existence hard relations and a general stateless MPC compilers, the hardness of signature schemes does* not *amplify to any $\varepsilon(n) = 2^{-n^{\Omega(1)}}$. This gives a counterexample to the strong dream conjecture for signature schemes. If we, in addition, assume the existence of $(2^{\Omega(n)}, 2^{-\Omega(n)})$-hard relations with witness size $c \cdot n$ for some constant $c \geq 0$, then there exist signature schemes whose hardness does not amplify* beyond negligible. *This gives a counterexample to the weak dream conjecture.*

Proof. We essentially just combine Theorem 3.2 and Theorem 2.5. This gives the first part of the corollary. For the second part of the corollary, a naive application of Theorem 2.5 would require that the entire signature scheme from our above counterexample is exponentially secure, which would require exponentially secure MPC. It turns out that this is overkill and we just need the relation $\mathcal{R}$ to be exponentially secure. In particular, for any negligible $\varepsilon(n)$, we just instantiate the relation $\mathcal{R}$ with security parameter $m(n) = -\frac{1}{c}\log(1/\varepsilon(n))$ in which case the relation remains $(\mathsf{poly}, \mathsf{negl})$-hard. Therefore, the resulting signature scheme is $(\mathsf{poly}, \mathsf{negl})$-secure but, by Theorem 3.2, the direct product can be broken in $\mathsf{poly}(n)$ time with probability $\varepsilon(n)$. ∎

Remark on Leakage Amplification. We can also use the above idea to get a counterexample to *leakage-amplification* of signature schemes. In particular, instead of using (just) a hard relation $\mathcal{R} = (R, \textsc{SamL})$, let us use a one-way relation $\mathcal{R} = (R, \textsc{SamR})$ where we can efficiently sample instance/witness pairs $(y, w) \stackrel{\$}{\leftarrow} \textsc{SamR}(1^n)$. Let us also change the ideal functionality $\mathcal{F}$ so that it samples $(y, w) \stackrel{\$}{\leftarrow} \textsc{SamR}(1^n)$, outputs y to all parties, and gives a *secret share* w_i of w to each party P_i so that $\bigoplus_{i=1}^{k} w_i = w$. We then use the same signature construction as above, but with the protocol Π implementing the new functionality $\mathcal{F}$. It is easy to see that the stand-alone construction remains a secure signature scheme in the standard sense without leakage, which also implies that it is secure to $O(\log n)$ bits of leakage on its secret. However, the direct product of any k signatures can be broken just by leaking the short witness w, whose size is independent of the number of schemes k. In particular, w can be efficiently computed given only the secret keys of the k signature schemes, since the randomness used by each signer during the execution of Π is a part of its secret key, and hence the leakage can just run the protocol Π in its head and reconstruct w. This counterexample yields a similar result as [23], showing lack of leakage amplification even when there are no common parameters between the different schemes, but under standard computational assumptions.

4 Counterexample for One-Way Functions and Relations

In Section 3, we proved that there exist signature schemes whose hardness does not amplify beyond negligible. This already rules out the general conjecture that "for *any* game for which hardness amplifies to negligible, hardness will also amplify to exponential (or at least beyond negligible)". Nevertheless, one might still think that the conjecture hold for more restricted classes of games. Perhaps the simplest such class to consider is one-way functions. Note that, unlike the case for signature schemes, the one-way function game does not allow interaction and has bounded communication between attacker and challenger. Thus, the general strategy we employed in Section 3 of embedding a multiparty computation inside signature queries, will no longer work. In this section, we propose an alternate strategy for showing that hardness of one-way functions does not amplify beyond negligible.

4.1 Extended Second-Preimage Resistant (ESPR) Hash Functions

Our construction is based on a new (non-standard) cryptographic security assumption on hash functions. Let $h : \{0,1\}^{2n} \mapsto \{0,1\}^n$ be a hash function. We define a *Merkle path of length* ℓ to be a tuple of the form

$$p_\ell = (x_0, (b_1, x_1), \ldots, (b_\ell, x_\ell)) \quad : \quad b_i \in \{0,1\}, x_i \in \{0,1\}^n.$$

Intuitively, x_0 could be the *leaf* of some Merkle tree of height ℓ, and the values $x_1, \ldots, x_\ell$ are the siblings along the path from the leaf to the *root*, where the bits b_i indicate whether the sibling x_i is a left or right sibling. However, we can also talk about a path p_ℓ on its own, without thinking of it as part of a larger tree. Formally, if p_ℓ is a Merkle path as above, let $p_{\ell-1}$ be the path with the last component (b_ℓ, x_ℓ) removed. The *value* of a Merkle path p_ℓ as above is defined iteratively via:

$$\bar{h}(p_\ell) = \begin{cases} h(\bar{h}(p_{\ell-1}), x_\ell) & \ell > 0, b_\ell = 1 \\ h(x_\ell, \bar{h}(p_{\ell-1})) & \ell > 0, b_\ell = 0 \\ x_0 & \ell = 0 \end{cases}$$

We call x_0 the *leaf* of the path p_ℓ, and $z = \bar{h}(p_\ell)$ is its *root*. We say that $y = (x_L, x_R) \in \{0,1\}^{2n}$ is the *known preimage of the path* p_ℓ if x_L, x_R are the values under the root, so that either $x_L = x_\ell$, $x_R = \bar{h}(p_{\ell-1})$ if $b_\ell = 0$, or $x_L = \bar{h}(p_{\ell-1}), x_R = x_\ell$ if $b_\ell = 1$. Note that this implies $h(y) = \bar{h}(p_\ell)$. We say that $y' \in \{0,1\}^{2n}$ is a *second preimage of the path* p_ℓ if $y' \neq y$ is *not* the known preimage of p_ℓ, and $h(y') = \bar{h}(p_\ell)$. We are now ready to define the *extended second-preimage resistance (ESPR)* assumption. This assumption says that, given a random challenge $x_0 \in \{0,1\}^n$, it is hard to find a (short) path p_ℓ containing x_0 as a leaf, and a second-preimage y' of p_ℓ.

Definition 4.1 (ESPR)**.** Let $h : \{\{0,1\}^{2n} \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a poly-time computable hash function. We define the *Extended Second Preimage Resistance* (ESPR) assumption on h via the following security game between a challenger and an adversary $\mathcal{A}(1^n)$:

1. The challenger chooses $x_0 \xleftarrow{\$} \{0,1\}^n$ at random and gives it to $\mathcal{A}$.
2. $\mathcal{A}$ *wins* if it outputs a tuple (p_ℓ, y'), where p_ℓ is a Merkle path of length $\ell \leq n$ containing x_0 as a leaf, and y' is a second-preimage of p_ℓ.

In addition to requiring the ESPR assumption to hold, we will also assume that h is at least slightly regular, so that every output has at least two pre-images. This implies that, for any path p_ℓ there *exists* a second preimage and hence there is always an *inefficient* attack against ESPR security.

Definition 4.2 (Slightly Regular Hash)**.** We say that $h : \{\{0,1\}^{2n} \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ is *slightly regular*, if for every $z \in \{0,1\}^n$ there exist at least two distinct pre-images $y \neq y'$ such that $h(y) = h(y') = z$.

