An extended abstract of this paper appears in *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997. This is the full version.

Does Parallel Repetition Lower the Error in **Computationally Sound** Protocols?

Mihir Bellare[*] Russell Impagliazzo[†] Moni Naor[‡]

August 17, 1997

Abstract

Whether or not parallel repetition lowers the error has been a fundamental question in the theory of protocols, with applications in many different areas. It is well known that parallel repetition reduces the error at an exponential rate in interactive proofs and Arthur-Merlin games. It seems to have been taken for granted that the same is true in arguments, or other proofs where the soundness only holds with respect to computationally bounded parties.

We show that this is not the case. Surprisingly, parallel repetition can actually fail in this setting. We present four-round protocols whose error does not decrease under parallel repetition. This holds for any (polynomial) number of repetitions. These protocols exploit non-malleable encryption and can be based on any trapdoor permutation. On the other hand we show that for three-round protocols the error does go down exponentially fast.

The question of parallel error reduction is particularly important when the protocol is used in cryptographic settings like identification, and the error represent the probability that an intruder succeeds.

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

[†]Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: russell@cs.ucsd.edu.

[‡]Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: naor@wisdom.weizmann.ac.il. Supported in part by BSF Grant 32-00032-1.

Contents

1 Introduction

Various notions of interactive protocols [23, 3, 6, 8] have found wide applicability over the last decade. They have turned out to be essential for cryptography but somewhat more surprisingly, they have also been key to complexity theory, in particular to the theory of hardness of approximation problems (see [1] for a survey). For many of these applications, the purpose of the protocol is for a "verifier" to distinguish between a "good" prover making a legitimate claim and a "bad" prover attempting to trick the verifier into accepting incorrectly. An "error bound" or "error probability" for the protocol is a value for which we have a guarantee that any bad prover will be caught except perhaps with probability this value. Many natural constructions of protocols have relatively large error bounds (constant or worse), and most applications require error bounds that are small (less than $1/\operatorname{poly}$ for any polynomial poly.)

To bridge this gap, there are two generic methods of repeating protocols intended to reduce the error: sequential repetition and parallel repetition. Sequential repetition, repeating the protocol several times, beginning the next run after the previous one terminates, reduces error in all important models. It also preserves desirable properties of the original protocol, such as zero-knowledge (see [21, 30]). However, this is an expensive solution, in that it increases the number of communication rounds of the protocol, which is undesirable for both practical and theoretical applications.

Parallel repetition was shown to reduce the error probability of Arthur-Merlin games at an exponential rate [3]. (That is, k parallel repetitions of a protocol with error ϵ results in a protocol with error ϵ^k for $k \leq \operatorname{poly}(n)$.) It can be shown that the same is true for interactive proofs, although a formal proof does not seem to have appeared. Beyond that, parallel repetition is more problematic. In single prover proofs, Goldreich and Krawczyk [19] showed that parallel repetition does not preserve zero-knowledge. In multi-prover proof systems, whether or not parallel repetition reduces the error has been the subject of much research (see [13] for a survey). There are examples of protocols for which two parallel repetitions fail to reduce the error at all [12], so a result as strong as for the single prover model does not hold. However, the error can be reduced at an exponential rate depending on the communication complexity of the given protocol [29], and this is the best possible [16].

Soon after the appearance of interactive proofs, the notion of arguments (also called computationally convincing protocols) was put forth by [8, 9]. The difference between a "proof" and an "argument" is that the verifier in a proof is protected against false provers of unlimited computational ability, whereas for an argument, the guarantee of protection is that it is computationally infeasible for a bad prover to convince the verifier with high probability. (More precisely, the soundness condition is "computational," holding only for polynomial time provers.) When designing cryptographic protocols it is natural to assume all parties are polynomial time, so this is a realistic model. Typically these computationally convincing protocols are designed based on complexity assumptions, so that the difficulty of convincing the verifier to accept a false claim is related to the difficulty of solving some hard underlying computational problem, like inverting a one-way function.

Many computationally convincing protocols have been designed, often involving parallel repetition or some variant that preserves zero-knowledge. It seems to have been assumed that, in analogy to the interactive proof case, parallel repetition does reduce the error in computationally convincing protocols. To our knowledge, we are the first to rigorously explore the question of whether this is the case in general.

What we find is somewhat surprising: the number of rounds determines whether or not there is a general parallel reduction. While in three rounds the error does go down as expected, there are four round protocols with no error reduction at all.

1.1 Our results

The main result of our paper is that parallel repetition *does not* always decrease error probabilities. Assuming the existence of trapdoor permutations we first show a protocol for which the error for two parallel repetitions is basically the same as the error of the original protocol. Next we show a protocol for which the error for $k = \text{poly}(n)$ parallel repetitions is essentially the same as the error of the original protocol.

In the last mentioned construction, the communication complexity of the original protocol depends linearly on k. Thus, these examples still hold out the possibility of a Raz-like [29] result in which the error does decrease but at a rate proportional to the communication complexity. However, we then present evidence that even this is unlikely to hold in the computational setting. We present a protocol for which there is no "black-box" error-reduction theorem, meaning that standard techniques will be unable to show any reduction in error for even an arbitrarily large polynomial number of repetitions.

These results exploit the notion and construction of non-malleable encryption schemes of [11].

We stress that this is independent of any zero-knowledge (ZK) concerns. As we indicated above, it is well known that zero-knowledge is not preserved under parallel repetition [19]. What we are saying is that even the error does not in general go down.

These results are somewhat surprising. Computationally convincing proofs have been around for a long time, and there are a large number of protocols in the literature that use parallel repetition or some variant that preserves zero knowledge. Our results say that a claim that these protocols have low error, if true, cannot rely on a general theorem but must be justified by proofs specific to the protocol at hand. For some constant round protocols, rigorous proofs of this sort have been provided [15, 5]. (Note the constructions there are not exactly parallel repetition.) More often, however, either no argument, or sketchy arguments which seem implicitly to assume parallel repetition works in general, are provided.

The example protocols that establish our negative results have four rounds of interaction. This means that for protocols of four or more rounds we cannot expect a general result saying parallel repetition reduces the error. The best we could hope for is that it does for protocols of three or less rounds. To round off our negative results, we prove this matching positive result, showing that for computationally convincing protocols of three or less rounds, parallel repetition reduces the error at about the best rate we could expect: exponentially until it becomes negligible. (We cannot expect the error probability of a computationally sound proof to ever go below negligible, as this typically is as low as we assume the probability of breaking the underlying hard computational problem like factoring.) The proof exploits techniques from [25].

These results indicate that there is a fundamental difference about computational soundness and the kind of "statistical" soundness that is the property of interactive proofs (whether single or multiple prover ones) as far as composition is concerned. They also indicate one must be careful in making claims about the error of specific computationally sound protocols.

1.2 The bigger picture

Although the main motivation came from the area of zero-knowledge arguments, the question of reducing error in computationally sound protocols makes sense whenever one polynomial time party is going to accept or reject the other, or, even more generally, whenever a party will produce a boolean output. One natural task where this occurs is identification [14]. Suppose that one can show a protocol where an unauthorized player has probability α of making the verifier accept (whereas an authorized player may know a strategy that is perfect). Here α is typically non-negligible and if this is to be used for identification, then the probability of success by an unauthorized player should be very small. It is tempting to run several copies of the protocol in parallel and hope that the probability that an unauthorized player succeeds in all of them goes down to the desired level.[1]

Other examples might include a coin-flipping protocol. If one party can bias the coin in one repetition by only a certain amount, what can we say about the probability that a fixed k-bit string is chosen if the protocol is executed k times in parallel? This is basically the same question: just view an outcome that corresponds to the target string as an "accept" and the reverse as a "reject".

1.3 A closer look

Let us try to give some idea of why error reduction by parallel repetition in computationally sound protocols might be problematic and what are the issues involved.

The basic question of whether repetition can reduce error is a form of the *direct product* question. A direct product conjecture asks whether, for some model, and a suitably hard computational problem for that model, several independent instances of the problem are harder than a single instance. A classical example of such a result is Yao's XOR lemma [31, 26, 20, 24, 25] which states that, if any feasible computation in a non-uniform model has a constant chance of failure at predicting a Boolean function, then the probability that a feasible computation could compute the $\oplus$ of several strings becomes negligible. To view this problem as a direct product result becomes clearer in the two rounds case. View the verifier's first message as a challenge or instance of a computational problem. The prover then needs to compute a string (the response) which bears a certain relationship to the challenge (the verifier accepts). Since the verifier's messages are independent, this is a direct product question for relations. One complication is that the prover cannot necessarily compute this relation, since whether the verifier accepts depends not just on the challenge, but on a random tape used to pick the challenge. (For example, the challenge could be a one-way function of the random tape.) This correspondence is the basic idea of the positive results for three round protocols, which uses a modified version of the proof in [25] of a direct product for Boolean functions. Intuitively, the reason this proof could be adapted for three round protocols is that, although we cannot tell whether the actual verifier accepts, we are interested in converting a strategy for the prover in k parallel runs to one for a single run . In using the parallel strategy, we can simulate all but one of the parallel verifiers, picking their random tapes. We can thus use whether the simulated verifiers accept as an indicator for whether the real verifier is likely to accept. If the simulated verifiers are likely to reject, then we back up and re-simulate them using fresh random tapes until most of them accept.

Unfortunately, this breaks down for the four round case. Here, there are two rounds of challenges and two rounds of responses. After the first round of challenges, the single run prover can pick a response that has a non-negligible chance of causing most of the simulated verifiers to accept. However, at this point, the prover must commit herself to the first response, and hence to the simulated verifier's first challenges. Their second challenges could be fixed as functions of their first challenges, so in general, if the simulated prover's response is rejected by many of the simulated verifiers there is no way to back up and try again. We formalize this by having the first challenge be an encryption of the second. It is important that this encryption be *non-malleable* ([11])[2]. The resulting protocol looks much like protocols obtained by a general technique to make trusted verifier zero-knowledge protocols truly zero-knowledge.

[1] The basic protocol is usually cryptographic, but this idea also makes sense in non-traditional situations. For instance, in [28] there is a proposal to use "an automated Turing test" to make sure that a human is requesting to use a resource like an on-line database and for combating junk-mail. The idea is that the user receives as a challenge a question (or task) that is easy for human but where computers have not made much progress (e.g. simple visual or linguistic problems). Since the probability that a computer will succeed in the task is non-negligible, a natural question is whether by repeating several task in parallel one reduces significantly the probability of success of a machine.

