On the Impossibility of Cryptography with Tamperable Randomness

Per Austrin[*], Kai-Min Chung[†], Mohammad Mahmoody[‡], Rafael Pass[§], Karn Seth[¶]

October 16, 2018

Abstract

We initiate a study of the security of cryptographic primitives in the presence of efficient tampering attacks to the randomness of honest parties. More precisely, we consider p-tampering attackers that may *efficiently* tamper with each bit of the honest parties' random tape with probability p, but have to do so in an "online" fashion. Our main result is a strong negative result: We show that any secure encryption scheme, bit commitment scheme, or zero-knowledge protocol can be "broken" with advantage $\Omega(p)$ by a p-tampering attacker. The core of this result is a new algorithm for biasing the output of bounded-value functions, which may be of independent interest.

We also show that this result cannot be extended to primitives such as signature schemes and identification protocols: assuming the existence of one-way functions, such primitives can be made resilient to $(1/\mathrm{poly}(n))$-tampering attacks where n is the security parameter.

Keywords Tampering, Randomness, Encryption.

[*]KTH Royal Institute of Technology austrin@kth.se. Work done while at Univ. of Toronto, funded by NSERC.

[†]Cornell chung@cs.cornell.edu. Supported in part by NSF Award CNS-1217821.

[‡]University of Virginia mohammad@cs.virginia.edu. Supported by NSF CAREER award CCF-1350939.

[§]Cornell {rafael}@cs.cornell.edu. Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2- 0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government

[¶]Cornell {karn}@cs.cornell.edu.

Contents

1 Introduction

A traditional assumption in cryptography is that the only way for an attacker to gather or control information is by receiving and sending messages to honest parties. In particular, it is assumed that the attacker may not access the *internal states* of honest parties. However, such assumptions on the attacker—which we refer to as *physical assumptions*—are quite strong (and even unrealistic). In real-life, an attacker may through a "physical attack" learn some "leakage" about the honest parties' internal states and may even tamper with their internal states. For instance, a computer virus may (e.g., using a, so-called, buffer overflow attack [Ale96, Fry00, PB04]) be able to bias the randomness of an infected computer. Understanding to what extents the traditional physical assumptions can be relaxed, to capture such realistic attacks, is of fundamental importance.

Indeed, in recent years *leakage-resilient cryptography*—that is, the design of cryptographic schemes and protocols that remain secure even when the attacker may receive (partial) leakage about the internal state of the honest parties—has received significant attention (see e.g., [MR04, DP08, AGV09, BKKV10, DHLAW10, DGK+10, KLR12, GR12, LL12, DSK12, Rot12]).

In this work, we focus on understanding the power of *tampering attacks*—that is, attacks where the adversary may partially modify (i.e., tamper with) the internal state of honest parties. Early results in the 1990's already demonstrate that tampering attacks may be very powerful: by just slightly tampering with the computation of specific implementations of some cryptographic schemes (e.g., natural implementations of RSA encryption [RSA78]), Boneh, DeMillo and Lipton [BDL97] demonstrated that the security of these schemes can be broken completely.

Previous works on tamper-resilient cryptography consider tampering of computation [AK96, BDL97, BS97, IPSW06, FPV11, LL12] and tampering with the memory of an honest party who holds a secret (e.g., a signing or a decryption algorithm) [GLM+04, DPW18, LL10, KKS11, LL12, CKM11]. This line of research culminated in strong *compilers* turning any polynomial-size circuit C into a new "tamper-resilient" polynomial-size circuit C'; tamper-resilience here means that having "grey-box" access to C' (i.e., having black-box access while tampering with the computation of C') yields no more "knowledge" than simply having black-box access to C. These works, thus, show how to keep a secret hidden from a tampering attacker. Our focus here is somewhat different. In analogy with recent work of leakage-resilient security, we aim to investigate to what extent a tampering attacker may violate the *security* of a cryptographic protocol by tampering with the internal state of honest parities.

For concreteness, let us focus on the security of public-key encryption schemes (but as we shall see shortly, our investigation applies to many more cryptographic tasks such as zero-knowledge proofs and secure computation). Roughly speaking, we require a tamper-resilient encryption scheme to guarantee that ciphertexts conceal the encrypted messages, even if the internal computation of the sender (of the ciphertext) has been tampered with.[1] As first observation note that if the attacker may completely change the computation of the sender, he could simply make the sender send the message in the clear. Thus, to hope for any reasonable notion of tamper-resilient security we need to restrict the attacker's ability to tamper with the computation.