Discussion. In the above definition, we want h to be a single *fixed* hash function and not a function family. The notion of ESPR security seems to lie somewhere in between second-preimage resistance (SPR) and collision resistance (CR), implying the former and being implied by the latter.[9] Unfortunately, collision resistance *cannot* be achieved by any fixed hash function (at least w.r.t non-uniform attackers), since the attacker can always know a single hard-coded collision as auxiliary input. Fortunately, there does not appear to be any such trivial non-uniform attack against ESPR security, since the attacker is forced to "incorporate" a random leaf x_0 into the Merkle path on which it finds a collision. Therefore, in this regard, it seems that ESPR security may be closer to SPR security, which can be achieved by a fixed hash function (if one-way functions exist). Indeed, in Section 4.4, we give a heuristic argument that modern (fixed) cryptographic hash functions already satisfy the ESPR property, even against non-uniform attackers. We do so by analyzing ESPR security in a variant of the random-oracle model, where the attacker may observe some "oracle-dependent auxiliary input". This model, proposed by Unruh [33], is intended to capture the properties of hash functions that can be achieved by fixed hash functions, rather than function families.

4.2 A Counterexample to Hard Relations from ESPR

We begin by giving a counterexample for *hard relations* and later extend it to counterexamples for *one-way* functions and relations.

[9] A hash function is SPR if, given a uniformly random y, it's hard to find any $y' \neq y$ such that $h(y) = h(y')$. It is CR if it is hard to find any $y \neq y'$ s.t. $h(y) = h(y')$.

Definition 4.3 (ESPR Relation)**.** Given a hash function h we can define the NP relation R_h with *statements* $x \in \{0,1\}^n$ and *witnesses* $w = (p_\ell, y')$ where p_ℓ is a Merkle path of length $\ell \le n$ containing x as leaf, and y' is a second-preimage of p_ℓ. We define the distribution $x \leftarrow \textsc{SamL}_h(1^n)$ which just samples $x \xleftarrow{\$} \{0,1\}^n$ uniformly at random and call $\mathcal{R}_h = (R_h, \textsc{SamL}_h)$ the *ESPR relation.*

Note that the NP language corresponding to R_h is defined as $L_h \stackrel{\text{def}}{=} \{x : \exists\, w \text{ s.t. } (x,w) \in R_h\ \}$, and, if h is slightly regular, then $L_h = \{0,1\}^*$ is just the language consisting of all bit strings. Therefore $\textsc{SamL}_h$ does indeed sample from the language and $\mathcal{R}_h$ satisfies the syntax of a hard relation. Moreover, if h is an $(s(n), \varepsilon(n))$-hard ESPR hash function, then $\mathcal{R}_h = (R_h, \textsc{SamL})$ is an $(s(n), \varepsilon(n))$*-hard relation*, since the security games are equivalent.

We now show that hardness does *not* amplify for the relation $\mathcal{R}_h$. The main idea is that, given k random and independent challenges $x^{(1)}, \ldots, x^{(k)}$, the attacker builds a Merkle tree with the challenges as leaves. Let z be the value at the top of the Merkle tree. Then the attack just guesses some value $y' \in \{0,1\}^{2n}$ at random and, with probability $\ge 2^{-2n}$, y' will be a second-preimage of z (i.e. $h(y') = z$ and y' is distinct from the known preimage y containing the values under the root). Now, for each leaf $x^{(i)}$, let p^i_ℓ be the Merkle path for the leaf $x^{(i)}$. Then the witness $w_i = (y', p^i_\ell)$ is good witness for $x^{(i)}$. So, with probability $\ge 2^{-2n}$ with which the attack correctly guessed y', it breaks *all* k independent instances of the relation $\mathcal{R}_h$, no matter how large k is! By changing the relation $\mathcal{R}_h = (R_h, \textsc{SamL})$ so that, on security parameter n, the sampling algorithm $\textsc{SamL}(1^n)$ chooses $x \xleftarrow{\$} \{0,1\}^m$ with $m = m(n)$ being some smaller function of n such as $m(n) = n^\delta$ for a constant $\delta > 0$ or even $m(n) = \log^2(n)$, we can get more dramatic counterexamples where hardness does not amplify beyond $\varepsilon(n) = 2^{-n^\delta}$ or even $\varepsilon(n) = n^{-\log n}$. We now summarize the above discussion with a formal theorem.

Theorem 4.4. *Let h be a slightly regular, ESPR-secure hash function and let $\mathcal{R}_h = (R_h, \textsc{SamL})$ be the corresponding hard relation. Then, for any polynomial $k = \mathsf{poly}(n)$, the k-wise direct product of $\mathcal{R}_h$ is* not $(\mathsf{poly}, 2^{-2n})$ *secure. That is, there is a $\mathsf{poly}(k,n)$ attack against the k-wise direct product of $\mathcal{R}_h$ having success probability 2^{-2n}.*

Proof. We first describe the attack. The attacker gets k independently generated challenges $x^{(1)}, \ldots, x^{(k)}$. Let ℓ be the unique value such that $2^{\ell-1} < k \le 2^\ell$, and let $k^* = 2^\ell$ be the smallest power-of-2 which is larger than k. Let us define additional "dummy values" $x^{(k+1)} = \ldots = x^{(k^*)} := 0^n$. The attack constructs a *Merkle Tree*, which is a full binary tree of height ℓ, whose k^* leaves are associated with the values $x^{(1)}, \ldots, x^{(k^*)}$. The value of any non-leaf node v is defined recursively as $val(v) = h(val(v_L), val(v_R))$ where v_L, v_R are the left and right children of v respectively. For any leaf $v^{(i)}$ associated with the value $x^{(i)}$, let $(v_1 = v^{(i)}, v_2, \ldots, v_\ell, r)$ be the nodes on the path from the leaf v_1 to the root r in the Merkle tree. The Merkle path associated with the value $x^{(i)}$ is then defined by $p^{(i)}_\ell = (x^{(i)}, (x_1, b_1), \ldots, (x_\ell, b_\ell))$ where each x_j is the value associated with the *sibling* of v_j, and $b_j = 0$ if v_j is a right child and 1 otherwise. Note that, if r is the root of the tree and $z = val(r)$ is the value associated with it, then $\bar{h}(p^{(i)}_\ell) = z$ for all paths $p^{(i)}_\ell$ with $i \in \{1, \ldots, k^*\}$. Furthermore let us label the nodes v_L, v_R to be the children of the root r, the values x_L, x_R be the values associated with them, and set $y := (x_L, x_R)$. Then y is the known preimage such that $h(y) = z$, associated with each one of the paths $p^{(i)}_\ell$.