[2] It is interesting to note that while the (single-fold) protocol works due to the non-malleability of the cryptosystem the repeated protocol fails due to the malleability of the protocol itself.

2 Definitions and setup

If $f_1, f_2\colon \Lambda \to \mathsf{R}$ are functions defined over some common domain $\Lambda \subseteq \{0,1\}^*$, we say that f_1 is *eventually less than* f_2, written $f_1 \leq_{\rm ev} f_2$, if there is an integer k such that $f_1(\lambda) \leq f_2(\lambda)$ for all $\lambda \in \Lambda$ with $|\lambda| \geq k$. We say that $f\colon \Lambda \to \mathsf{R}$ is *negligible* if $f \leq_{\rm ev} |\cdot|^{-c}$ for every positive integer c, where $|\cdot|^{-c}$ stands for the function $\lambda \mapsto |\lambda|^{-c}$. For any integer m we let $[m] = \{1, \ldots, m\}$.

2.1 Computationally sound protocols

Two-party protocols. We consider a very general two party protocol setting. Think of the players as having some common initial context, represented by a binary string λ, member of an underlying set $\Lambda \subseteq \{0,1\}^*$ called the *domain*. (This λ might be, for example, messages from some previous protocol, thus possibly involving other parties, or public keys of these or other players.) The length of λ, denoted n, functions as the security parameter. The actual input for the protocol between the two players is a string x, drawn according to some input distribution I, namely $x \stackrel{R}{\leftarrow} I(\lambda)$. The first party, called the prover, is trying to convince the second party, called the verifier, of some claim related to x. They exchange messages, and, at the end of the interaction, the verifier either accepts (outputs 1) or rejects (outputs 0). We are mostly interested in the case where the parties run in time polynomial in n. The verifier is fixed in our setting, so that the protocol is fully specified given the strategy of the verifier.

We view a party B (whether prover or verifier) as an interactive algorithm. It takes inputs x, the conversation $M_1 \ldots M_i$ so far, and its random tape R to output the next message, denoted $B(x, M_1 \ldots M_i; R)$. (For simplicity we omit λ from the notation. It is assumed all parties always have access to this context.) In the case of the verifier, the last message is identified with the bit that indicates its decision. We let $B_x(\cdot;\cdot) = B(x,\cdot;\cdot)$ denote B with input fixed to x.

Computational soundness. Let A be any interactive algorithm playing the role of the prover. We let $\mathsf{Acc}(A, V, x)$ denote the probability that V accepts in its interaction with A on common input x, the probability being over the coin tosses of both parties, with x fixed. We let $\mathsf{Acc}(A, V, I, \lambda)$ denote the probability that V accepts in its interaction with A on common input x where the probability is over x drawn randomly from $I(\lambda)$ and the coin tosses of both parties. We are interested in computational soundness, namely the probability that V can be made to accept, by a polynomial time prover, measured as a function of n. The error probability is given by a function $\epsilon\colon \Lambda \to \mathsf{R}$.

Definition 2.1 *Let V be a verifier strategy over a domain Λ and input distribution I. We say that V has (computational) error probability $\epsilon(\cdot)$ if for every polynomial time prover P it is the case that* $\mathsf{Acc}(P, V, I, \cdot) \leq_{\rm ev} \epsilon(\cdot)$.

That is, the probability that a prover can convince V to accept is at most $\epsilon(\lambda)$ for long enough λ, with how long depending on the prover.

Remarks. As indicated above, this is a very general setup in that we allow a context and input distribution. The "arguments" model of [9, 8] is typically presented in terms of language recognition. That's a special case of our setup. To discuss a proof system or argument for a language L let $\Lambda = \overline{L}$ and let $I(\lambda)$ simply assign probability one to λ and zero to every other string, for each $\lambda \in \Lambda$. Then the soundness condition of Definition 2.1 collapses to the standard one. In particular, this means all our positive results apply to the standard argument model.

In any usage, the protocol must also satisfy a completeness condition. This says that there is a particular, polynomial time prover strategy P that, if provided with some "secret" information associated to the input x, succeeds in making V accept with high probability (for example, with probability 1). This P is called the honest prover. We do not formally make such a condition because the soundness, which is the main object of our study, is an independent property. But it should be understood that meaningful protocols will satisfy some form of completeness. (For example the ones in our negative results do.)

We are not discussing zero-knowledge. This may or may not be a property of our protocols. We are concerned only with soundness error.

2.2 Parallel repetition

Parallel repetition means the original protocol (specified by some verifier V) is repeated *independently* k times in parallel, where $k \leq \text{poly}(n)$. We let V^k denote the corresponding verifier, whose strategy consists of running k independent copies of V, and accepting iff V would accept in all sub-protocols. More formally, the random tape R of V^k has the form $R_1 \ldots R_k$ where each R_i is a random tape for V. The i-th message in the protocol, denoted M_i, is parsed as $M_i = M_{i,1} \ldots M_{i,k}$. The reply $M_{i+1} = V^k(x, M_1 \ldots M_i; R)$ of V^k to some message M_i is defined via $M_{i+1,j} = V(x, M_{1,j} \ldots M_{i,j}; R_j)$ for $j = 1, \ldots, k$. Note that the verifier V^k looks for a unanimous vote amongst the sub-protocols, accepting iff *all* of them are accepting for V.

Note that the prover need not respect the sub-protocols in his replies. That is, if the prover must compute a reply M_{i+1} to M_i, it can choose $M_{i+1,j}$ to depend on all previous information, including values $M_{t,l}$ where $l \neq j$ (and $t \leq i$). This is what makes parallel repetition difficult to analyze.

Fix some domain Λ and input distribution I. Say V has error probability $\epsilon(\cdot)$ over Λ and I, as per Definition 2.1. The parallel repetition problem is: what is the error probability ϵ_k of V^k?

It is important for the meaningfulness of the parallel repetition problem that we deny the verifier any secret information about either the common input x or the context λ. See Appendix C.

2.3 Black-box amplification

"Black-box" error-reduction (or amplification), as we now discuss it, is a way of proving error-reduction that is interesting for two reasons. First, it yields a strong result, and thus is desirable. Second, whenever we can prove error-reduction at all, it appears we can prove black-box error-reduction. Thus it makes sense to focus on this method.

One proves amplification by reduction. Namely, given a prover A, for the protocol defined by V^k, such that $\mathsf{Acc}(A, V^k, x) > \epsilon$, we construct a prover B, for the protocol defined by V, such that $\mathsf{Acc}(A, V^k, x) > \delta$. (This means that if the error probability of the protocol defined by V is at most δ then that of the protocol defined by V^k is at most ϵ.) The natural way (it is hard to imagine an alternative) to accomplish this transformation is "black-box." We specify an oracle machine S such that $B = S^A$. Namely, to define B, we need just one strategy which can call A as a subroutine.

Definition 2.2 *Let V be a verifier strategy over a domain Λ and input distribution I. Suppose $\epsilon, \delta \colon \Lambda \to [0, 1]$. A (k, δ, ϵ)* black-box prover transform *for V is a probabilistic, polynomial time oracle algorithm S such that for* any *prover A the following is true for all $\lambda \in \Lambda$: if $\mathsf{Acc}(A, V^k, I, \lambda) > \epsilon(\lambda)$ then $\mathsf{Acc}(S^A, V, I, \lambda) > \delta(\lambda)$. We say that V has a (k, δ, ϵ)-*black-box error-reduction procedure *if there exists a black-box prover transform for V.*

We explain what we mean by providing A to S as an oracle. The manner in which S can call the probabilistic, interactive function A is constrained. When the common input is x (a point in the support of $I(\lambda)$), algorithm S has oracle access to A_x. (It cannot invoke A on common inputs other than x.) Furthermore it does not directly supply, or even have access to, the random tape to A_x. Think of a random tape R for A_x as automatically chosen at random and fixed. S can then supply conversation prefixes c and get back $A_x(c; R)$. (Note this means S can "back-up" A on the same random tape.) Also, S has a special "reset" button: when it hits this, a new random tape R is chosen at random and fixed for A_x.

It is important that S is polynomial time, but note that A is not restricted to be polynomial time. Of course in the computational soundness setting we are only interested in the case where A is polynomial time, but it is hard to imagine a natural black-box procedure that differentiates these cases. From the point of view of S, prover A is just an oracle to be invoked (at unit cost per oracle call): the efficiency of A doesn't matter. Of course, S can only call A a polynomial number of times.

Note that our error-reduction theorem for three round protocols (namely Theorem 4.1) indeed presents a black-box prover transform. In fact it is stronger.

3 Parallel repetition fails in general

In this section we provide our negative results. Proofs of all claims here can be found in Appendix A.

3.1 Non-malleable encryption

Our constructions exploit non-malleable encryption schemes as defined and constructed in [11]. Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ specify a public key encryption scheme. The key generator algorithm $\mathcal{G}$ takes input 1^n and produces a pair (pk, sk) of matching public and secret keys. Given the public key pk, a message bit b, and coins r, the encryption algorithm $\mathcal{E}$ produces a ciphertext written $C = \mathcal{E}_{pk}(b; r)$. (When we write $C \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)$ it means the underlying choice of r is made at random.) Decryption algorithm $\mathcal{D}$ takes the secret key and ciphertext to recover the message, $b = \mathcal{D}_{sk}(C)$. Note $\mathcal{G}$ and $\mathcal{E}$ are probabilistic while $\mathcal{D}$ is deterministic, and all algorithms are polynomial time. We assume *unique decryptability:* for any bit b and string r it is the case that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(b; r)) = b$. This means there does not exist a triple C, r_0, r_1 such that $C = \mathcal{E}(0; r_0) = \mathcal{E}(1; r_1)$. This will be important for us.

Suppose the adversary is given $C \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)$ for some random bit b. According to the standard notion of semantic security [22] she cannot figure out b. We want a stronger property, namely that she cannot modify C to some different ciphertext C' whose corresponding plaintext is related to the plaintext of C. This is not guaranteed by semantic security (and in fact for many semantically secure cryptosystems it is easy given an encryption of a bit, to create an encryption of the complement bit). But it is guaranteed by non-malleability. We do not provide a formal definition here (see [11]).