[1] Let us remark that the simulation property of tamper-resilient compilers do not necessarily guarantee that if the sender algorithm is compiled into a "tamper-resilient" version, then the encryption scheme is tamper-resilient. This is due to the fact that the simulation property of those compilers only guarantee that an attacker cannot learn more from tampering with the sender strategy than it could have with black-box access to it. But in the case of encryption schemes, it is actually the *input* to the algorithm (i.e., the message to be encrypted) that we wish to hide (as opposed to some secret held by the algorithm).

Tampering with Randomness. Among various computational resources, randomness might be one of the hardest to protect against tampering. This is due to the fact that randomness is usually *generated* (perhaps based on some "physical" resources available to the system) and any malicious attacker who is able to change the bits along their generation can mount a tampering attack against the randomness. Indeed given the breakthrough results of [HDWH12, LHA+12a, LHA+12b] it is becoming even more clear that randomness is one of the most vulnerable aspects of a cryptographic system. Thus, a very basic question is to what extent we can protect our systems against tampering with randomness. In this work we initiate a formal study of this question by considering tampering attacks to the randomness of the honest players; namely we study the following basic question:

> *Can security of cryptographic primitives be preserved under tampering attacks to the randomness of honest parties?*

Note that we need to restrict the tampering ability of the attacker, for otherwise the adversary can effectively make the scheme deterministic by always fixing the randomness of the honest parties to all zeros. But it is well-known that deterministic encryption schemes cannot be semantically secure. Therefore, here we initiate study of the power of *weak* types of tampering attacks to the randomness of the honest parties.

General Model: The Tampering Virus. We envision the adversary as consisting of two separate entities: **(1)** a *classical attacker* who interacts with the honest parties only by sending/receiving messages to/from them (without any side-channels), and **(2)** a *tampering circuit* (a.k.a. the "virus") who observes the internal state of the honest parties and may *only* tamper with their randomness (but may not communicate with the outside world, and in particular with the classical attacker). The tampering circuit only gets to tamper with a *small fraction* of the random bits, and in an *efficient* manner. Note that this model excludes a scenario in which the virus (even efficiently) samples the *whole* randomness, regardless of the original randomness sampled by the system, because in this cases all of the sampled tampered bits might be different from the system's original random seed. However, here we study weak attackers who only tamper with a *small fraction* of the random bits. In fact, previous works on *resettable cryptography* [CGGM00] can be interpreted as achieving tamper resilience against adversaries who tamper with *all* of the randomness of the honest parties by *resampling* the randomness of the honest parties and executing them again and again. This is incomparable to our model, since our adversary does not have control over the honest parties' execution (to run them again), but is more powerful in that it could change the value of some of the random bits.

Online Tampering. Let $0 < p < 1$ be the parameter describing the "power" of adversary (which defines the fraction of tampered bits). It still remains to clarify how the tampering is done over these bits. The first naive model would allow the adversary to tamper with a p fraction of the bits *after* all the bits are sampled by the system (and, thus, are known to the virus as well). However, this is not realistic since the sequence of random bits used by the system could always be sampled in an *online* manner; namely, the system could sample the i-th random bit whenever it needs it and might use it "on the fly". Therefore, a tampering adversary also needs to tamper with them one-by-one, efficiently, and in an on-line fashion.

More precisely, we consider a so-called *p-tampering attack*, where the adversary gets to tamper with the random tape of the honest players as follows. The randomness of honest parties is generated bit-by-bit, and for each generated bit x_i the efficient tampering circuit gets to tamper with it with independent probability p having only knowledge of previously generated random bits $x_1, x_2, \ldots, x_{i-1}$ (but not the value of the random bits tossed in the future). Roughly speaking, requiring security with respect to p-tampering attacks amounts to requiring that security holds even if the honest players' randomness comes from a *computationally efficient* analog of a Santha-Vazirani (SV) source [SV86]. Recall that a random variable $X = (X_1, \ldots, X_n)$ over bit strings is an SV source with parameter δ if for every $i \in [n]$ and every $(x_1, \ldots, x_i) \in \{0,1\}^i$, it holds that $\delta \leq \mathsf{P}[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 1 - \delta$. It is easy to see that the random variable resulting from performing a p-tampering attack on a uniform n-bit string is an SV source with parameter $(1-p)/2$; in fact, any SV source is equivalent to performing a *computationally unbounded* p-tampering attack on a uniform n-bit string.

The main focus of this work is on the following question:

Can security be achieved under p-tampering attacks?