The attack guesses a value $y' \xleftarrow{\$} \{0,1\}^{2n}$ at random and, outputs the k-tuple of witnesses $(w_1, \ldots, w_k)$ where $w_i = (p_\ell^{(i)}, y')$. With probability at least 2^{-2n}, y' is a second-preimage of z with $h(y') = z$ and $y' \neq y$ (since h is slightly regular, such second preimage always exists). If this is the case, then y' is also a second preimage of *every* path $p_\ell^{(i)}$. Therefore, with probability $\geq 2^{-2n}$ the attack finds a witness for each of the k instances and wins the hard relation game for the direct product relation $\mathcal{R}_h^k$. □

Corollary 4.5. *Assuming the existence of a slightly regular* $(\mathsf{poly}, \mathsf{negl})$*-hard ESPR hash functions, the hardness of hard relations does* not *amplify to* $2^{-n^{\Omega(1)}}$*, giving a counterexample to the stronger dream conjecture for hard relations. If we instead assume the existence of* $(2^{\Omega(n)}, 2^{-\Omega(n)})$*-hard ESPR hash functions, then the hardness of hard relations does* not *amplify beyond negligible, giving a counterexample to the weaker dream conjecture.*

Proof. Follows directly by combining Theorem 2.5 and Theorem 4.4. □

4.3 Extension to One-Way Relations and One-Way Functions

One-Way Relations. We can get essentially the same results as above for *one-way* relations rather than just *hard* relations. Assume that $\mathcal{R}_{ow} = (R_{ow}, \textsc{SamR}_{ow})$ is *any one-way relation*, and let $\mathcal{R}_{hard} = (R_{hard}, \textsc{SamL}_{hard})$ be a *hard* relation. Define the OR relation $\mathcal{R}_{or} = (R_{or}, \textsc{SamR}_{or})$ via:

$$R_{or} \stackrel{\text{def}}{=} \{(y_1, y_2), (w_1, w_2) \ : \ (y_1, w_1) \in R_{hard} \text{ or } (y_2, w_2) \in R_{ow}\}$$
$$\textsc{SamR}_{or}(1^n) : \text{Sample } y_1 \leftarrow \textsc{SamL}_{hard}(1^n), (y_2, w_2) \leftarrow \textsc{SamR}_{ow}(1^n). \text{ Out: } ((y_1, y_2), (0, w_2)).$$

Then $\mathcal{R}_{or}$ is $(\mathsf{poly}, \mathsf{negl})$-hard as long as both $\mathcal{R}_{OW}$ and $\mathcal{R}_{hard}$ are. However, the k-wise direct product of R_{or} is *no harder* than that of $\mathcal{R}_{hard}$. Therefore the results of Corollary 4.5 translate directly to *one-way* relations as well.

One-Way Functions. Let $i(n) \geq n$ be a polynomial and $f \ : \ \{\{0,1\}^{i(n)} \to \{0,1\}^n \ \}_{n \in \mathbb{N}}$ be a *one-way function*. Let $\mathcal{R} = (R, \textsc{SamL})$ be a hard relation where the sampling algorithm $\textsc{SamL}(1^n)$ induces the same distribution over $\{0,1\}^n$ as the distribution $\{f(x) \ : \ x \xleftarrow{\$} \{0,1\}^{i(n)}\}$. Assume that for $y \in \{0,1\}^n$, and witness w such that $(y, w) \in R$, the witness-size is bounded by $|w| = u(n)$. We define the counterexample one-way function

$$F \ : \ \{0,1\}^{i(n)} \times \{0,1\}^n \times \{0,1\}^{u(n)} \times \{0,1\}^{4n} \to \{0,1\}^n$$

via:

$$F(x, y, w, z) \stackrel{\text{def}}{=} \begin{cases} y & \text{If } (y, w) \in R \wedge z = 0^{4n} \\ f(x) & \text{Otherwise.} \end{cases}$$

Note that the distribution of $F(x, y, w, z)$ is statistically close to that of $f(x)$ since the probability of $z = 0^{4n}$ is negligible.[10] The preimage of any $y \in \{0,1\}^n$ is either of the form $(\cdot, y, w, \cdot)$ where $(y, w) \in R$ or of the form $(x, \cdot, \cdot, \cdot)$ where $f(x) = y$, and hence breaking the one-wayness of F is no easier then breaking that of f or breaking the hard relation $\mathcal{R}$.

[10] This is the only reason we include the input z.

Theorem 4.6. *Assume that f is a one-way function and $\mathcal{R}$ a hard relation with matching output distributions as above, and that both are $(s(n), \varepsilon(n))$-hard. Then the function F is a $(s(n), 2\varepsilon(n) + 2^{-4n})$-hard one-way function. On the other hand, if there is a $\mathsf{poly}(k, n)$ attack on the k-wise direct product $\mathcal{R}^k$ of the relation $\mathcal{R}$ with success probability $\varepsilon(n)$, than there is also a $\mathsf{poly}(k, n)$ attack on the k-wise direct product F^k of the one-way function F with success probability $\geq \varepsilon(n) - 2^{-3n}$.*

Proof. Assume that $\mathcal{A}$ is of size $s(n)$ and has probability $\varepsilon_F(n)$ in inverting F. When (x, y, w, z) is chosen at random, the probability that $z = 0^{4n}$ is just 2^{-4n} and hence the distribution of $F(x, y, w, z)$ is (2^{-4n})-close to that of $f(x)$. So, we have $\Pr[F(\mathcal{A}(y)) = y \mid x \xleftarrow{\$} \{0,1\}^n, y = f(x)] \geq \varepsilon_F(n) - 2^{-4n}$. For $(x', y', w, z) = \mathcal{A}(y)$, let W_{func} be the event that with $f(x') = y$ and let W_{rel} be the event that $(y, w) \in R$. Then $2\varepsilon(n) \geq \Pr[W_{func}] + \Pr[W_{rel}] \geq \varepsilon_F(n) - 2^{-4n}$, which concludes the first part of the theorem.

For the second part of the theorem, assume that there is an attack $\mathcal{A}$ on $\mathcal{R}^k$. That is, given $(y_1, \dots, y_k) \xleftarrow{\$} \textsc{SamL}^k(1^n)$, and $(w_1, \dots, w_k) = \mathcal{A}(y_1, \dots, y_k)$ we get $(y_1, w_1), \dots, (y_k, w_k) \in R$ with probability $\geq \varepsilon'(n)$. Then, $\mathcal{A}(y_1, \dots, y_k)$ also satisfies this condition when we choose $\{y_i \xleftarrow{\$} F(x_i, y_i', w_i, z_i)\}_{i=1}^k$ with probability $\geq \varepsilon'(n) - k2^{-4n} \geq \varepsilon'(n) - 2^{-3n}$ since the two distributions are $(k2^{-4n})$-statistically-close. Therefore, to invert F^k on output $(y_1, \dots, y_k)$ we can just run $(w_1, \dots, w_k) = \mathcal{A}(y_1, \dots, y_k)$ and output the k pre-images of the form $(0^{i(n)}, y_i, w_i, 0^n)$. □

The only challenge left is to instantiate the one-way function f and the relation $\mathcal{R}$. We would like to use the counterexample hard-relation $\mathcal{R}_h$ from Theorem 4.4 (or the variant using smaller instances as in Corollary 4.5). In this case, we seem to need f to be a *regular* one-way function so that the distribution of $f(x)$ is just uniformly random n-bit strings.[11] We summarize this below.