Our first protocol requires only "complement security," meaning it is hard, given an encryption C of a bit b, to come up with an encryption C' of $1 - b$. Our second protocol requires "copy security," meaning it is hard, given an encryption C of a bit b, to come up with an encryption C' of b such that $C' \neq C$. (Note in either case if C' is not a valid encryption of any bit, then it "counts" as a failure.) Any non-malleable scheme has these two properties. Our third protocol actually uses non-malleability in its strongest form as per [11].

It is shown in [11] that non-malleable (and hence complement and copy secure) encryption schemes with unique decryptability exist given the existence of trapdoor permutations. Accordingly we assume such a scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is given and let $\nu\colon \{1^n : n \in \mathsf{N}\} \to \mathsf{R}$ be a negligible function that eventually upper bounds the success of any (polynomial time) adversary. We call this function the *security* of the encryption scheme. See Appendix A for a more formal development.

3.2 Two fold parallel repetition fails

We specify a four round protocol that has error about 1/2, but when repeated twice in parallel the error is still about 1/2, rather than 1/4.

The input to the parties is a public key pk of the above encryption scheme. The prover is claiming that he "knows" the decryption key sk, or, more exactly, that he knows how to decrypt. The verifier V wants to test this. Roughly, the idea is that V sends a ciphertext $B \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)$ for a random bit b, and the prover must succeed in returning an encryption of the bit b complemented. The ability to do this is viewed as corresponding to the ability to decrypt B. The full specification of the protocol is below.

In specifying the protocol we give the instructions for V and also indicate what kinds of messages the prover is providing. But we specify no particular strategy for the prover: this is irrelevant to the analysis since we are interested only in soundness. The name of the protocol is DD_2, standing for "Don't Do twice."

Protocol DD_2
Common input: pk

(1) V picks a random bit $b \in \{0,1\}$ and coins r. It sets $B = \mathcal{E}_{pk}(b;r)$ and sends B to the prover
(2) The prover sends a ciphertext C to V
(3) V sends b, r to the prover
(4) The prover sends a bit c and a string s to V
(5) V accepts iff $\mathcal{E}_{pk}(c;s) = C$ and $b \neq c$.

To complete the protocol specification we still have to ask questions like: where does the key pk come from? Formally, we fit the protocol into the framework of Section 2.1 by considering the context to be the security parameter, $\lambda = 1^n$, so that the domain is $\Lambda = \{\, 1^n \,:\, n \geq 1 \,\}$. The input distribution algorithm I takes 1^n, runs $\mathcal{G}(1^n)$ to get back (pk, sk), and outputs pk. (Intuitively, this operation is performed by the honest prover who keeps sk. The dishonest prover, who is our concern for the soundness, does not know sk.) Now, for any polynomial time prover P, we can consider the acceptance probability $\mathsf{Acc}(P, V, I, 1^n)$.

It is easy to see that there is a (polynomial time) strategy for the prover to make V accept with probability 1/2. In Step (2) it just picks a bit c at random, picks coins s at random, sets $C = \mathcal{E}_{pk}(c;s)$, and sends C to V. In Step (4) it sends c, s. It wins if $b = c$ which happens half the time. It turns out it is not possible to do much better.

Claim 3.1 *If the encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is non-malleable, then the error probability of protocol* DD_2 *is* $1/2 + \nu(\cdot)$.

Intuitively this is true since in order to win the prover must create C to be an encryption of the complement of b, given $B \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)$, and this is hard if the encryption scheme is complement secure. Indeed, given a prover with probability of success α, we can turn it into an adversary A that complements and has advantage at least $\alpha - 1/2$.

Now consider DD_2^2, the two-fold parallel repetition of DD_2. We claim its error probability is not less than 1/2 (let alone being about 1/4 as one may have wanted).

Claim 3.2 *In the protocol consisting of two parallel repetitions of* DD_2*, there is a polynomial time strategy for the prover to make the verifier accept with probability at least* $1/2$.

3.3 Many-fold parallel repetition fails

We now generalize the protocol of the previous section to k repetitions. We show that for any k there exists a protocol DD_k which is a four round protocol that has error probability or about 1/2, but when repeated k times in parallel the error probability does not reduce. As in DD_2, the input to the parties is a public key pk of the above encryption scheme. The prover is claiming that he "knows" the decryption key sk, or, more exactly, that he knows how to decrypt.

Protocol DD_k
Common input: pk

(1) V picks a random bit $b \in \{0,1\}$ and coins r. It sets $C = \mathcal{E}_{pk}(b;r)$ and sends C to the prover
(2) The prover sends $k-1$ (single-bit) ciphertexts $C_1, C_2, \ldots C_{k-1}$
(3) V sends b, r to the prover
(4) The prover sends $k-1$ pairs of (bit, random-string) $(c_1, s_1), (c_2, s_2) \ldots (c_{k-1}, s_{k-1})$ to V
(5) V accepts iff

- For all $1 \leq i \leq k-1 : \mathcal{E}_{pk}(c_i; s_i) = C_i$
- $B \notin \{C_1, C_2, \ldots C_{k-1}\}$
- $\oplus_{i=1}^{k-1} c_i \neq b$.

Using similar argument to the proof of Claim 3.1 we can show

Claim 3.3 *If the encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is non-malleable, then the error probability of protocol* DD_k *is* $1/2 + \nu(\cdot)$.

However, we can also claim that if protocol DD_k is repeated k times in parallel the error probability does not reduce significantly:

Claim 3.4 *In the protocol consisting of k parallel repetitions of* DD_k, *there is a polynomial time strategy for the prover to make the verifier accept with probability at least* $1/2$.

3.4 Failure of parallel error reduction with low communication

In light of the results of Raz [29] and Feige and Verbitsky [16] a reasonable conjecture at this point is that the failure of error-reduction in protocol DD_k is due to the fact that the communication complexity is proportional to k (the number times we are going to execute the protocol in parallel). In other words, the above results still leave open the possibility that for any protocol there is a value $\alpha > 0$ such that if the protocol is repeated k times in parallel, then the probability of failure in all k execution is α^{-k}, for sufficiently large k (the value of α is determined by the amount of communication in the protocol). This is the case with two-prover proof systems.

However, we now give strong indication that for computationally sound protocols even this is *not* the case. We show (assuming non-malleable cryptosystems exist) that there is no general black-box error-reduction procedure, where black-box means that one does not look inside computation of the players, but can just execute them and watch their behavior, as defined in Definition 2.2. We do this by presenting a particular protocol LC ("low communication") for which we can prove (assuming non-malleable cryptosystems exist) that there is no black-box error-reduction procedure.

The common input in protocol LC consists of a pair (pk_1, pk_2) of public keys drawn independently at random according to $\mathcal{G}$. Thus, formally, the domain is again $\Lambda = \{ 1^n : n \in \mathsf{N} \}$ and the input distribution I is the function which on input 1^n outputs (pk_1, pk_2) where $pk_1 \stackrel{R}{\leftarrow} \mathcal{G}(1^n)$ and $pk_2 \stackrel{R}{\leftarrow} \mathcal{G}(1^n)$.

Protocol LC
Common input: pk_1, pk_2

(1) V picks a random trit $b \in \{0,1,2\}$ and coins r. It sets $B = \mathcal{E}_{pk_1}(b;r)$ and sends B to the prover
(2) The prover sends a ciphertext C to V
(3) V sends b, r to the prover
(4) The prover sends a trit $c \in \{0,1,2\}$ and a string s to V
(5) V accepts iff $\mathcal{E}_{pk_2}(c;s) = C$ and $b + c = 0 \bmod 3$.

The following theorem says there is no black-box prover transform to show that the error of LC even reduces a little under lots of repetitions. (It is understood we mean over input distribution I defined above.)

Theorem 3.5 *Assume $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a non-malleable encryption scheme with security a negligible function. Let $k = k(n)$ be any polynomial. Let $\delta > 1/3$ be a constant and let $\epsilon \colon \Lambda \to [0, 1]$ be arbitrary. Then there in no (k, ϵ, δ)-black-box error-reduction procedure for* LC.

(Actually it is enough that $\delta(\cdot) \geq 1/3 + \tau(\cdot)$ where τ is a non-negligible function.) Note the communication complexity of LC does not depend on k, unlike DD_k.

Theorem 3.5 clearly follows from Claim 3.6 below, which says that there is a prover strategy F for LC^k which succeeds in convincing V^k with probability about 1/3, no matter how large is k. However, given this strategy as an oracle, there is no way to convince the original verifier with probability significantly more than 1/3.

Claim 3.6 *Assume $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a non-malleable encryption scheme with security a negligible function. Let $k = k(n)$ be any polynomial. There is a prover F for protocol LC^k and a negligible function $\nu(\cdot)$ such that:*

(1) $\mathsf{Acc}(F, V^k, I, 1^n) \geq 1/3 - \nu(1^n)$ *for all* $n \in \mathsf{N}$

(2) $\mathsf{Acc}(S^F, V, I, \cdot) \leq_{\mathrm{ev}} 1/3 + \nu(\cdot)$ *for any polynomial-time oracle algorithm S.*

The prover F will not be polynomial time. But that's permitted by Definition 2.2 and we explained there the rationale for this decision.

In fact we show something stronger. We will now define a certain "oracle" $\mathcal{O}_{pk_1,pk_2}(\cdot)$ and make some claims about the protocol when parties have access to this oracle. From this we will deduce Claim 3.6. The oracle is, intuitively, like a prover for LC^m. (For our purposes it suffices to set $m = k - 1$.) It has two "stages," and maintains state between invocations of the two stages. In the first stage it takes as input m ciphertexts $C_1, \ldots, C_m$ under key pk_1.

Oracle $\mathcal{O}_{pk_1,pk_2}((C_1, \ldots, C_m))$

(1) If any of $C_1, \ldots, C_m$ is invalid (meaning not the encryption of any bit under pk_1) then reject. (The ciphertexts in the cryptosystem of [11] are self-validating, so this step does not reveal any extra information to the caller of the oracle.) If not, we know there are bits $x_1, \ldots, x_m$ and strings $t_1, \ldots, t_m$ such that $C_i = \mathcal{E}_{pk_1}(x_i; t_i)$ for $i = 1, \ldots, m$.

(2) If there is some $i \neq j$ such that $C_i = C_j$ then again reject.