1.1 Our Results

Our main result is a strong negative answer to the question above for a variety of basic cryptographic primitives. A p-tampering attacker can break all of the following with advantage $\Omega(p)$: **(1)** the security of any CPA-secure (public-key or private-key) encryption scheme, **(2)** the zero-knowledge property of any efficient-prover proof (or argument) system for nontrivial languages, **(3)** the hiding property of any commitment scheme, and **(4)** the security of any protocol for computing a "nontrivial" finite function. More formally, we prove the following theorems.

Theorem 1.1 (Impossibility of Encryption)**.** *Let Π be any CPA-secure public-key or private-key encryption scheme. Then a p-tampering attacker can break the security of Π with advantage $\Omega(p)$. Moreover, the attacker only tampers with the random bits of the* encryption *(not the key-generation)* without *knowing the encrypted message.*

Theorem 1.2 (Impossibility of Zero-Knowledge)**.** *Let (P, V) be an efficient prover proof/argument system for a language $L \in \mathsf{NP}$ such that the view of any p-tampering verifier can be simulated by an efficient simulator with indistinguishability gap $o(p)$, then the language L is in* BPP.

Theorem 1.3 (Impossibility of Commitments)**.** *Let (S, R) be a bit-commitment scheme. Then, either an efficient malicious sender can break the biding with advantage $\Omega(p)$ (without tampering), or an efficient malicious p-tampering receiver can break the hiding with advantage $\Omega(p)$.*

Following [GLM+04] we consider two-party functions $f \colon \mathcal{D}_1 \times \mathcal{D}_2 \mapsto \mathcal{R}$ where only one player gets the output. A function f is called trivial in this context, if there is a deterministic single-message (i.e., only one player speaks) protocol for computing f that is statistically secure.

Theorem 1.4 (Impossibility of Secure Computation)**.** *The security of any protocol for computing a two-party non-trivial function can be broken with advantage $\Omega(p)$ through a p-tampering attack.*

Positive Results. We complement the above negative results by demonstrating some initial positive results: Assuming the existence of one-way functions, for any $p = n^{-\alpha}$, where $\alpha > 0$ is a constant and n is the security parameter, every implementation of signature schemes, identification protocols, and witness hiding protocols can be made resilient against p-tampering attackers. We also present a relaxed notion of semantic security for encryption schemes that can be achieved under $n^{-\alpha}$-tampering attacks. We show that for these primitives, security holds even if the randomness source "min-entropy loss" of at most $O(\log n)$. We next show how to use PRGs to ensure that a tampering attacker will only be able to decrease the overall (pseudo) min-entropy by $O(\log n)$. The above mentioned primitives already imply the existence of one-way functions [IL89], thus preventing against $n^{-\alpha}$-tampering attacks can be achieved for these primitives unconditionally. Finally, we present positive results for tamper-resilient key-agreement and secure multi-party computation in the presence of (at least) two honest players. We provide the details on this in Section 6.

1.2 Related Work

Subliminal Channels and Kleptography. Cryptographic research on "subliminal channels" [Sim94] and the related field of "kleptography" [YY96] study whether a cryptographic scheme can be "misused" for a purpose other than the original purpose it was designed for (e.g., by putting an undetectable trapdoor in the systems). The existence of subliminal channels between "outside" and "inside" adversaries could be a huge security concern in certain scenarios such as voting schemes [FB09]. Our Theorem 1.1 shows that *any* efficient encryption scheme always has a subliminal channel between an outsider adversary and an insider virus who is (only) able to tamper with the randomness of the encryption device no matter how the encryption algorithm tries to "detect" a virus who is signaling a bit of information to the adversary.

Undetectablity and Algorithm Substitution Attacks. As discussed above, our tampering attacks are *undetectable* in the sense that they would pass any extra check done over the randomness, as part of scheme, as long as these extra checks do not harm the completeness of the scheme. Perhaps surprisingly, our tampering attack could be set to remain perfectly undetachable in the sense that it generates a tampered random seed which is *uniformly* distributed, just like the original un-tampered randomness. As a result, the behavior of the algorithm using the tampered random seed remains identical to the when no attack occurs. More formally, we present tampering attacks against randomness of encryption, commitment, and secure computation schemes, breaking them with advantage $1/\operatorname{poly}(n)$ by tampering with only *one* bit of randomness that keeps the distribution of the tampered bits uniformly distributed (see Remark 5.11.)

The beautiful work of Bellare et al. [BPR14] studies *algorithm substitution* attacks against symmetric key encryption schemes in which the "big brother" substitutes the encryption algorithm E (that is assumed to contain the encryption key) with its own variant E' with the goal of learning nontrivial information about the encrypted message M through accessing to the ciphertext (encrypted by E') transmitted over the public channel. [BPR14] presented attacks against specific popular encryption schemes. Our tampering attacks against symmetric encryption schemes could be interpreted as algorithm substitution attacks in which the new substituted encryption algorithm is the affected encryption algorithm by adversary's virus that tampers with its randomness. Using our completely undetectable tampering attack, we could obtain algorithm substitution attacks against *any* semantically secure encryption scheme.