Corollary 4.7. *Assume that there exists a regular one-way function $f : \{\{0,1\}^{i(n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ and slightly regular ESPR hash function $h : \{\{0,1\}^{2n} \to \{0,1\}^n\}_{n \in \mathbb{N}}$, both of which are $(\mathsf{poly}, \mathsf{negl})$-hard. Then the hardness of one-way functions does not amplify beyond $2^{-\Omega(n)}$, giving a counterexample to the stronger dream conjecture for one-way functions. If f and h are both $(2^{\Omega(n)}, 2^{-\Omega(n)})$-hard then the hardness of one-way functions does not amplify beyond negligible, giving a counterexample to the weaker dream conjecture.*

Proof. Using Theorem 4.4, the k-wise direct product $\mathcal{R}_h^k$ of $\mathcal{R}_h$ has $\mathsf{poly}(k, n)$ sized attack with success probability 2^{-2n}. Using the second part of Theorem 4.6, this means that F^k has a $\mathsf{poly}(k, n)$ sized attack with success probability $2^{-2n} - 2^{-3n} \geq 2^{-3n}$. On the other hand, by the first part of Theorem 4.6, F is $(\mathsf{poly}, \mathsf{negl})$-hard if both $\mathcal{R}_h, f$ are and is $(2^{\Omega(n)}, 2^{-\Omega(n)}$-hard if both $\mathcal{R}_h, f$ are. The corollary then follows by applying Theorem 2.5 to the one-way function F. □

It turns out that we do not need to make an additional assumption about regular one-way functions f and instead directly use the ESPR hash function h in place of f. Firstly, we need to show that ESPR security implies one-wayness, which is very similar to the proof that SPR (second-preimage resistance) implies one-wayness. Secondly, we need to show that the relation $\mathcal{R}_h$ used in our counterexample remains hard even when the instance x is sampled as $h(x_1, x_2)$ instead of uniformly at random.

[11] A function is *regular* if the every output has the same number of preimages.

Lemma 4.8. *Let $h : \{0,1\}^{2n} \to \{0,1\}^n$ be an $(s(n), \varepsilon(n))$-hard ESPR hash function. Then h is also an $(s(n), \varepsilon(n) + 2^{-n})$-hard one-way function.*

Proof. Assume that there is an attack $\mathcal{A}$ of size $s(n)$ which breaks one-wayness of h with probability $\varepsilon'(n)$. Let us consider the experiment where we sample $y = (x_0, x_1) \xleftarrow{\$} \{0,1\}^{2n}$, $z = h(y)$, $y' = \mathcal{A}(z)$. Let W be the event that $h(y') = z$ and E be the event that $y' = y$. By assumption $\Pr[W] = \varepsilon'(n)$. On the other hand $\Pr[E] \le 2^{-n}$ since $\mathcal{A}$ must guess a random $2n$ bit value y given only n bits of information z about it. Therefore $\Pr[W \wedge \neg E] \ge \varepsilon'(n) - 2^{-n}$.

Let $\mathcal{A}'(x_0)$ be an attacker which samples $x_1 \xleftarrow{\$} \{0,1\}^n$, sets $y = (x_0, x_1)$, $z = h(y)$ and calls $\mathcal{A}(z)$. Let $p_1 = (x_0, x_1)$ be a Merkle path of length 1 which evaluates to z and whose known preimage is $y = (x_0, x_1)$. If the event $W \wedge \neg E$ occurs, than y' is a second preimage of the path p_1 starting at the challenge leaf x_0 and therefore $\mathcal{A}$ breaks ESPR security. This proves the lemma. □

Lemma 4.9. *Let $h : \{0,1\}^{2n} \to \{0,1\}^n$ be an $(s(n), \varepsilon(n))$-hard ESPR hash function. Let $\mathcal{R}'_h = (R'_h, \textsc{SamL}'_h)$ be just like the hard relation from our counterexample, with the modification that the distribution $\textsc{SamL}'_h(1^n)$ samples $x' = h(x_L, x_R)$ where $(x_L, x_R) \xleftarrow{\$} \{0,1\}^{2n}$ instead of sampling x' as a uniformly random n-bit value. Then there is some polynomial $p(n)$ such that $\mathcal{R}'_h$ is $(s(n) - p(n), \varepsilon(n))$-hard.*

Proof. Assume that $\mathcal{A}(x')$ is an attack against the relation $\mathcal{R}'_h$. Then we can construct an attack $\mathcal{A}'(x)$ against the ESPR security of h as follows. On random challenge x, the attack $\mathcal{A}'(x)$ sets $x_L = x$ and samples $x_R \xleftarrow{\$} \{0,1\}^n$. It computes $x' = h(x_L, x_R)$ and calls $\mathcal{A}(x')$ which outputs a Merkle path $p_\ell = (x', (x_1, b_1) \ldots, (x_\ell, b_\ell))$ of size $\ell \le n-1$ along with a second preimage y' of p_ℓ. The attack $\mathcal{A}'$ then outputs the path $p'_{\ell+1} = (x_L, (x_R, 1), (x_1, b_1) \ldots, (x_\ell, b_\ell))$ and the same preimage y'. Note that, if y' is a second preimage of p_ℓ, it is also a second preimage of $p'_{\ell+1}$. Therefore, the success probability of $\mathcal{A}'$ against the ESPR security of h is the same as that of $\mathcal{A}$ against the hardness of $\mathcal{R}'_h$. □

Putting Lemmas 4.8 and 4.9 together, we get a one-way function h and a relation $\mathcal{R}'_h$ that meet the requirements of Theorem 4.6 under just the ESPR assumption. Namely,the output distribution of h matches the distribution of instances on which $\mathcal{R}'_h$ is hard. Therefore, by applying Theorem 2.5, we can get the same results as in Corollary 4.7 under just the ESPR assumption.

Corollary 4.10. *If there exists a slightly regular ESPR hash function $h : \{\{0,1\}^{2n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ which is $(\mathsf{poly}, \mathsf{negl})$-hard, then the hardness of one-way functions does not amplify beyond $2^{-\Omega(n)}$, giving a counterexample to the stronger dream conjecture for one-way functions. If h is $(2^{\Omega(n)}, 2^{-\Omega(n)})$-hard then the hardness of one-way functions does not amplify beyond negligible, giving a counterexample to the weaker dream conjecture.*

4.4 Justifying the ESPR Assumption

We now give some justification that ESPR hash functions may exist by showing how to construct them in in a variant of the random-oracle (RO) model. Of course, constructions in the random-oracle model do not seem to offer any meaningful guarantees for showing that the corresponding primitive may be realized by a *fixed* hash function: indeed the RO model immediately implies collision resistance which cannot be realized by a fixed hash function. Rather, the RO model is usually interpreted as implying that the given primitive is likely to be realizable by a *family of hash functions*. Therefore, we will work with a variant of the RO model in which the attacker is initialized with some arbitrary "oracle-dependent auxiliary input". This model was proposed by Unruh [33] with the explicit motivation that:

> "... oracle-dependent auxiliary input provides a tool for distinguishing the cases where the use of a single function is sufficient (e.g., in the case where we require only one-wayness) and where a keyed family of functions is necessary (e.g., in the case that we require collision-resistance)."

For example, the auxiliary input may include some small number of fixed collisions on the RO and therefore collision-resistance is unachievable in this model. By showing that ESPR security *is* achievable, we provide some justification for this assumption.