(3) Else, decrypt the ciphertexts to get the plaintexts $x_1, \ldots, x_m$. Set $x = x_1 + \ldots + x_m \bmod 3$.

(4) Pick s at random and return $C = \mathcal{E}_{pk_2}(x; s)$.

(5) Store a "context" for this invocation, including the information x, s.

Having been invoked on $C_1, \ldots, C_m$ and having returned C as above, the oracle can be invoked for a continuation of the interaction. Here, it is fed $(b_1, r_1), \ldots, (b_m, r_m)$.

Oracle $\mathcal{O}_{pk_1,pk_2}((C_1, \ldots, C_m), ((b_1, r_1), \ldots, (b_m, r_m)))$.

(1) Check that $\mathcal{E}_{pk_1}(b_i; r_i) = C_i$ for all $i \in [m]$ and if this is not true then reject.

(2) Return (x, s) where these quantities are as computed, and stored, in the code for the first stage above.

Note that $\mathcal{O}$ is *not* computable in polynomial time, since (in its first stage) it decrypts, without having access to the decryption key.

Also $\mathcal{O}$ it is not an "oracle" in the traditional sense since it maintains state between invocations. Since $\mathcal{O}$ is just a tool in proving Claim 3.6 this doesn't matter much, but in any case we note that this state can be eliminated by specifying s as $F_K((C_1, \ldots, C_m))$ where F is a pseudorandom function family [18] and the key K is chosen at random and made a part of the description of $\mathcal{O}$.

We first claim that given access to this oracle, it is possible to make the verifier V^k of LC^k accept 1/3 of the time.

Claim 3.7 *There is a polynomial time oracle algorithm M and a negligible function $\nu(\cdot)$ such that $\mathsf{Acc}(M^{\mathcal{O}_{pk_1,pk_2}}, V^k, (pk_1, pk_2)) \geq 1/3 - \nu(1^n)$ for any input (pk_1, pk_2) to LC^k.*

On the other hand, the non-malleability of the underlying cryptosystem is strong enough to ensure that access to this oracle does not help a polynomial time prover to convince the verifier of the original protocol with probability significantly above 1/3. This is (clearly) a consequence of the following:

Claim 3.8 *Assume that the encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is non-malleable and A is a probabilistic polynomial time oracle algorithm that is given $B \stackrel{R}{\leftarrow} \mathcal{E}_{pk_1}(b)$ where $b \stackrel{R}{\leftarrow} \{0, 1, 2\}$, and access to an oracle as described above. The probability that A succeeds in coming up with C such that $C = \mathcal{E}_{pk_2}(-b, s)$ (for some s) is bounded by $1/3 + \nu(\cdot)$ for a negligible function ν.*

Proof of Claim 3.6: Just set F to the prover $M^{\mathcal{O}_{pk_1,pk_2}}$ where M is as in Claim 3.7. The latter claim implies part (1) of Claim 3.6. Now we argue part (2) of Claim 3.6. We said above that (as a consequence of Claim 3.8), oracle access to $\mathcal{O}_{pk_1,pk_2}$ won't help a polynomial time prover convince the verifier of LC with probability significantly above 1/3. Then oracle access to F certainly won't help do it, since F can be implemented in polynomial time with access to $\mathcal{O}_{pk_1,pk_2}$. ▌

As already indicated, Theorem 3.5 follows. It remains to show the last two claims. Their proofs are in Appendix A.

3.5 Extensions

We could use non-malleable bit commitment instead of non-malleable encryption in some of these protocols to get the same result. This might lead to reducing the complexity assumptions under which the result is obtained. However, the current protocol is more natural. Also using the currently best known schemes for bit commitment would have increased the number of rounds of the protocols.

We can get similar results for protocols for proving membership in an NP language, like this. Let L be any such language. Let the input be x, y where y is an input for one of the above protocols. (That is, a public key for a non-malleable encryption scheme in the first two protocols, and two such keys for the second.) The domain is $\Lambda = \overline{L} \times \{ 1^n \ : \ n \geq 1 \}$ and the input distribution puts on y the probability as needed by our protocols above. Run some standard argument protocol on input x and then (after this protocol has completed) run one of our protocols on input y. Accept iff both sub-protocols accept. This protocol is a proof of membership in L but the error does not reduce under parallel repetition.

4 Parallel repetition reduces the error of three round protocols

In this section, we show that parallel repetition does indeed decrease error probabilities for any three pass protocol where the verifier has no secret input. The technique used is based on the XOR Casino game of [25].

Let V be a verifier defining a three message protocol. Thus V's output is either 1 (accept) or 0 (reject). Say V's random tape is of length r. Let k be a positive integer, The following theorem states a very general error-decreasing property for three-round protocols. We let $\mathsf{Comm}(A, B, x)$ be the communication complexity of the interaction between parties A, B on an input x.

Theorem 4.1 *Suppose $0 < \epsilon, \delta < 1$ and $k \geq 2$ is an integer. Suppose $\epsilon > (16/\delta) \cdot e^{-\delta^2 k/128}$. Then there is an oracle algorithm S such that for any prover P^*, verifier V and input string x, the following is true: If $\mathsf{Acc}(P^*, V^k, x) \geq 2\epsilon$ then $\mathsf{Acc}(S^{P^*,V}, V, x) \geq 1 - \delta$. Furthermore, $S^{P_x^*,V_x}$ runs in time* $\mathrm{poly}(k, |x|, 1/\epsilon, \mathsf{Comm}(P^*, V^k, x))$.

Here S requires only oracle access to P^*. But in fact S does not depend on V either, in the sense that oracle access to V will suffice too. A consequence of this is:

Corollary 4.2 *Let V be a three round verifier strategy over a domain Λ and input distribution I, and let $k = k(n)$ be $O(\log n)$. Then V has a $(k, \epsilon, 1 - \delta)$-black-box error-reduction procedure for any ϵ, δ: $\Lambda \to (0, 1)$ satisfying $\epsilon > (32/\delta) \cdot e^{-\delta^2 k/128}$.*

In terms of error probabilities, this implies the error decreases at an exponential rate:

Corollary 4.3 *Let V be a three round verifier strategy over a domain Λ and input distribution I with error probability $1 - \delta$, and let $k = k(n)$ be $O(\log n)$. Then V^k has error probability ϵ where $\epsilon(\cdot) = (32/\delta(\cdot)) \cdot e^{-\delta(\cdot)^2 k/128}$.*

Note the error goes down at an exponential rate but only to $1/\mathrm{poly}(n)$. Since the running time of S in Theorem 4.1 is a polynomial in $1/\epsilon$, and this running time must stay polynomial, we can only allow k to go as low as $O(\log n)$, which means ϵ is $1/\mathrm{poly}(n)$.

5 Open problems and on-going work

Can one show a positive result (ie. that parallel repetition reduces the error) for Arthur-Merlin games of more than three rounds? As a first step one might consider any constant number of rounds, and then more.

Our results are about soundness. What about proofs of knowledge [4]? Bellare, Halevi and Naor are investigating this question. By using the protocols here they have similar negative results for proofs of knowledge.

6 Acknowledgments

We thank Oded Goldreich for helpful comments on an earlier version of this paper.

References

[1] S. Arora and C. Lund Hardness of Approximations. In *Approximation algorithms for NP-hard problems*, edited by Dorit Hochbaum, PWS Publishing Company, Boston, 1997.

[2] L. Babai. Trading Group Theory for Randomness. *Proceedings of the* 17th *Annual Symposium on the Theory of Computing*, ACM, 1985.

[3] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *J. Computer and System Sciences* Vol. 36, 254–276, 1988.

[4] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. *Advances in Cryptology – Crypto* 92 *Proceedings*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[5] M. Bellare, M. Jakobsson and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. *Advances in Cryptology – Eurocrypt* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997.

[6] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover interactive proofs: How to remove intractability assumptions. *Proceedings of the* 20th *Annual Symposium on the Theory of Computing*, ACM, 1988.

[7] M. Blum. Coin Flipping over the Telephone. IEEE COMPCON 1982.

[8] G. Brassard and C. Crépeau. Non-transitive Transfer of Confidence: A perfect Zero-knowledge Interactive protocol for SAT and Beyond. *Proceedings of the* 27th *Symposium on Foundations of Computer Science*, IEEE, 1986.

[9] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Computer and System Sciences*, Vol. 37, 1988, pp. 156–189.

[10] G. Brassard, C. Crépeau and M. Yung. Constant round perfect zero knowledge computationally convincing protocols. *Theoretical Computer Science,* Vol. 84, No. 1, 1991.

[11] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *Proceedings of the* 23rd *Annual Symposium on the Theory of Computing*, ACM, 1991. Full version available from authors.

[12] U. Feige. On the success probability of two provers in one round proof systems. *Proceedings of the* 6th *Annual Conference on Structure in Complexity Theory*, IEEE, 1991.

[13] U. Feige. Error reduction by parallel repetition - the state of the art, Technical Report CS95-32, Weizmann Institute.

[14] U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, Vol. 1, 1988, pp. 77–94.

[15] U. Feige and A. Shamir. Zero-knowledge proofs of knowledge in two rounds. *Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[16] U. Feige and O. Verbitsky. Error reduction by parallel repetition– A negative result. *Proceedings of the* 11th *Annual Conference on Structure in Complexity Theory*, IEEE, 1996.

[17] O. Goldreich. Foundations of cryptography: Fragments of a book. Weizmann Institute of Science, February 1995.

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

[19] O. Goldreich and H. Krawczyk. On the Composition of Zero Knowledge Proof Systems. *SIAM J. on Computing*, Vol. 25, No. 1, pp. 169–192, 1996.

[20] O. Goldreich, N. Nisan and A. Wigderson. On Yao's XOR lemma. *Electronic Colloquim on Computational Complexity*, TR95-050, 1995.

[21] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, Vol. 7, No. 1, 1994, pp. 1–32.

[22] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Computer and System Sciences*, Vol. 28, 1984, pp. 270–299.

[23] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, Vol. 18, No. 1, pp. 186–208, February 1989.

[24] R. Impagliazzo. Hard-core distributions for somewhat hard problems. *Proceedings of the* 36th *Symposium on Foundations of Computer Science*, IEEE, 1995.

[25] R. Impagliazzo and A. Wigderson. P = BPP unless E has sub-exponential circuits. *Proceedings of the* 29th *Annual Symposium on the Theory of Computing*, ACM, 1997.