Tampering with Randomness vs. Imperfect Randomness. Our negative results are closely related to the impossibility result of Dodis et al. [DOPS04] on the "impossibility of cryptography with imperfect randomness", where the security of cryptographic primitives are analyzed assuming that the honest parties only have access to randomness coming from an SV source (as opposed to the randomness being perfectly uniform). [DOPS04] present several strong impossibility results for secure realizability of cryptography primitives in a setting where players only have access to such imperfect randomness. The SV sources they consider for their impossibility results, however, may not be efficiently computable.

The key-difference between tamper-resilient security in our setting and security with imperfect randomness is that we restrict to randomness sources that are efficiently sampled through an (online) p-tampering attack; thus achieving tamper-resilient security becomes easier than resilience to imperfect randomness. Note that even if one can *efficiently* sample from the sources employed by [DOPS04], that still does not solve our main question, because by sampling fresh randomness for the system the adversary is indeed tampering with *all* of the random seed. As we discussed above, however, in such scenario the adversary can always fix the randomness to zero and so we are essentially down to the deterministic case. Another, perhaps less important difference, is that for primitives with simulation-based security, we allow the simulator to depend on the p-tampering attacker, whereas [DOPS04] the simulator must work for any randomness source; this further makes achieving tamper-resilient security easier than resilience to imperfect randomness.

Comparison with [KK08]. The work of Kamara and Katz studies the security of private-key encryption schemes in a related, but quite different model that also lets the adversary tamper with the randomness of the encryption. Indeed [KK08] allows a CPA attacker A to *arbitrarily* choose the randomness r of their *CPA encrypting oracle*, but their attacker A does not have any control over the randomness used to encrypt the actual challenge ciphertext. They showed how to achieve security in this model, while our result, prove in our model, is a *negative* one; thus the two models are incomparable. In particular, even though the tampering happens in the model of [KK08], the adversary cannot make this tampering based on private data stored on the encrypting device. On the other hand, in our model, the only actual tampering attack is launched over the randomness used for encrypting the challenge ciphertext, while this randomness remains intact in [KK08].

Preserving Secrets vs. Achieving Security. Some of the previous works on tamper-resilient cryptography focused on compiling a circuit C holding a secret into a new circuit C' that hides the secret even if the attacker gets to tamper with the computation of C'. In contrast, in this work we focus on whether one can preserve the security of cryptographic schemes under tampering attacks. The simulation property of tamper-resilient compilers do not necessarily guarantee that if the sender algorithm is compiled into a "tamper-resilient" version, then the encryption scheme is tamper-resilient. This is due to the fact that the simulation property of those compilers only guarantee that an attacker cannot learn more from tampering with the sender strategy than it could have with black-box access to it. But in the case of encryption schemes, it is actually the *input* to the algorithm (i.e., the message to be encrypted) that we wish to hide (as opposed to some secret held by the algorithm). In the following, we review these related works in two categories: aiming resilience against tampering with memory and against tampering with the computation.

Tampering with Memory. The work of Gennaro et al. [GLM+04] was the first to formally study tamper resilience from an algorithmic (as opposed to a hardware-based) point of view. In their model they deal with a cryptographic algorithm $G(s, x)$ holding a secret s that given an input x outputs some y (e.g., G could be a signing or a decryption algorithm). The adversary gets hold to a box running G and is allowed to perform tampering attacks over the "memory" of G that holds the secret state of G (which in particular includes the secret s). The adversary's goal is to break the security of G with respect to the *original* value of s (e.g., forge a signature). Gennaro et al. showed that any cryptographic protocol that can be "tested" for malfunctioning (e.g., a signature box can be publicly tested by verifying the signatures it produces) can be broken by an adversary that only performs "resetting" attacks over the bits of the secret state. The idea is to test the bits of s one by one. In i^{th} step the adversary sets the bit s_i of s to zero. Then if the newly tampered value of s passes the malfunctioning test (e.g., can be used to sign messages successfully), it means that an acceptable value for the first i bit of s is discovered; otherwise we set s_i to one.