Oracle-Dependent Auxiliary Input. Let $\mathcal{O} : \{0,1\}^{2n} \mapsto \{0,1\}^n$ be a fixed length random oracle. We define "oracle-dependent auxiliary input" of size $p(n)$ as an arbitrary function $z : \{\{0,1\}^{2n} \mapsto \{0,1\}^n\} \mapsto \{0,1\}^{p(n)}$ which can arbitrarily "compresses" the entire oracle $\mathcal{O}$ into $p(n)$ bits of auxiliary information $z(\mathcal{O})$. When considering security games in the oracle-dependent auxiliary input model, we consider attackers $\mathcal{A}^{\mathcal{O}}(z(\mathcal{O}))$ which are initialized with polynomial-sized oracle-dependent auxiliary input $z(\cdot)$.

In Appendix C, we prove the following theorem.

Theorem 4.11. *Let $\mathcal{O}$ be modeled as a random oracle, and consider the ESPR game in which h is replaced with $\mathcal{O}$. Then, for any attacker $\mathcal{A}^{\mathcal{O}}(z(\mathcal{O}))$ with polynomial-sized auxiliary input $z(\cdot)$ and making at most polynomially many queries to $\mathcal{O}$, its probability of winning the ESPR game is at most $\varepsilon = 2^{-\Omega(n)}$.*

5 Conclusion

In this work, we provide coutnerexamples to hardness amplification beyond negligible for signatures and one-way functions. This raises several interesting open problems for future research. Firstly, it would be very interesting to get a counterexample for one-way functions under more standard assumptions, rather than assuming extended second-preimage resistance (ESPR). Another interesting problem would be to find similar counterexamples for the XOR lemma rather than for direct-product theorems. Lastly, since our counterexamples are highly contrived, it would be interesting to identify some abstract properties that allow them to go through, and attempt to form a more targeted conjecture on hardness amplification beyond negligible for schemes which do not have these properties. In particular, despite our counterexamples, we would tend to think that hardness *should* amplify exponentially with the number of repetitions for most natural primitives.

References

[1] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[2] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[3] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[4] M. Bellare. A note on negligible functions. *J. Cryptology*, 15(4):271–284, 2002.

[5] M. Bellare, R. Impagliazzo, and M. Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS*, pages 374–383, 1997.

[6] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[7] R. Canetti, S. Halevi, and M. Steiner. Hardness amplification of weakly verifiable puzzles. In *TCC*, pages 17–33, 2005.

[8] R. Canetti, R. L. Rivest, M. Sudan, L. Trevisan, S. P. Vadhan, and H. Wee. Amplifying collision resistance: A complexity-theoretic treatment. In *CRYPTO*, pages 264–283, 2007.

[9] K.-M. Chung, F.-H. Liu, C.-J. Lu, and B.-Y. Yang. Efficient string-commitment from weak bit-commitment. In *ASIACRYPT*, pages 268–282, 2010.

[10] Y. Deng, V. Goyal, and A. Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[11] Y. Dodis, R. Impagliazzo, R. Jaiswal, and V. Kabanets. Security amplification for interactive cryptographic primitives. In *TCC*, pages 128–145, 2009.

[12] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *EUROCRYPT*, pages 342–360, 2004.

[13] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[14] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, pages 218–229, 1987.

[15] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(50), 1995.

[16] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[17] V. Goyal and H. K. Maji. Stateless cryptographic protocols. In *FOCS*, 2011.

[18] V. Goyal and A. Sahai. Resettably secure computation. In *EUROCRYPT*, pages 54–71, 2009.

[19] I. Haitner. A parallel repetition theorem for any interactive argument. In *FOCS*, pages 241–250, 2009.

[20] S. Halevi and T. Rabin. Degradation and amplification of computational hardness. In *TCC*, pages 626–643, 2008.

[21] J. Håstad, R. Pass, D. Wikström, and K. Pietrzak. An efficient parallel repetition theorem. In *TCC*, pages 1–18, 2010.

[22] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Chernoff-type direct product theorems. *Journal of Cryptology*, 2008. (published online September 2008); preliminary version in CRYPTO'07.

[23] A. Jain and K. Pietrzak. Parallel repetition for leakage resilience amplification revisited. In *TCC*, pages 58–69, 2011.

[24] C. S. Jutla. Almost optimal bounds for direct product threshold theorem. In *TCC*, pages 37–51, 2010.

[25] A. Lewko and B. Waters. On the insecurity of parallel repetition for leakage resilience. In *FOCS*, pages 521–530, 2010.

[26] M. Luby and C. Rackoff. Pseudo-random permutation generators and cryptographic composition. In *STOC*, pages 356–363, 1986.

[27] U. M. Maurer and S. Tessaro. Computational indistinguishability amplification: Tight product theorems for system composition. In *CRYPTO*, pages 355–373, 2009.

[28] U. M. Maurer and S. Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak prgs with optimal stretch. In *TCC*, pages 237–254, 2010.

[29] M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *J. of Cryptology*, 12:29–66, 1999. Preliminary version in: *Proc. STOC 97*.

[30] R. Pass and M. Venkitasubramaniam. An efficient parallel repetition theorem for Arthur-Merlin games. In *STOC*, pages 420–429, 2007.

[31] K. Pietrzak and D. Wikstrom. Parallel repetition of computationally sound protocols revisited. In *TCC*, pages 86–102, 2007.

[32] S. Tessaro. Security amplification for the cascade of arbitrarily weak prps: Tight bounds via the interactive hardcore lemma. In *TCC*, pages 37–54, 2011.

[33] D. Unruh. Random oracles and auxiliary input. In *CRYPTO*, pages 205–223, 2007.

[34] A. C.-C. Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.

A Stateless Multiparty Computation

In our counter-example for signature schemes given in Section 3, we use stateless multi-party computation as a core building block. Essentially, we consider multi-party computation in the setting where the parties are stateless devices, i.e., each party only supports a "request-reply" interaction (i.e., the party just outputs $f(x)$ when fed with x for some fixed $f(x)$). Secure MPC protocols for this setting were recently given by Goyal and Maji [17]. In order to construct such protocols, [17] first define *resettably-secure* multiparty computation and give a protocol satisfying their notion. They then discuss a transformation from a resettably-secure protocol into a stateless one using standard techniques.[12]

For our purposes, we find it more convenient to directly define the notion of stateless multiparty computation. In order to do so, we turn to the standard real/ideal paradigm. Very briefly, the ideal model is identical to the ideal model in the definition of resettably-secure computation [17], where the adversary is allowed to reset the ideal functionality. The real world is also defined in a manner similar to [17], except that we consider protocol execution between stateless parties instead of stateful, resettable parties. Before we proceed to formally describe the real and ideal world executions, we first describe the model of protocol execution that we will consider for stateless parties.

Protocol execution in the public state model. Let Π_k be a protocol for k stateful parties $P_1, \ldots, P_k$. For simplicity, we assume that protocol Π_k proceeds in rounds and that in any round j, each party P_i' sends a protocol message. We note that this is without loss of generality since we can require parties to send "empty" messages. We further assume that the messages in Π_k are sent over the broadcast channel. We note that in the construction of our counter-example for signature schemes in Section 3, we will only be interested in the case where an adversary corrupts exactly $k-1$ parties in the protocol Π_k. In this case, the above requirement is automatically satisfied. We now explain how the protocol Π_k can be executed when $P_1, \ldots, P_k$ are *stateless* machines.