[26] L. Levin. One-way functions and pseudorandom generators. *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357–363.

[27] S. MICALI. CS proofs. *Proceedings of the* 35th *Symposium on Foundations of Computer Science*, IEEE, 1994.

[28] M. NAOR. Verification of a human in the loop or Identification via the Turing Test. Manuscript, 1996.

[29] R. RAZ. A parallel repetition theorem. *Proceedings of the* 27th *Annual Symposium on the Theory of Computing*, ACM, 1995.

[30] M. TOMPA AND H. WOLL. Random Self-Reducibility and Zero-Knowledge Proofs of Possession of Information. *Proceedings of the* 28th *Symposium on Foundations of Computer Science*, IEEE, 1987.

[31] A. C. YAO. Theory and Applications of Trapdoor functions. *Proceedings of the* 23rd *Symposium on Foundations of Computer Science*, IEEE, 1982.

A Proofs for Section 3

Let's formalize complement and copy security, beginning with the former. Let A be an adversary that takes pk, C, where $C = \mathcal{E}_{pk}(b)$, and outputs C'. We say A is successful if $\mathcal{D}_{sk}(C') = 1 - b$, and let

$$\begin{aligned} CompSucc_A(1^n) \;&=\; \Pr\Big[\mathcal{D}_{sk}(C') = 1 - b \,:\, (pk, sk) \stackrel{R}{\leftarrow} \mathcal{G}(1^n)\,;\, b \stackrel{R}{\leftarrow} \{0,1\}\,; \\ &\qquad C \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)\,;\, C' \stackrel{R}{\leftarrow} A(pk, C)\Big] \end{aligned}$$

Of course, A can be successful half the time by guessing b and encrypting its complement. A's *advantage* is the excess of its success probability over one-half. Set $CompAdv_A(1^n) = CompSucc_A(1^n) - 1/2$.

As for copy security, we should first rule out the case of exact copying, which is of course impossible to prevent. Formally, let A be an adversary that takes pk, C and tries to output C'. We say A is successful if $\mathcal{D}_{sk}(C') = b$ and $C' \neq C$. We let

$$\begin{aligned} CopySucc_A(1^n) \;&=\; \Pr\Big[\mathcal{D}_{sk}(C') = b \text{ and } C \neq C' \,:\, (pk, sk) \stackrel{R}{\leftarrow} \mathcal{G}(1^n)\,;\, b \stackrel{R}{\leftarrow} \{0,1\}\,; \\ &\qquad C \stackrel{R}{\leftarrow} \mathcal{E}_{pk}(b)\,;\, C' \stackrel{R}{\leftarrow} A(pk, C)\Big] \end{aligned}$$

and $CopyAdv_A(1^n) = CopySucc_A(1^n) - 1/2$.

Definition A.1 *Encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is complement secure with failure probability $\nu(\cdot)$ if for every polynomial time adversary A it is the case that $CompAdv_A(\cdot) \leq_{\rm ev} \nu(\cdot)$. Encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is copy secure with failure probability $\nu(\cdot)$ if for every polynomial time adversary A it is the case that $CopyAdv_A(\cdot) \leq_{\rm ev} \nu(\cdot)$.*

¿From [11] we have

Claim A.2 *If an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is non-malleable, then it is complement secure and copy secure with negligible failure probability $\nu(\cdot)$.*

Proof of Claim 3.1: We have to show that Definition 2.1 is met with $\epsilon(\cdot) = 1/2 + \nu(\cdot)$. Assume not. So there is some polynomial time prover P such that $\mathsf{Acc}(P, V, I, 1^n) > 1/2 + \nu(1^n)$ for all $n \in N$, where N is some infinite set of integers. We claim this contradicts the complement-security of the underlying encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. To show this we define the following adversary A. It receives input pk, B. It thinks of B as the first message from V, and computes $C = P(pk, C; R)$, the response that P would give, where R is a random string used as the coins A choses for P. Now it outputs C.

We claim that $CompSucc_A(1^n) \geq \mathsf{Acc}(P, V, I, 1^n)$ for all n. Note there is something interesting in this claim made at this point, since A has not fully simulated the interaction between P and V to the point of seeing whether the latter accepts. Indeed, A cannot expect to do that, since in the next step V provides the decryption of B, and A does not know this. Nonetheless, we want to claim that *if* the interaction had continued, V would indeed have accepted.

If P were able to make V accept, it would have been by providing a bit c and coins s such that $\mathcal{E}_{pk}(c; s) = C$. Now let $d = \mathcal{D}_{sk}(C)$. This is not something A can compute, but we can think about it in the proof. The unique decryptability property that we have assumed on the encryption scheme means that if V accepts, it must be that $d = c$, and hence that $c = 1 - b$. But A is successful exactly when $c = 1 - b$. So A is successful whenever V would have accepted, meaning $CompSucc_A(1^n) \geq \mathsf{Acc}(P, V, I, 1^n)$.

Hence $CompAdv_A(1^n) = CompSucc_A(1^n) - 1/2 > \nu(1^n)$ for all $n \in N$. But this contradicts the assumption that the encryption scheme is complement secure with failure probability $\nu(\cdot)$. ∎

Proof of Claim 3.2: We are running DD_2 twice in parallel. The notation we use is to superscript the quantities in DD_2 with the index (1 or 2) of the run. We now describe the strategy of the clever prover.

The prover receives B^1, B^2 from the verifier. It computes its return ciphertexts by "crossing" these. Namely set $C^1 = B^2$ and $C^2 = B^1$, and return C^1, C^2. Next it receives $(b^1, r^1), (b^2, r^2)$ from the verifier. Again it crosses, setting $(c^1, s^1) = (b^2, r^2)$ and $(c^2, s^2) = (b^1, r^1)$, and sends $(c^1, s^1), (c^2, s^2)$ to the verifier.

The verifier accepts if $c^1 \neq b^1$ and $c^2 \neq b^2$. (Also it must check that the coin tosses provided by the prover are consistent with the bits, but it is clear this works out, so we skip it.) With the responses defined as above, this amounts to accepting if $b^2 \neq b^1$ which happens with probability exactly 1/2, since b^1 and b^2 were chosen independently at random by the verifier. ∎

Proof of Claim 3.4: We are running DD_k k times in parallel. The notation we use is to superscript the variables in DD_k with the index $1, \ldots, k$ of the run. We now describe the strategy of the clever prover. We assume that none of $B^1, B^2, \ldots B^{k-1}$ are equal[3]. When the verifier sends $B^1, B^2, \ldots B^{k-1}$, the prover sends for the ith game

$$(C_1^i, C_2^i, \ldots C_{k-1}^i) = (B^1, \ldots B^{i-1}, B^{i+1}, \ldots B^k)\ .$$

After the verifier sends in Step (3) the pairs $(b^1, r^1), (b^2, r^2) \ldots (b^k, r^k)$, the Prover can open up correctly all the ciphertexts in Step (4).

It remains to see when the verifier accepts. The verifier accepts in all the k games iff $\oplus_{i=1}^{k-1} b^i = 1$. The probability that this happens is 1/2 therefore this is the probability of the strategy's success. ∎

Proof of Claim 3.7: The prover strategy $M^{\mathcal{O}_{pk_1, pk_2}}$ is as follows.

(1) Receives ciphertexts $B^1, \ldots, B^k$ from V^k where $B^i = \mathcal{E}_{pk_1}(b^i; r^i)$ for $i = 1, \ldots, k$.

(2) For each $i \in [k]$ set $C^i \leftarrow \mathcal{O}_{pk_1, pk_2}((B^1, \ldots, B^{i-1}, B^{i+1}, \ldots B^k))$. Send $C^1, \ldots, C^k$ to V^k.

(3) Get back $(b^1, r^1), \ldots, (b^k, r^k)$ from V^k.

(4) For each $i \in [k]$ set (c^i, s^i) to

$$\mathcal{O}_{pk_1, pk_2}((B^1, \ldots, B^{i-1}, B^{i+1}, \ldots B^k), ((b^1, r^1), \ldots, (b^{i-1}, r^{i-1}), (b^{i+1}, r^{i+1}), \ldots, (b^k, r^k)))\ .$$

Send $(c^1, s^1), \ldots, (c^k, s^k)$ to V^k.

We claim that the probability this strategy convinces V^k to accept is at least $1/3 - \nu(\cdot)$ for some negligible function $\nu(\cdot)$. To see this, assume $b^1 + \ldots + b^k \equiv 0 \bmod 3$. This happens with probability

[3] The probability of this event is low, being bounded by $k^2 \nu(\cdot)$. However, even this negligible failure probability can be eliminated; we defer the discussion.

1/3. The chance that two ciphertexts are equal is negligible, so the oracle returns answers. Now note that for any $i \in [k]$ it is the case that

$$b^i + c^i \bmod 3 \;=\; b^i + (b^1 + \ldots + b^{i-1} + b^{i+1} + \ldots + b^k) \bmod 3 \;=\; 0 \bmod 3,$$

using the definition of the oracle. Thus V^k accepts by definition of LC^k. ∎

Proof of Claim 3.8: We use here the fact that non-malleable encryption is immune against very powerful attacks: the adversary is given a challenge ciphertext and a sequence of additional ciphertexts and should come up with a ciphertext whose corresponding plaintext satisfies some relation $\mathcal{R}$ with the challenge plaintext and the plaintexts of the ciphertext sequence. Also the adversary is be given access to the decryption box *after* it receives the challenge and can query it on any ciphertext of its choice except the challenge ciphertext and sequence. Still, it must fail[4].

For our context, consider specifically an algorithm A' that receives, in addition to $B \stackrel{R}{\leftarrow} \mathcal{E}_{pk_1}(b)$, where $b \stackrel{R}{\leftarrow} \{0,1,2\}$, also $3t$ ciphertexts

$$B_0^1, B_0^2, \ldots, B_0^t,\; B_1^1, B_1^2, \ldots, B_1^t,\; B_2^1, B_2^2, \ldots, B_2^t\,,$$

such that $B_j^i \stackrel{R}{\leftarrow} \mathcal{E}_{pk_2}(b + j \bmod 3)$ for $j = 0,1,2$ and $i \in [t]$, where t is an upper bound on the number of oracle queries made by our given adversary A. (Note the additional ciphertexts are under pk_2, not pk_1.) It wins if it comes up with a valid encryption, under pk_2, of $-b \bmod 3$. Also A' has access to $\mathcal{D}_{sk_1}(\cdot)$ but is not allowed to invoke it on ciphertext B. The notions and constructions of non-malleable cryptosystems from [11] imply that such an adversary A' has only a $1/3 + \nu(\cdot)$ chance of winning where $\nu(\cdot)$ is a negligible function.