Gennaro et al. also present a positive result based on the assumption that there is a tamper proof component (e.g., some "circuitry") available. This way, a trusted party who generates the circuit of G would sign the original secret state of G and hardwire this signature together with the corresponding verification key into the tamper proof part of the code of G. The execution of G will always start by first testing the signature of its own secret state. If the signature verification passes it will use this state to run its algorithm, and otherwise it will "self-destruct". The work of Dziembowski et al. [DPW18] extended the positive result of [GLM+04] and achieved information theoretic security for a more restricted class of tampering functions through introducing and constructing "non-malleable" codes. These codes, then, are used to encode and decode the internal secret state of G before and after using it (instead of self-destruct). The works of [KKS11, LL12] go beyond only tamper resilience by achieving also *leakage resilience* when both of tampering and leakage are performed only over the memory. Finally the work of Choiet al. [CKM11] studies the tamper resilience in the context of universal composability and studies *affine* tampering functions.

Tampering with Computation. Boneh, DeMillo and Lipton [BDL97] showed that introducing minor random errors during the computation of some implementations of certain cryptographic schemes can be exploited by the adversary to a large extent and break the scheme completely. The result was rather shocking, since some natural implementations would completely break down by a *single* call along with a random tampering performed. Ishai et al. [IPSW06] took on the positive side and showed how to make a circuit that is already accessible by the adversary as an input-output functionality secure against being tampered with up to t wires in every input-output call. The security here means that the view of any such adversary can be simulated by a simulator who does *not* tamper with the circuit and only uses it as a black-box. Thus the compiler of [IPSW06] shows how to keep a key inside a circuit in a secure way against tampering (e.g., a decryption circuit, or a signing circuit). The tampering functions here are restricted in the following sense: they only choose a set of t wires, and for each of them decide whether to set them to zero or one, or to flip their values. Our positive results (see Section 6), however, apply even to the case that the adversary can observe the whole internal state of the tampered algorithm, and choose the value of the tampered "random" bits based on that. The subsequent work of Faust et al. [FPV11] followed the framework of [IPSW06] and extended their work to the setting that there is no fixed upper bound t on the number of tampered wires in each round of executing the tampered algorithm, but there is a constant probability $\delta > 0$ that each chosen wire remains untampered.

1.3 Our Techniques

Our main technical contribution is the development of new methods for biasing Boolean, and more generally, bounded-value functions, using a p-tampering attack.

1.3.1 Biasing Bounded-Value Functions

Our first (negative) result proves an efficient version of the Santha-Vazirani theorem: Any balanced (or almost balanced) efficiently computable Boolean function f can be biased by $\Omega(p)$ through an efficient p-tampering attack.

Specifically, let U_n denote the uniform distribution over $\{0,1\}^n$ and let $U_n^{\mathsf{Tam},p}$ denote the distribution obtained after performing a p-tampering attack on U_n using a tampering algorithm Tam; more precisely, let $U_n^{\mathsf{Tam},p} = (X_1, \ldots, X_n)$ where with probability $1-p$, X_i is a uniform random bit, and with probability p, $X_i = \mathsf{Tam}(1^n, X_1, \ldots, X_{i-1})$.

Theorem 1.5. (Biasing Boolean Functions). *There exists an oracle machine* Tam *with input parameters* n *and* $\varepsilon < 1$ *that runs in time* $\mathrm{poly}(n/\varepsilon)$ *and for every* $n \in N$ *and* $\varepsilon \in (0,1)$*, every Boolean function* $f : \{0,1\}^n \to \{-1,1\}$*, and every* $p < 1$*, for* $\mu = \mathsf{E}[f(U_n)]$ *it holds that*

$$\mathsf{E}[f(U_n^{\mathsf{Tam}^f,p})] \geq \mu + p \cdot (1 - |\mu| - \varepsilon).$$

The tampering algorithm Tam is extremely simple and natural; it just greedily picks the bit that maximizes the bias at every step. More precisely, $\mathsf{Tam}^f(x_1, \ldots, x_{i-1})$ estimates the value of

$$\mathsf{E}[f(x_1, \ldots, x_{i-1}, b, U_{n-i})]$$

for both of $b = 0$ and $b = 1$ by sampling, and sets x_i to the bit b with larger estimated expectation.

Theorem 1.5 suffices for our impossibility result for tamper-resilient zero-knowledge. For all our remaining impossibility results, however, we need a more general version that also deals with bounded value functions $f : \{0,1\}^n \to [-1,1]$. Our main technical theorem provides such a result.

Theorem 1.6. (Main Technical Theorem: Biasing Bounded-Value Functions). *There exists an efficient oracle machine* Tam *such that for every* $n \in N$*, every bounded-value function* $f : \{0,1\}^n \to [-1,1]$*, and every* $p < 1$*,*

$$\mathsf{E}[f(U_n^{\mathsf{Tam}^f,p})] \geq \mathsf{E}[f(U_n)] + \Omega(p \cdot \mathrm{Var}[f(U_n)]).$$

Note that in Theorem 1.6 the dependence on the variance of f is necessary because f may be the constant function $f(x) = 0$, whereas for the case of balanced Boolean functions this clearly cannot happen.