We consider execution in the *public state model* described as follows. Let $\mathcal{A}$ denote the adversary. For each party P_i, let x_i, r_i denote its input and (fixed) random tape respectively. We consider a state variable state_i that contains the "current" protocol state of party P_i (which is a function of its input x_i, random tape r_i and the protocol messages received so far) at any point during the protocol execution. We assume that state_i is a *public* string and is available to $\mathcal{A}$. Note that state_i can be made public by using standard techniques: suppose that each party P_i has a secret key s_i of an authenticated encryption scheme; then state_i is defined as the authenticated encryption (w.r.t. key s_i) of the actual protocol state of P_i. Without loss of generality, we assume that s_i is contained in r_i. Further note that each state_i is initialized to the "empty" string.

In order to describe how $P_1, \ldots, P_k$ execute the protocol, we need to define the *next message function* for each party P_i. The next message function NM_i for party P_i takes the following four values as input: (a) a string π_{j-1} that consists of all the messages sent in any round $j-1$ of the protocol, (b) the public state state_i of party P_i, (c) the input x_i, and (d) the (fixed) random tape r_i. On receiving an input of the form $\pi_{j-1} \| \mathsf{state}_i \| x_i \| r_i$, NM_i outputs P_i's message in round j along with the updated value of state_i. For notation convenience, we define π_0 to be the "empty" string.

Now, note that at any point, k public strings $\mathsf{state}_1, \ldots, \mathsf{state}_k$ are available to $\mathcal{A}$. Then, to execute the j^{th} round of the protocol, the adversary $\mathcal{A}$ sends the string π_{j-1} (consisting of all the messages from round $j-1$) and state_i to each party P_i. On receiving these values,

[12] Very briefly, the transformation works as follows. For simplicity, we only consider the two-party case. Let $\Pi' = \{P_1', P_2'\}$ be a resettably-secure protocol. The protocol Π for stateless parties P_1 and P_2 is defined as follows. Each party P_i has a secret key of an authenticated encryption scheme. Party P_1 computes the first protocol message in the same manner as P_1'; however, it sends to P_2 not only the computed message but also an authenticated encryption of its current protocol state. Party P_2 computes the reply in the same manner as P_2' and sends to P_1 not just the computed reply but also (a) the received encrypted state of P_1, and, (b) an authenticated encryption of the current state of P_2 using its own key. Thus, in each subsequent round, each P_i "recover" its state from the previous round, and use it to compute the next outgoing message.

P_i applies the next message function NM_i over $\pi_{j-1} \| \mathsf{state}_i \| x_i \| r_i$ to compute its outgoing message in round j and the updated value of state_i.

This completes our discussion on protocol execution in the public state model. We now formally define the ideal and real world experiments and then give the security definition for stateless MPC. Some of the text below is taken from [17].

Ideal World. We first define the ideal world experiment, where k parties interact with an ideal functionality for computing a randomized function $\mathcal{F}$. The execution in the ideal world starts by the adversary $\mathcal{S}$ (who is given auxiliary input z) selecting an arbitrary subset $I \subset \{P_1, \ldots, P_k\}$ of parties to corrupt. The adversary also defines a number $\mathsf{in}_i \in \mathsf{poly}(n)$ for each party which defines the total number of (possible) incarnations for each party P_i. Each party P_i receives an input vector $x^{(i)} = \{x_1^{(i)}, \ldots, x_{\mathsf{in}_i}^{(i)}\}$ consisting of in_i inputs, one for each incarnation. From here onwards, the ideal world execution proceeds as follows:

Select incarnation: The adversary $\mathcal{S}$ selects the incarnation for each of the honest parties. The trusted party forwards each honest party P_i their respective incarnation index.

Send inputs to trusted party: Each honest party P_i sends its input $x_\ell^{(i)}$ (where ℓ is the incarnation index for P_i chosen by $\mathcal{S}$) to the trusted party. For each corrupted party $P_i \in I$, $\mathcal{S}$ may select any input value and send it to the trusted party.

The trusted party computes the function: Let $x'_1, \ldots, x'_k$ be the inputs that were sent to the trusted party. If $\mathcal{F}$ is a randomized functionality, then the trusted party samples fresh randomness r and computes $y = \mathcal{F}(x'_1, \ldots, x'_k; r)$, else it computes $y = \mathcal{F}(x'_1, \ldots, x'_k)$.

Adversary learns output: The trusted party sends y to $\mathcal{S}$.

Honest parties learn output: $\mathcal{S}$ prepares a list of honest parties who should learn the output. The trusted party forwards y to all parties whom $\mathcal{S}$ intends to receive their output. The remaining honest parties receive $\perp$.

Reset: The adversary can reset the ideal world at any point of time. When $\mathcal{S}$ decides to reset the ideal world, it requests the trusted party to reset all honest parties and the trusted party sends a reset signal to all honest parties. At this point, the ideal world returns to the first stage where the adversary can select an incarnation for all parties.

Outputs: Each honest party outputs the message that it received from the trusted party. $\mathcal{S}$ outputs an arbitrary PPT function of its entire view.

The output of the ideal world execution consists of the outputs of the honest parties along with the output of the adversary $\mathcal{S}$. We denote it by $\text{IDEAL}^{\mathcal{F}}_{\mathcal{S},I}(1^n, \vec{X}, z)$, where n is the security parameter and $\vec{X} = x^1, \ldots, x^k$.

Real World. The real-world execution begins by an adversary $\mathcal{A}$ (with auxiliary input z) selecting any arbitrary subset $I \subset \{P_1, \ldots, P_k\}$ of the parties to corrupt. $\mathcal{A}$ also specifies the possible incarnations for each party P_i. The ℓ^{th} incarnation of a party P_i is defined by its input $x_\ell^{(i)}$ and random tape $r_\ell^{(i)}$.

To the start the protocol execution, the adversary chooses the initial incarnation ℓ_i for each party P_i. This fixes the random tape of each party P_i to $r_{\ell_i}^{(i)}$. Now, parties $P_1, \ldots, P_k$

(with their respective incarnations $\ell_1, \ldots, \ell_k$) begin to execute the k-party protocol Π_k in the public state model in the manner as discussed earlier. The adversary $\mathcal{A}$ sends all messages on behalf of the corrupted parties, and may follow an arbitrary polynomial-time strategy. In contrast, the honest parties follow the instructions of Π_k. The adversary $\mathcal{A}$ can reset any honest party P_i at any point of time during the execution of the protocol. $\mathcal{A}$ may change P_i's incarnation to ℓ'_i, in which case P_i uses the independently chosen random tape $r^{(i)}_{\ell'_i}$. Otherwise, P_i may be reset to an earlier state of the same incarnation ℓ_i; in this case P_i reuses the same random tape $r^{(i)}_{\ell_i}$.

At the conclusion of the protocol execution, each honest party P_i, $i \notin I$ reports its output as instructed by Π_k. The adversary may output an arbitrary PPT function of its view. The overall output of the real-world experiment is defined as the output vector of the honest parties and $\mathcal{A}$. We denote it by $\text{REAL}^{\Pi_k}_{\mathcal{A},I}(1^n, \vec{X}, z)$.

Security Definition. Having defined the ideal and real world experiments, we now give the formal definition of stateless MPC. Roughly, we say that a protocol Π_k securely computes a functionality $\mathcal{F}$ if for every (resetting) adversary $\mathcal{A}$ in the real world, there exists a (resetting) simulator $\mathcal{S}$ in the ideal world who can simulate the view of $\mathcal{A}$.