Let $\mathcal{B} = \{\, B_i^j \;:\; i \in [k] \text{ and } j = 0,1,2 \,\}$. Now, we show that if A exists we can construct an A' as above that has a chance of winning larger than possible, thus contradicting the assumed security of the non-malleable cryptosystem. A' will run A and itself provide answers to A's oracle queries. A' gives B as input to A. A' maintains counters f_0, f_1, f_2, all initially zero.

When A asks its oracle a query $(C_1, \ldots, C_m)$, our algorithm A' uses its decryption box to get $c_i \leftarrow \mathcal{D}_{sk_1}(C_i)$ for any $i \in [m]$ such that $C_i \neq B$. Let j be the sum modulo 3 of the c_i's, taken over all i for which $C_i \neq B$. We now consider two cases, that B is one of $C_1, \ldots, C_m$ and that it is not. In case none of $C_1, \ldots, C_m$ was equal to B, algorithm A' picks s at random and returns $\mathcal{E}_{pk_2}(j; s)$ as the answer to the oracle query. In case there was an i such that $C_i = B$ (note we can assume i is unique because otherwise the oracle rejects anyway and thus is easy to simulate), A' increments f_j by one and returns $B_j^{f_j}$. One can check that the distribution of these answers to oracle queries provided by A' is identical to that provided by the real oracle.

Next we must consider a second stage oracle query of the form $(C_1, \ldots, C_m), ((b_1, r_1), \ldots, (b_m, r_m))$. A' checks that indeed $\mathcal{E}_{pk_1}(b_i; r_i) = C_i$ for each $i \in [m]$ and returns a reject otherwise, just as $\mathcal{O}_{pk_1,pk_2}$ would do. Now, assuming the check passes, there are two cases just as above, namely is $B \in \{C_1, \ldots, C_m\}$ or not. If not, A' can easily return (j, s) where these were the quantities it chose in its first stage. On the other hand if $B = C_i$ for some $i \in [m]$ then A' has found b, because $b = b_i$ and b_i was provided by A. Thus A' wins directly, because it can just encrypt $-b \bmod 3$ under pk_2 and output the result. On the other hand if A' doesn't directly win it is simulating the oracle correctly.

Finally A returns, by assumption, a valid encryption, under pk_2, of $-b \bmod 3$, and A' can just output this. ∎

---

[4] Note that the security with respect to several ciphertexts follows from that security with resepct to a single one. This is discussed in the full version of [11].

B Proof of Theorem 4.1

In the given protocol (namely that defined by V) we can assume that the prover makes the first move; otherwise, V makes the last move, but this does not influence whether V accepts or rejects, so we could just suppress it. Let R denote V's random tape. Denote the prover's first move by M. We call V's response $V(M;R)$ the "challenge" and denote it by C. We call the prover's response the "answer" and denote it by A. Then whether V accepts is a function $V(M\,.\,C\,.\,A;R) \in \{0,1\}$ of M, A, R.

We can also assume that P_x^* is deterministic , by viewing its random tape as fixed to one with at least half the average probability of acceptance by V^k, namely $2\epsilon/2 = \epsilon$. (By randomly sampling an expected $\Theta(1/\epsilon)$ random tapes for P_x^* and testing them via simulating V^k and P_x^*, S can find a tape with at least an ϵ conditional probability of acceptance by V^k.)

Now consider the interaction between V^k and a prover P^* for the k-fold game, on common input x. Let $\vec{M} = (M_1,..M_k)$ be P_x^*'s first move. The following algorithm THS (Trust Halving Strategy) will be used as a sub-routine by S. It takes input an index $I \in \{1,\ldots,k\}$ indicating a sub-game, a particular challenge C of the original verifier, and $\vec{M}$, and will either produce an answer or a special symbol $\perp$ indicating failure.

Algorithm $\mathrm{THS}(I, C, (M_1, \ldots, M_k))$
  $C_I \leftarrow C$
  For $j = 1, \ldots, k$ such that $j \neq I$ **do**
    $R_j \leftarrow \{0,1\}^r$; $C_j \leftarrow V(M_j; R_j)$
  $(A_1, \ldots, A_k) \leftarrow P_x^*(M_1 \ldots M_k\,.\,C_1 \ldots C_k)$
  For $j = 1, \ldots, k$ such that $j \neq I$ **do**
    $d_j \leftarrow V(M_j\,.\,C_j\,.\,A_j; R_j)$ (a bit indicating accept or reject for the j-th sub-game)
  $t \leftarrow |\{\, j \in [k] - \{I\} \,:\, d_j = 0 \,\}|$
  With probability 2^{-t} output A_I, else output $\perp$

The idea of the trust halving strategy is that S wants to use P_x^*'s answer in the I-th game as the response to challenge C_I if P_x^* is actually producing good answers. However, most of the time (namely a $1-\epsilon$ fraction of the time), P_x^* may produce an arbitrary mixture of answers that are and are not accepted. We use the answers on the other sub-games to decide whether or not to trust P_x^*'s answer on C_I. The trust halving strategy was introduced in the context of direct product theorems in [25].

Since THS usually produces no output, S will actually need to run it several times. Also, the prover for a single run will need to send its first message M_I before receiving a challenge, and so one good value of I needs to be found.

We usually choose R_I according to some distribution and then run $\mathrm{THS}(I, C, \vec{M})$ where $C = V(M_I; R_I)$. The THS algorithm now picks at random $R_1, \ldots, R_{I-1}, R_{I+1}, \ldots, R_k$. We refer to $\vec{R} = (R_1, \ldots, R_k)$ as the *execution base* of this run of the algorithm.

Fix $I \in [k]$, a random tape $R \in \{0,1\}^r$ for V, and a sequence $\vec{M}$ of messages from P_x^*. For $d \in \{0,1\}$ (that is, accept or reject) let

$$T_d(I, R, \vec{M}) \;=\; \Pr\left[\, C \leftarrow V(M_I; R)\,;\; A \leftarrow \mathrm{THS}(I, C, \vec{M}) \;:\; A \neq \perp \;\text{ and }\; V(M_I\,.\,C\,.\,A; R) = d \,\right] .$$

In other words, $T_1(I, R, \vec{M})$ (resp. $T_0(I, R, \vec{M})$) is the probability that THS produces an output $A \neq \perp$, and this answer makes the original verifier accept (resp. reject). The probability here is over the random choices of $R_1, \ldots, R_{I-1}, R_{I+1}, \ldots, R_k$, and the randomness underlying the final 2^{-t} probability, made by THS. Say that $(I, R, \vec{M})$ is *good* if

$$T_1(I, R, \vec{M}) \;\geq\; \frac{2}{\delta} \cdot T_0(I, R) + \frac{\epsilon}{16} \,.$$

If $(I, R, \vec{M})$ is not good, say it is *bad*. For $I \in \{1, \ldots, k\}$ let

$$\begin{aligned} \mathcal{B}(I, \vec{M}) &= \{ R \in \{0,1\}^r : (I, R, \vec{M}) \text{ is bad} \} \\ b(I, \vec{M}) &= |\mathcal{B}(I, \vec{M})| . \end{aligned}$$

Notice that by sampling we can estimate $b(I, \vec{M})$ given I. (More precisely, given I, oracles for P_x^*, V_x, and any parameters $\epsilon', \delta' > 0$, we can produce, in time $\mathrm{poly}(1/\epsilon', \log(1/\delta'), \mathsf{Comm}((, P,^) *, V^k, x))$, an estimate $\tilde{b}(I, \vec{M})$ such that $|\tilde{b}(I, \vec{M}) - b(I, \vec{M})| \leq \epsilon'$ with probability at least $1 - \delta'$.) For simplicity we assume henceforth that it is possible to actually compute $b(I, \vec{M})$. We are now ready to specify how our prover provides the answer.

Algorithm $S^{P_x^*, V_x}(x, \cdot; \cdot)$

(1) $(M_1, \ldots, M_k) \leftarrow P_x^*()$

(2) By estimating $b(i, \vec{M})$ for all $i \in [k]$, find a value of $I \in [k]$ such that $b(I, \vec{M}) \leq (\delta/4) \cdot 2^r$. (Such a value exists by Lemma B.3.)

(3) Send M_I to the verifier, and receive in response a challenge C

(4) Run $\mathrm{THS}(I, C, \vec{M})$ until either an output A is produced or a total of $16 \ln(4/\delta)/\epsilon$ runs are completed. In the former case, provide A as the answer to the verifier. In the latter case, fail.

The following lemma considers the interaction between prover $S^{P_x^*, V}$ and verifier V on input x, and says the probability of acceptance is quite high.

Lemma B.1 $\mathsf{Acc}(S^{P_x^*, V_x}, V, x) \geq 1 - \delta$.