The greedy algorithm does not work in the non-Boolean case anymore. The problem, roughly speaking, is that a greedy strategy will locally try to increase the expectation, but that might lead to choosing a wrong path. As a "counter-example" consider a function f such that: conditioned on $x_1 = 0$ f is a constant function ε, but conditioned on $x_1 = 1$, f is a Boolean function with average $-\varepsilon$. For such f, the greedy algorithm will set $x_1 = 0$ and achieves bias at most ε, while by choosing $x_1 = 1$ more bias could be achieved. To circumvent this problem we use a "mildly greedy" strategy: we take only one sample of $f(\cdot)$ by choosing $x'_i, x'_{i+1}, \ldots, x'_n$ at random ($x_1, \ldots, x_{i-1}$ are already fixed). Then, we keep the sampled x'_i with probability proportional to how much the output of f is close to our "desired value", and flip the value of x'_i otherwise.

More precisely, $\mathsf{Tam}(1^n, x_1, \ldots, x_{i-1})$ proceeds as follows:

- Samples $(x'_i, x'_{i+1}, \ldots, x'_n) \leftarrow U_{n-i+1}$ and compute $y = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$.
- Sample $\mathsf{Tam}(1^n, x_1, \ldots, x_{i-1})$ from a Boolean random variable with expectation $y \cdot x'_i$. Namely, output x'_i with probability $\frac{1+y}{2}$, and $-x'_i$ with probability $\frac{1-y}{2}$.

Note that our mildly greedy strategy is even easier to implement than the greedy one: to tamper with each bit, it queries f *only once*.

1.3.2 Impossibility Results for Tamper-Resilient Cryptography

We employ the biasing algorithms of Theorems 1.5 and 1.6 to obtain our negative results using the following blue-print: We first prove a *computational version* of the "splitting lemma" of [DOPS04] (Lemma 1.7 below which follows from Corollary 3.2 in [DOPS04]). Then we will use the same arguments as those of [DOPS04] to derive our impossibility results.

Lemma 1.7 (Proved in [DOPS04])**.** *Let $f_0, f_1 \colon \{0,1\}^{\mathrm{poly}(m)} \mapsto \{0,1\}^m$ be two efficient functions such that $\Pr[f_0(U_m) \neq f_1(U_m)] \geq 1/\mathrm{poly}(n)$. Then there is a Santha-Vazirani source X with parameter $1/2 - 1/\mathrm{poly}(n)$ such that $f_0(X)$ is computationally distinguishable from $f_1(X)$.*

We use our Theorem 1.6 to prove the following computational version of Lemma 1.7 which allows one to distinguish the functions f_0, f_1 by tampering with their random input.

Lemma 1.8 (Computational Splitting Lemma)**.** *Let f_0 and f_1 be two efficient functions from $\{0,1\}^m$ to $\{0,1\}^{\mathrm{poly}(m)}$ and $\Pr[f_0(U_m) \neq f_1(U_m)] \geq \varepsilon > 1/\mathrm{poly}(m)$. Then one can efficiently find $\mathrm{poly}(m)$-size circuits f and (tampering circuit) Tam such that*

$$\Pr[f(f_1(U_n^{\mathsf{Tam},p})) = 1] \geq \Pr[f(f_0(U_n^{\mathsf{Tam},p})) = 1] + \Omega(\varepsilon \cdot p).$$

Proof Outline. We derive Lemma 1.8 from Theorem 1.6 as follows. We use Theorem 1.6 to bias the *difference* function $g_f(x) = f(f_1(x)) - f(f_0(x))$ (with domain $\{-1, 0, +1\}$) towards 1 by a tampering circuit Tam. It is easy to see that if f is Boolean, doing this is equivalent to the goal of Lemma 1.8. We show that if one samples f from a family of pairwise independent Boolean functions, then the resulting function $g_f(\cdot)$ has sufficient variance as needed by Theorem 1.6.

We use our Lemma 1.8 similar to the way Lemma 1.7 is employed in [DOPS04] to derive our impossibility results for tamper resilient: encryption schemes, commitments, and two-party secure function evaluation protocols. For all these primitives an adversary uses Lemma 1.8 to generate a tampering circuit Tam that later on lets him distinguish the corresponding challenges (generated using the tampered randomness) and break the security.