Definition A.1 (Stateless MPC)**.** A protocol Π_k for stateless parties $P_1, \ldots, P_k$ *securely computes* a functionality f if for every PPT real adversary $\mathcal{A}$, there exists an (expected) PPT ideal adversary $\mathcal{S}$ such that for every $z \in \{0,1\}^*$, input vector $\vec{X}$, and $I \subset \{P_1, \ldots, P_k\}$, it holds that

$$\{\text{IDEAL}^{\mathcal{F}}_{\mathcal{S},I}(1^n, \vec{X}, z)\}_{n\in\mathbb{N}} \stackrel{c}{\equiv} \{\text{REAL}^{\Pi_k}_{\mathcal{A},I}(1^n, \vec{X}, z)\}_{k\in\mathbb{N}}.$$

In this paper, we will use the following corollary of the main result in [17]. We refer the reader to [17] for more details.

Corollary A.2 ([17])**.** *Assuming semi-honest oblivious transfer, collision-resistant hash functions, zaps, and a lossy encryption scheme, there exists a stateless MPC protocol for every randomized functionality that does not take any inputs. Further, the number of resets in the ideal world and the real world are equal.*

Additional Properties. For our purposes, we will require the following additional properties from Π_k:

Output delivery round: We require that the final round of protocol Π_k is the *output delivery round*, i.e., each party P_i can compute the protocol output (which may be a valid output or simply $\perp$) from all the messages sent in the final round, and its own protocol state. We note that this property can be easily enforced in the GMW protocol [14] by first having the parties commit to their output shares (and proving correctness via zero-knowledge); then in the final round, each party decommits (via a *single* message) to its output share. Thus, on receiving all the decommitments, each party can compute the correct output (or reject). Then, instantiating the stateless MPC protocol of Goyal and Maji [17] with semi-honest GMW and then applying the same idea as above yields a stateless MPC protocol with the desired property.

Now recall the next message function NM_i of party P_i. Since (from the above assumption) the last round of Π_k is the output delivery round, when NM_i is computed over the protocol messages from the last round of Π_k (and all the other necessary information; see Section A), we define NM_i to simply output the protocol output computed by P_i.

Publicly computable output: For simplicity we will restrict ourselves to functionalities where all parties receive a common output. We will assume that the output of protocol Π_k can be determined "publicly" from the transcript of the output delivery round. Note that this property is satisfied by the GMW protocol; thus the protocol of Goyal and Maji instantiated with semi-honest GMW inherits this property as well.

B Security of the Signature Scheme

Proof Sketch. We will prove Theorem 3.1 by contradiction. Specifically, we will show that given a PPT adversary $\mathcal{A}$ that forges signatures for the signature scheme $(\textsc{Gen}, \textsc{Sign}, \textsc{Verify})$ with non-negligible probability δ, we can construct a PPT adversary $\mathcal{B}$ that forges signatures for the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with non-negligible probability $\delta' = \delta - \mathsf{negl}(n)$.

Description of $\mathcal{B}$. Let C denote the challenger for the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. Very roughly, $\mathcal{B}$ works by internally running the adversary $\mathcal{A}$; $\mathcal{B}$ answers $\mathcal{A}$'s (signing) queries by using the responses (to its own queries) from C, and then outputs the signature forgery output by $\mathcal{A}$. We now give more details.

On receiving a public key pk from C, $\mathcal{A}$ samples a random tape $K \leftarrow \{0,1\}^{\mathsf{poly}(n)}$ and a random identity $\mathsf{id} \in \mathsf{poly}(n)$, and sends $PK = (pk, \mathsf{id})$ as the public key to $\mathcal{A}$. Now, whenever $\mathcal{A}$ sends a signing query m, $\mathcal{B}$ does the following:

1. $\mathcal{B}$ forwards the (signing) query m to C and obtains a signature σ_1 on m.
2. Further, using identity id and random tape K, $\mathcal{B}$ computes σ_2 in the same manner as the honest signing algorithm $\textsc{Sign}$, except that whenever $\textsc{Sign}$ is forced to output the secret key, $\mathcal{B}$ outputs $\perp$.

   More specifically, recall that on receiving a signing query of the form "$\mathtt{prot}$"$\|\mathsf{IM}\|\pi_j\|\mathsf{state}\|w$, where $\mathsf{IM} = k\|\mathsf{id}_1\|\ldots\|\mathsf{id}_k$ and $\mathsf{state} = \mathsf{state}_1\|\ldots\|\mathsf{state}_k$, $\textsc{Sign}$ plays the role of party P_i in protocol Π_k and computes σ_2 as P_i's outgoing message in round $j+1$ of Π_k by applying the next message function NM_i over π_j and state_i. Further recall that when j corresponds to the final round of Π_k, then $\textsc{Sign}$ computes the protocol output y and if $(y, w) \in \mathcal{R}$, then it output σ_2 as the secret key. Then, $\mathcal{B}$ computes σ_2 in the same manner as $\textsc{Sign}$, except that when j corresponds to the final round of Π_k, if $(y, w) \in \mathcal{R}$, $\mathcal{B}$ simply outputs $\perp$.

Finally, when $\mathcal{A}$ outputs a forgery (m^*, σ^*), $\mathcal{B}$ outputs (m^*, σ^*) and stops. This completes the description of $\mathcal{B}$.

Now, let ε be the probability that $\mathcal{B}$ outputs the abort symbol. Then, it follows immediately from the above description that $\mathcal{B}$ outputs a valid forgery for $(\textsc{Gen}, \textsc{Sign}, \textsc{Verify})$ with probability $\delta' = \delta - \varepsilon$. We now argue that $\varepsilon = \mathsf{negl}(n)$.

Let q be the total number of queries that $\mathcal{A}$ makes. Then, it follows that there are at most q different "initialization messages" IM_j sent by $\mathcal{A}$ (as part of its signing queries). Let $m \leq q$ be the total IM's, denoted by $\mathsf{IM}_1, \ldots, \mathsf{IM}_m$. As a first step, instead of using $PRF_K(\cdot)$ to generate the random tape for the protocol party (say) P_i played by $\mathcal{B}$ corresponding to any instance of $\mathsf{IM}_j = k_j\|\mathsf{id}_1^j\|\ldots\|\mathsf{id}_{k_j}^j$, we now simply pick a fresh random tape. It follows from the security of the PRF family that the view of $\mathcal{A}$ in this experiment is indistinguishable from the previous one.

Note that if the total probability of abort is ε, then there must exist an IM_j such that $\mathcal{A}$ causes an abort for this IM_j with probability at least $\frac{\varepsilon}{m}$. From the previous step, we have that each $\mathsf{IM}_j = k_j \| \mathsf{id}_1^j \| \ldots \| \mathsf{id}_{k_j}^j$ corresponds to an *independent* execution of Π_{k_j} (under reset attacks by $\mathcal{A}$). We now pick an $\mathsf{IM}_j = k_j \| \mathsf{id}_1^j \| \ldots \| \mathsf{id}_{k_j}^j$ at random; with probability at least $\frac{1}{m}$, $\mathcal{A}$ must cause an abort in this instance with probability at least $\frac{\varepsilon}{m}$.