Proof: From lemma B.3, S will succeed in finding an $I \in [k]$ so that $b(I, \vec{M}) \leq (\delta/4) \cdot 2^r$. If $(I, R, \vec{M})$ is good then the probability that S produces an A such that $V(M_I C A; R) = 1$ is at least $1 - 3\delta/4$ by Lemma B.2. Thus the overall probability that V accepts in its interaction with S is at least $(1 - \delta/4)(1 - 3\delta/4) > 1 - \delta$. ∎

Lemma B.2 *Suppose S chooses I as the index in Step 2 and $(I, R, \vec{M})$ is good. Let $C = V(M_I; R)$. Then the probability that S produces an answer A such that $V(M_I C A; R) = 1$ is at least $1 - 3\delta/4$. (The probability is over the random choices of S in the steps following Step 2.)*

Proof: Since $(I, R, \vec{M})$ is good, we have $T_1(I, R, \vec{M}) > (2/\delta) \cdot T_0(I, R, \vec{M}) + \epsilon/16$. Then S makes independent simulations of THS until the latter produces some output, or until $16(\ln(4/\delta))/\epsilon$ such simulations have been completed. The probability that THS produces some output is at least $T_1(I, R, \vec{M}) > \epsilon/16$. So the probability of S "timing out" without producing output, is at most $(1 - \epsilon/16)^{16(\ln(4/\delta))/\epsilon} \leq \delta/4$. On the other hand, since S makes independent tries of THS until output is produced, given that S produces an output, the probability that V accepts is

$$\begin{aligned} \frac{T_1(I, R, \vec{M})}{T_1(I, R, \vec{M}) + T_0(I, R, \vec{M})} &\geq \frac{T_1(I, R, \vec{M})}{T_1(I, R, \vec{M}) + (\delta/2)[T_1(I, R, \vec{M}) - \epsilon/16]} \\ &\geq \frac{T_1(I, R, \vec{M})}{T_1(I, R, \vec{M}) + (\delta/2) T_1(I, R, \vec{M})} \\ &= \frac{1}{1 + \delta/2} \\ &\geq 1 - \delta/2 . \end{aligned}$$

Thus, the probability that S produces an output A and $V(M_I C A; R) = 1$ is at least $1 - \delta/2 - \delta/4 = 1 - 3\delta/4$. ∎

Lemma B.3 *There is an $I \in [k]$ such that $b(I, \vec{M}) \leq (\delta/4) \cdot 2^r$.*

Proof: Assume not, meaning $b(i, \vec{M}) > (\delta/4) \cdot 2^r$ for all $i = 1, \ldots, k$. Now for each $i = 1, \ldots, k$ fix a set $B(i, \vec{M}) \subseteq \mathcal{B}(i, \vec{M})$ such that $|B(i, \vec{M})|$ is *exactly* $(\delta/4) \cdot 2^r$. We claim that

$$\mathbf{E}\left[\, i \leftarrow [k]\,;\; R \leftarrow B(i, \vec{M}) \;:\; T_1(i, R, \vec{M}) - (2/\delta) \cdot T_0(i, R, \vec{M}) \,\right] \;>\; \frac{\epsilon}{16}\,. \tag{1}$$

This implies that there exists an index $i \in [k]$ and a string $R \in B(i, \vec{M})$ such that $T_1(i, R, \vec{M}) - (2/\delta) \cdot T_0(i, R, \vec{M}) > \epsilon/16$, contradicting the fact that $(i, R, \vec{M})$ is bad for any $R \in B(i, \vec{M})$, and thus proves the lemma. It remains to prove Equation (1).

We first introduce some notation. For $\vec{R} = (R_1, \ldots R_k)$ a fixed sequence of random tapes, let

$$\begin{aligned} L(\vec{R}, \vec{M}) &= \{\, i \in [k] \;:\; R_i \in B(i, \vec{M}) \,\} \\ l(\vec{R}, \vec{M}) &= |L(\vec{R}, \vec{M})|\,. \end{aligned}$$

With $\vec{R}$ still fixed let $\vec{C} = (C_1, \ldots, C_k)$ where $C_j = V(M_j; R_j)$ for $j = 1, \ldots, k$, and let $(A_1, \ldots, A_k) \leftarrow P_x^*(\vec{M}\vec{C})$. Now let $T(\vec{R}, \vec{M})$ be the set of all $i \in [k]$ such that $V(M_i C_i A_i; R_i) = 0$. Let $t(\vec{R}, \vec{M}) = |T(\vec{R}, \vec{M})|$.

We now define a distribution $\mu_{\vec{M}}$: $\{0,1\}^{rk} \to [0,1]$ which assigns a probability $\mu_{\vec{M}}(\vec{R})$ to any random tape $\vec{R} = (R_1, \ldots, R_k)$ of V^k, computed as follows:

$$\begin{aligned} &\mu_{\vec{M}}(\vec{R}) = \\ &\quad \Pr\left[\, i \leftarrow [k]\,;\; R_i' \leftarrow B(i, \vec{M})\,;\; R_j' \leftarrow \{0,1\}^r \text{ for } j \in [k] - \{i\} \;:\; (R_1', \ldots, R_k') = (R_1, \ldots, R_k) \,\right]\,. \end{aligned} \tag{2}$$

This is the probability that $\vec{R}$ is the execution base for THS in the experiment where we first pick $I \leftarrow [k]$ and $R_I \leftarrow B(I, \vec{M})$, set $C = V(M_I; R_I)$, and then run $\mathrm{THS}(I, C, \vec{M})$.

Claim 1. $\mu_{\vec{M}}(\vec{R}) = \dfrac{4 \cdot l(\vec{R}, \vec{M})}{\delta k} \cdot 2^{-rk}$.

Proof. Refer to Equation (2) and consider the chance that the experiment in question indeed yields the particular outcome $\vec{R}$. For this to happen it must be that the randomly chosen i lands in $L(\vec{R}, \vec{M})$, and this happens with probability $l(\vec{R}, \vec{M})/k$. Next, R_i' must equal R_i, and this happens with probability $1/|B(i, \vec{M})| = 2^{-r}/(\delta/4)$. Finally it must be that $R_j' = R_j$ for $j \neq i$ and this happens with probability $2^{-(k-1)r}$. So

$$\mu_{\vec{M}}(\vec{R}) = \frac{l(\vec{R}, \vec{M})}{k} \cdot \frac{2^{-r}}{\delta/4} \cdot 2^{-(k-1)r}\,,$$

which simplifies to the claimed quantity. □

Claim 2. Given $\vec{R}$ drawn according to distribution $\mu_{\vec{M}}$, the distribution on the index i (in the experiment of Equation (2)) is uniform over $L(\vec{R}, \vec{M})$.

Proof. Clear. □

Now we need a few more definitions. For a fixed $\vec{R} = (R_1, \ldots, R_k)$ let $\mathrm{THS}_{\vec{R}}(I, \vec{M})$ be the output of $\mathrm{THS}(I, V(M_I; R_I), \vec{M})$ when the execution base is $\vec{R}$. (This is a random variable depending on the random choices underlying the 2^{-t} probability in the description of THS, the last being the only randomness left when the execution base is fixed.) For any $\vec{R}$ such that $\mu_{\vec{M}}(\vec{R}) > 0$, and each $d \in \{0,1\}$, let

$$T_d(\vec{R}, \vec{M}) = \mathbf{E}\left[\, i \leftarrow L(\vec{R}, \vec{M})\,;\; A \leftarrow \mathrm{THS}_{\vec{R}}(i, \vec{M}) \;:\; A \neq \perp \text{ and } V(M_i C A; R_i) = d \,\right]$$

be the chance that THS produces an output with outcome d when its execution base is fixed to $\vec{R}$ and i is chosen at random from $L(\vec{R},\vec{M})$. Let

$$X(\vec{R},\vec{M}) \;=\; T_1(\vec{R},\vec{M}) - \frac{2}{\delta}\cdot T_0(\vec{R},\vec{M})\,. \tag{3}$$

Notice that

$$\mathbf{E}\left[\,\vec{R}\leftarrow\mu_{\vec{M}} \;:\; X(\vec{R},\vec{M})\right] \;=\; \mathbf{E}\left[\,i\leftarrow[k]\,;\; R\leftarrow B(i,\vec{M}) \;:\; T_1(i,R,\vec{M}) - (2/\delta)\cdot T_0(i,R,\vec{M})\right].$$

Thus to show Equation (1) we wish to show that

$$\mathbf{E}\left[\,\vec{R}\leftarrow\mu_{\vec{M}} \;:\; X(\vec{R},\vec{M})\right] \stackrel{\text{def}}{=} \textstyle\sum_{\vec{R}\in\{0,1\}^{rk}} X(\vec{R},\vec{M})\cdot\mu_{\vec{M}}(\vec{R}) \;>\; \frac{\epsilon}{16}\,. \tag{4}$$

Claim 3. Fix $\vec{R}$ such that $\mu_{\vec{M}}(\vec{R})>0$ and let $\vec{C}=V_x^k(\vec{M};\vec{R})$.

(1) Let $\vec{A}=P_x^*(\vec{M}\vec{C})$. If $V_x^k(\vec{M}\vec{C}\vec{A};\vec{R})=1$ (equivalently, $t(\vec{R},\vec{M})=0$) then $X(\vec{R},\vec{M})=1$.

(2) $X(\vec{R},\vec{M}) \;\geq\; \dfrac{2^{-t(\vec{R},\vec{M})}}{l(\vec{R},\vec{M})}\cdot\left[l(\vec{R},\vec{M}) - \left(1+\dfrac{4}{\delta}\right)\cdot t(\vec{R},\vec{M})\right]$.

(3) $X(\vec{R},\vec{M}) \;\geq\; -\dfrac{4}{\delta}\cdot 2^{-t(\vec{R},\vec{M})}$.

Proof. For (1), note that $V_x^k(\vec{M}\vec{C}\vec{A};\vec{R})=1$ implies $T_1(\vec{R},\vec{M})=1$ and $T_0(\vec{R},\vec{M})=0$.

Now we claim

$$T_1(\vec{R},\vec{M}) \;\geq\; \frac{l(\vec{R},\vec{M})-t(\vec{R},\vec{M})}{l(\vec{R},\vec{M})}\cdot 2^{-t(\vec{R},\vec{M})} \tag{5}$$

$$T_0(\vec{R},\vec{M}) \;\leq\; \frac{t(\vec{R},\vec{M})}{l(\vec{R},\vec{M})}\cdot 2^{-[t(\vec{R},\vec{M})-1]}\,. \tag{6}$$

By Equation (3), this implies both part (2) and part (3) of Claim 3, so it suffices to show these two bounds.

To see Equation (5), consider when does $\mathrm{THS}_{\vec{R}}(i,\vec{M})$ produce an answer that is accepted by V, in the experiment defining $T_1(\vec{R},\vec{M})$. This happens only when i lands in the set $L(\vec{R},\vec{M})-T(\vec{R},\vec{M})$. Furthermore, since there is an acceptance in the i-th sub-game, the number t of rejections t that THS counts will equal $t(\vec{R},\vec{M})$, and thus, given that i lands in the set in question, there is a $2^{-t(\vec{R},\vec{M})}$ chance that an accepted output is produced.