Zero-Knowledge. Zero-knowledge proofs in the setting of [DOPS04] require a *universal* simulator that simultaneously handles a large class of imperfect randomness sources. We can also use our Lemma 1.8 to rule out such tamper-resilient zero-knowledge proofs. In the computational setting, however, it is the malicious verifier who generates the bad source of randomness, and so we shall allow the simulator to depend on the tampering circuit as well. Thus, the simulator in our setting has more power. This prevents us from being able to apply Lemma 1.8 directly.

We proceed in using the following high level outline. In the first step, we generalize a result by Goldreich and Oren [GO94] showing that non-trivial zero-knowledge protocols cannot have deterministic provers. Our generalization to [GO94] shows that non-trivial zero-knowledge protocols

require having prover messages with min-entropy $\omega(\log n)$. This means that the verifier can apply a (seeded) randomness extractor to the transcript and obtain one almost unbiased bit. In a second step, we show how to use (the proof of) Theorem 1.5 to tamper with the prover's randomness so as to signal bits of the witness to the verifier.

This outline, however, oversimplifies: is it not the case that every non-trivial zero-knowledge protocol requires the prover messages to have min-entropy $\omega(\log n)$; in fact, for some "easy" instances, the prover may not communicate at all. Rather, we demonstrate that an "instance-based" version of the min-entropy extension of the Goldreich and Oren [GO94] theorem holds, and using it we can prove that *either* the prover's messages have high min-entropy (and thus the witness can be leaked to the verifier), or the instance can be decided "trivially". It follows that in either case, we can correctly decide the instance and thus the language must be trivial.

2 Preliminaries

Notation. We use $x \leftarrow X$ to denote the sampling of x from the random variable (or rather distribution) X. By U_n we refer to the uniform distribution over $\{0,1\}^n$. We use the notation that multiple uses of the same random variable in a probability or expectation refers to the same instantiation. For example $\Pr[f(U_n) = g(U_n)] = \Pr[f(U_n) = g(U_n)]$. On few occasions, to emphasize over the random variable over which the probability or expectation is defined, we might use the notation $\Pr_{x \leftarrow X}[\cdot]$ or $\mathsf{E}_{x \leftarrow X}[\cdot]$. By a *negligible function*, denoted as $\mathrm{negl}(n)$, we mean any function which is of the form $n^{-\omega(1)}$. By a *noticeable* function $f(n)$ we mean any function of the form $n^{\Omega(1)}$. We use the notation PPT do denote *probabilistic polynomial time*. We might use the terms "efficient" and PPT interchangeably. For interactive algorithms (A, B) and $C \in \{A, B\}$, by $\mathsf{View}_C\langle A, B\rangle(x)$ we denote the view of C in an execution of the interaction between A, B, where this view includes the common input, the private randomness of C, and the messages exchanged. In addition, by $\langle A, B\rangle(x)$ we denote the output of the interaction between (randomized) interactive algorithms A, B over common input x. By $\Delta(X, Y)$ we denote the *statistical distance* between the random variables X, Y defined as $\frac{1}{2}\sum_x |\mathsf{P}[X = x] - \mathsf{P}[Y = x]|$. By $\mathrm{H}_\infty(X)$ we denote the *min-entropy* of the random variable X defined as the largest k such that $\mathsf{P}[X = a] \leq 2^{-k}$ for every a. We call a sequence $\{\mathbf{X}_x\}$ of random variables indexed by $x \in I \subseteq \{0,1\}^*$ an *ensemble* of random variables. We call two ensembles of random variables $\{\mathbf{X}_x\}$ and $\{\mathbf{Y}_x\}$ (with the same index set) $\alpha(|x|)$-*indistinguishable* if for every polynomial $p(\cdot)$ and every sequence of Boolean circuits C_n of size $p(n)$, we have $\Delta(C_{|x|}(x, \mathbf{X}_x), C_{|x|}(x, \mathbf{Y}_x)) \leq \alpha(|x|)$. we use the term *computationally indistinguishable* to refer to the case where $\alpha(\cdot) = \mathrm{negl}(\cdot)$. For function $f\colon \{0,1\}^n \mapsto \mathbb{R}$ by $\mathsf{E}[f]$ and $\mathrm{Var}[f]$ we mean $\mathsf{E}[f(U_n)]$ and $\mathrm{Var}[f(U_n)]$.

Definition 2.1 (SV Sources)**.** *The random bit string $X = (X_1, \ldots, X_n)$ is a Santha-Vazirani (SV) source with parameter δ if for every $i \in [n]$ and every $(x_1, \ldots, x_i)$, it holds that $\delta \leq \mathsf{P}[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 1 - \delta$. It is easy to see that $\mathrm{H}_\infty(X) \geq n \cdot \lg(1 - \delta)$ holds for any SV source $X = (X_1, \ldots, X_n)$ with parameter δ.*

3 Biasing Functions via Online Tampering

In this section we study how much the output of a bounded function can be biased through a tampering attack. First we formally define an online tampering process and a tampering source of

randomness (as a result of an online tampering attack performed on a uniform distribution).