Now, suppose that ε is non-negligible in n. Then, since $m \in \mathsf{poly}(n)$, we have that $\mathcal{A}$ causes an abort in the instance IM_j with non-negligible probability. We will show how to break the security of the hard relation $\mathcal{R}$ with non-negligible probability, which is a contradiction. To see this, note that if $\mathcal{A}$ causes an abort for this instance of IM_j, then there must exist a witness $w^* \in \mathcal{R}(y)$ in the view of $\mathcal{A}$, where y is the protocol output. By the security of the resettably-secure protocol Π_{k_j}, it follows that there exists a simulator $\mathcal{S}$ that produces an indistinguishable view containing a $w^* \in \mathcal{R}(y)$. We can thus simply take y from an external party, and run $\mathcal{S}$ to find a witness $w^* \in \mathcal{R}(y)$, which contradicts the security of the hard relation $\mathcal{R}$.

C ESPR in Random Oracle Model with Auxiliary Input

We begin with some tools from the work of [33], which make it very easy to prove theorems in this model.

Definition C.1 (Pre-Sampling [33])**.** For a set $S = \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq \{0,1\}^{2n} \times \{0,1\}^n$ with x_i distinct, we define the *random-oracle with pre-sampling* S as $\mathcal{P}[S] : \{0,1\}^{2n} \mapsto \{0,1\}^n$ which, on query x, checks if there exists some $(x, y) \in S$ and if so returns y. If x has already been queried, the same answer is given again. Otherwise, a uniformly random element $y \xleftarrow{\$} \{0,1\}^{2n}$ is given.

The following lemma follows from the main theorem of [33] and shows that giving an attacker oracle-dependent auxiliary input on the RO $\mathcal{O}$ is no worse than using some RO $\mathcal{P}[S]$ with a pre-sampling S of some slightly super-polynomial size.

Lemma C.2. *Let $\mathcal{C}^{\mathcal{O}}$ be the challenger for a game in the random oracle model and let $\mathcal{A}^{\mathcal{O}}(z(\mathcal{O}))$ be an attacker with $p = p(n)$-sized auxiliary input. Assume $\mathcal{C}, \mathcal{A}$ together make at most $q = q(n)$ queries to the RO and* $\Pr\left[\mathcal{A}^{\mathcal{O}}(z(\mathcal{O})) \text{ wins } \mathcal{C}^{\mathcal{O}}\right] \geq \varepsilon$*. Then, for any $s = s(n)$, there exists some $\mathcal{B}$ making at most q RO queries and some set S of size s such that* $\Pr\left[\mathcal{B}^{\mathcal{P}[S]} \text{ wins } \mathcal{C}^{\mathcal{P}[S]}\right] \geq \varepsilon - \sqrt{\frac{pq}{2s}}$.

(More directly, the main theorem from [33] just says that there exists some pre-sampling $S(\mathcal{O})$, which depends on $\mathcal{O}$, such that $\Pr[\mathcal{A}^{\mathcal{P}[S(\mathcal{O})]}(z(\mathcal{O})) \text{ wins } \mathcal{C}^{\mathcal{P}[S(\mathcal{O})]}] \geq \varepsilon - \sqrt{\frac{pq}{2s}}$. Our version then follows by fixing the best choice of $\mathcal{B} = \mathcal{A}(z(\mathcal{O})), S = S(\mathcal{O})$.)

Given the above tools, we are ready to prove Theorem 4.11, showing that the ESPR assumption holds in the random oracle model with auxiliary input.

Proof of Theorem 4.11. First assume that there is an attacker $\mathcal{A}$ with $p = p(n)$-sized auxiliary input $z(\cdot)$ and making $q = q(n)$ queries to the RO for which the probability of $\mathcal{A}^{\mathcal{O}}(z(\mathcal{O}))$ winning the ESPR game is ε. Then, by applying Lemma C.2, we get that there is an attacker $\mathcal{B}$ and some set S of size $s = 2^{n/2}$ such that the probability of "$\mathcal{B}^{\mathcal{P}[S]}$ winning the ESPR game when $\mathcal{O}$ is replaced with $\mathcal{P}[S]$" (dented by the event W) is at least $\Pr[W] \geq \varepsilon - \sqrt{p \cdot q \cdot 2^{-n/2-1}} = \varepsilon - 2^{-\Omega(n)}$.

For simplifying notation we define the function

$$flip(x_0, x_1, b)) = \begin{cases} (x_0, x_1) & \text{if } b = 1 \\ (x_1, x_0) & \text{otherwise.} \end{cases}$$

The attacker $\mathcal{B}$, on challenge $x \xleftarrow{\$} \{0,1\}^n$ outputs the Merkle path $p_\ell = (x, (x_1, b_1) \ldots, (x_\ell, b_\ell))$ and y'. Let us assume (w.l.o.g.) that $\mathcal{B}$ always makes queries to its oracle $\mathcal{P} = \mathcal{P}[S]$ on y' and on all of the values needed to evaluate the Merkle path:

$$\mathcal{P}(flip(x, x_1, b_1)) \to x'_2,\ \mathcal{P}(flip(x'_2, x_2, b_2)) \to x'_3, \ldots, \mathcal{P}(flip(x'_\ell, x_\ell, b_\ell)) \to z.$$

Let $y = flip(x'_\ell, x_\ell, b_\ell)$ be the *known preimage* associated with the Merkle path p_ℓ whose root has value z. Let us define E to be the event that the known preimage satisfies $y \in S$. Then $\Pr[W] \le \Pr[W \wedge E] + \Pr[W \wedge \neg E]$.

Since the event $W \wedge \neg E$ implies that the attacker finds a collision $\mathcal{P}[S](y) = \mathcal{P}[S](y')$ and with $y \notin S$, we can bound it by $\Pr[W \wedge \neg E] \le O(q^2)/2^n = 2^{-\Omega(n)}$.

We now bound $\Pr[W \wedge E] \le \Pr[E]$. For $v \in \{0,1\}^n$, we say that v is S*-extendable* if there exist some $v' \in \{0,1\}^n$ such that one of $flip(v, v', 0)$ or $flip(v, v', 1)$ is contained in S. Let E_1 be the event that the attacker makes a RO query $u \in (\{0,1\}^n)^2$ such that $u \notin S$, and gets a response v which is S-extendable. Let E_2 be the event that the random challenge x is S-extendable. Then $\Pr[E] \le \Pr[E_1] + \Pr[E_2]$. To see this, consider the first x'_j among $x'_1 = x, x'_2, \ldots, x'_\ell$ which is S-extendable (there must be one if E occurs). Then either $j = 1$, in which case E_2 occurs, or $j > 1$, in which case $flip(x'_{j-1}, x_{j-1}, b_{j-1})$ is not in S, but x'_j is S-extendable and so E_1 occurs. We can bound $\Pr[E_1] \le O(q \cdot s)/2^n$ and $\Pr[E_2] \le O(s)/2^n$ so $\Pr[W \wedge E] \le O(q \cdot s)/2^n = 2^{-\Omega(n)}$.

Putting it all together we get $\varepsilon - 2^{-\Omega(n)} \le \Pr[W] \le 2^{-\Omega(n)}$ implying that $\varepsilon = 2^{-\Omega(n)}$. $\square$