To see Equation (6), consider when does $\mathrm{THS}_{\vec{R}}(i,\vec{M})$ produce an answer that is rejected by V, in the experiment defining $T_0(\vec{R},\vec{M})$. This happens only when i lands in the set $L(\vec{R},\vec{M})\cap T(\vec{R},\vec{M})$, whose size is upper bounded by $t(\vec{R},\vec{M})$. Furthermore, since there is a rejection in the i-th sub-game, the number t of rejections that THS counts will equal $t(\vec{R},\vec{M})-1$, and thus, given that i lands in the set in question, there is a $2^{-[t(\vec{R},\vec{M})-1]}$ chance that a rejected output is produced. □

Let $l_0=(\delta k)/8$ and $t_0=l_0/(1+4/\delta)$. Partition $\{0,1\}^{kr}$ into four sets as follows:

(1) ACCEPTED $=\{\,\vec{R}\in\{0,1\}^{kr} \;:\; t(\vec{R},\vec{M})=0\,\}$ (In other words, those random tapes on which V^k accepts P_x^*. These give THS a non-negligible advantage.)

(2) EASY $=\{\,\vec{R}\notin$ ACCEPTED $:\; b(\vec{R},\vec{M})<l_0\,\}$ (In other words, those random tapes without many "hard to please" components. These are rare.)

(3) ADVANTAGE $=\{\,\vec{R}\in\{0,1\}^{kr} \;:\; b(\vec{R},\vec{M})\geq l_0$ and $0<t(\vec{R},\vec{M})\leq t_0\,\}$ (Those random tapes where P_x^* convinces the verifier on sufficiently many protocols that it is to our advantage to use one of them as our move.)

(4) $\textsc{Mistakes} = \{ \vec{R} \in \{0,1\}^{kr} \ : \ b(\vec{R},\vec{M}) \geq l_0 \text{ and } t(\vec{R},\vec{M}) > t_0 \}$ (Those random tapes where P_x^* fails to convince the verifier on many runs of the protocol. These will almost always produce no output for THS.)

For any set $A \subseteq \{0,1\}^{kr}$ let

$$\mathcal{S}(A) \ = \ \textstyle\sum_{\vec{R} \in A} X(\vec{R},\vec{M}) \cdot \mu_{\vec{M}}(\vec{R}) \, .$$

We analyze $\mathcal{S}(\{0,1\}^{kr})$ by breaking it up over the four sets defined above.

Suppose we flip k coins, independently, where each has probability $p = \delta/4$ of being heads. Let Q be the probability that we get fewer than $l_0 = (\delta k)/8$ heads. By Chernoff bounds,

$$Q \ < \ e^{-(\delta k/8)^2/(2k)} \ = \ e^{-\delta^2 k/128} \, . \tag{7}$$

Claim 4.

(1) $\mathcal{S}(\textsc{Accepted}) \geq (\epsilon - Q)/2$
(2) $\mathcal{S}(\textsc{Easy}) \geq -Q/\delta$
(3) $\mathcal{S}(\textsc{Advantage}) \geq 0$
(4) $\mathcal{S}(\textsc{Mistakes}) \geq -(4/\delta)2^{-t_0}$

Proof. From the first part of Claim 3 we know that $X(\vec{R},\vec{M}) = 1$ for every $\vec{R} \in \textsc{Accepted}$, so $\mathcal{S}(\textsc{Accepted}) = \Pr\left[\, \vec{R} \leftarrow \mu_{\vec{M}} \ : \ \vec{R} \in \textsc{Accepted} \,\right]$ is just the probability, under distribution $\mu_{\vec{M}}$, of the set $\textsc{Accepted}$. By assumption $\mathsf{Acc}(P^*, V^k, x) \geq \epsilon$ so $|\textsc{Accepted}| \geq \epsilon 2^{rk}$. Let $\textsc{HardAccepts} = \{ \vec{R} \in \textsc{Accepted} \ : \ l(\vec{R},\vec{M}) \geq l_0 \}$. Then $|\textsc{HardAccepts}| \geq (\epsilon - Q)2^{rk}$, since Q is the fraction of sequences with $l(\vec{R},\vec{M}) < l_0$. For every $\vec{R} \in \textsc{HardAccepts}$, we have (using Claim 1)

$$\mu_{\vec{M}}(\vec{R}) \ = \ \frac{4l(\vec{R},\vec{M})}{\delta k} \cdot 2^{-rk} \ = \ \frac{l(\vec{R},\vec{M})}{2l_0} \cdot 2^{-rk} \ \geq \ (1/2)2^{-rk} \, .$$

Thus,

$$\begin{aligned}
\mathcal{S}(\textsc{Accepted}) \ &= \ \Pr\left[\, \vec{R} \leftarrow \mu_{\vec{M}} \ : \ \vec{R} \in \textsc{Accepted} \,\right] \\
&\geq \ \textstyle\sum_{\vec{R} \in \textsc{HardAccepts}} \mu_{\vec{M}}(\vec{R}) \\
&\geq \ |\textsc{HardAccepts}| \cdot (1/2)2^{-rk} \\
&\geq \ (\epsilon - Q)/2 \, .
\end{aligned}$$

This proves the first part.

Similarly to the first part, $|\textsc{Easy}| \leq Q2^{rk}$, and for every $\vec{R} \in \textsc{Easy}$, $\mu_{\vec{M}}(\vec{R}) \leq (1/2)2^{-rk}$. Since $X(\vec{R},\vec{M}) \geq -2/\delta$ for any $\vec{R}$, we have

$$\mathcal{S}(\textsc{Easy}) \ \geq \ (-2/\delta) \textstyle\sum_{\vec{R} \in \textsc{Easy}} \mu_{\vec{M}}(\vec{R}) \ \geq \ (-2/\delta)|\textsc{Easy}|(1/2)2^{-rk} \geq -Q/\delta \, .$$

This proves the second part.

For any $\vec{R} \in \textsc{Advantage}$ we have

$$\begin{aligned}
X(\vec{R},\vec{M}) \ &\geq \ \frac{2^{-t(\vec{R},\vec{M})}}{l(\vec{R},\vec{M})} \cdot \left[l(\vec{R},\vec{M}) - \left(1 + \frac{4}{\delta}\right) \cdot t(\vec{R},\vec{M}) \right] \quad \text{(by Claim 3)} \\
&\geq \ \frac{2^{-t(\vec{R},\vec{M})}}{l(\vec{R},\vec{M})} \cdot \left[l_0 - \left(1 + \frac{4}{\delta}\right) \cdot t_0 \right] \quad \text{(because } t(\vec{R},\vec{M}) \leq t_0 \text{ and } l(\vec{R},\vec{M}) \geq l_0) \\
&= \ 0 \quad \text{(because } t_0 = l_0/(1 + 4/\delta)) \, .
\end{aligned}$$

This proves the third part.

¿From part three of Claim 3, for any $\vec{R} \in$ MISTAKES we have $X(\vec{R},\vec{M}) \geq -(4/\delta)2^{-t(\vec{R},\vec{M})} \geq -(4/\delta)2^{-t_0}$. This proves the fourth part.

This concludes the proof of Claim 4. □

Now we can apply Claim 4 to get

$$
\begin{aligned}
\mathcal{S}(\{0,1\}^{rk}) &= \mathcal{S}(\text{ACCEPTED}) + \mathcal{S}(\text{EASY}) + \mathcal{S}(\text{ADVANTAGE}) + \mathcal{S}(\text{MISTAKES}) \\
&\geq (\epsilon - Q)/2 - Q/\delta - 0 - (4/\delta)2^{-t_0} \\
&\geq \epsilon/2 - Q/2 - Q/\delta - (4/\delta)e^{-\frac{\delta k/8}{1+4/\delta}} \\
&= \epsilon/2 - Q/2 - Q/\delta - (4/\delta)e^{-\frac{\delta^2 k}{8(4+\delta)}} \\
&\geq \epsilon/2 - Q/2 - Q/\delta - (4/\delta)e^{-\frac{\delta^2 k}{40}} \quad \text{(because } \delta < 1) \\
&> \epsilon/2 - Q/2 - Q/\delta - (4/\delta)e^{-\frac{\delta^2 k}{128}} .
\end{aligned}
$$

The assumption in the statement of Theorem 4.1 is that $\epsilon > (16/\delta)e^{-\delta^2 k/128}$. Combining this with Equation (7) yields $Q < e^{-\delta^2 k/128} < \delta\epsilon/16 < \epsilon/16$. Using these estimates in the above we get

$$
\begin{aligned}
\mathcal{S}(\{0,1\}^{rk}) &\geq \epsilon/2 - \epsilon/32 - \epsilon/16 - (4/\delta)(\delta\epsilon/16) \\
&> \epsilon/2 - 6\epsilon/16 \\
&> \epsilon/8 .
\end{aligned}
$$

This yields Equation (4), which we had already said yields Equation (1), and thus concludes the proof of Lemma B.3. ▮

C Parallel repetition when the verifier has secret information

In our model the verifier is not given any secret information pertaining to the input x or to the context λ: its strategy must be a computable in (probabilistic) polynomial time given x, λ. This is important to make the parallel repetition question meaningful as we now explain.

If we were to allow the verifier access to private information about x or λ then it is trivial to see that parallel repetition fails to lower the error. In fact one can easily specify a two round protocol which has error $1/2$ but, when repeated k times in parallel, the error actually increases with k, tending to 1. For example, say x (or λ) is a random integer product of two primes p, q such that V knows p, q. Let the protocol be that V flips a coin and sends in its first message p, q if the coin was 0 and nothing if the coin was 1. It accepts if the prover's reply is p, q. The error probability of this protocol is $1/2 + \nu(\cdot)$ where $\nu(\cdot)$ is negligible. However, if we repeat the protocol k times there is a strategy for the prover to win with probability $1 - 2^{-k}$, because except with probability 2^{-k} the factorization p, q is released by the verifier in *some* run, and then the prover can echo it in all runs.

Is it reasonable that we deny the verifier secret information? Certainly it seems reasonable that the verifier has no secret information about x, because in cryptographic settings x is chosen by the (honest) prover and given to V, and the prover is trying to prove something about x to V, so the latter would not have any information about x other than what he could compute in polynomial time given x, which is what we assume. It is possible that V does have secret information related to λ. (For example the latter could contain his public key, of which he holds the corresponding secret key.) Still, it makes sense that the protocol, whose goal is to prove a claim about x, does not use this secret information, so that as far as the protocol is concerned, we can assume the verifier strategy does not depend on that secret information. Certainly all "useful" protocols fall in our model; we are eliminating only artificial ones.