Definition 3.1. *A distribution $X = (X_1, \ldots, X_n)$ over $\{-1,1\}^n$ is an (efficient) p-tampering source if there exists an (efficient) tampering algorithm Tam such that X can be generated in an online fashion as follows: For $i = 1, \ldots, n$,*

$$X_i = \begin{cases} \mathsf{Tam}(1^n, X_1, \ldots, X_{i-1}) & \text{with probability } p, \\ U_1^i & \text{with probability } 1-p, \end{cases}$$

where U_1^i denotes a uniformly random bit over $\{-1,1\}$. In other words, with probability p, Tam gets to tamper the next bit with the knowledge of the previous bits (after the tampering)[2]. The tampering algorithm Tam might also receive an auxiliary input and use it in its tampering strategy.[3] We use $U_n^{\mathsf{Tam},p}$ to denote the p-tampered source obtained by the above tampering process with tampering algorithm Tam.

Note that in the definition above, the tampering algorithm Tam might be completely oblivious to the parameter p. By referring to Tam as a *p-tampering* algorithm, we emphasize on the fact that Tam's algorithm might depend on p.

Remark 3.2. *Every p-tampering source is also a Santha-Vazirani source [SV86] with parameter $\delta = (1-p)/2$. In fact, it is not hard to see that without the efficiency consideration, the two notions are equivalent.*

3.1 Preliminaries: Calculating the Effect of a Single Variable

Since the constants that emerge in our analysis are standard Fourier notation, in the following we will first recall the Fourier notation before using it in our presentation of the proofs. However, we emphasize that no knowledge of Fourier analysis is needed for understanding our proofs.

Recall that the Fourier coefficients of any function $f : \{-1,1\}^n \to [-1,1]$ are indexed by the subsets S of $[n]$ and are defined as $\hat{f}(S) := \mathsf{E}[f(U_n)\chi_S(U_n)]$, where $\chi_S(x) := \prod_{i \in S} x_i$. Note that the Fourier coefficient of the empty set $\hat{f}(\emptyset)$ is simply the expectation $\mathsf{E}[f(U_n)]$.

For every prefix $x_{\leq i} = (x_1, \ldots, x_i)$, let $f_{x_{\leq i}} : \{-1,1\}^{n-i} \to [-1,1]$ be the restriction of f on $x_{\leq i}$, i.e., $f_{x_{\leq i}}(x_{i+1}, \ldots, x_n) := f(x_1, \ldots, x_n)$. We note that the variables of $f_{x_{\leq i}}$ are named $(x_{i+1}, \ldots, x_n)$ and thus the Fourier coefficients of $f_{x_{\leq i}}$ are $\hat{f}_{x_{\leq i}}(S)$'s with $S \subseteq \{i+1, \ldots, n\}$. The following basic identity can be proved by straightforward calculation.

$$\hat{f}_{x_1}(\emptyset) = \hat{f}(\emptyset) + \hat{f}(\{1\}) \cdot x_1. \tag{1}$$

Recall that $\hat{f}(\emptyset)$ and $\hat{f}_{x_1}(\emptyset)$ are simply expectations. One interpretation of the above identity is that $\pm\hat{f}(\{1\})$ is the change of expectation when we set $x_1 = \pm 1$. This is thus useful for analyzing the bias introduced as the result of a tampering attack.

[2] In a stronger variant of tampering attacks, the attacker might be completely stateful and memorize the original values of the previous bits before and after tampering and also the places where the tampering took place, and use this extra information in its future tampering. Using the weaker stateless attacker of Definition 3.1 only makes our negative results stronger. Our *positive* results hold even against stateful attackers.

[3] The auxiliary input could, e.g., be the information that the tampering algorithm receives about the secret state of the tampered party; this information might not be available at the time the tampering circuit is generated.

Using the above identity with a simple induction, we can express $f(x)$ as a sum of Fourier coefficients of restrictions of f. Namely, $f(x)$ equals to the expectation $\hat{f}(\emptyset)$ plus the changes in expectation when we set x_i bit by bit.

Lemma 3.3. *For every $x \in \{-1,1\}^n$, it holds that $f(x) = \hat{f}(\emptyset) + \sum_{i=1}^{n} \hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i$.*

Proof. By expanding $\hat{f}_{x_{\leq j}}(\emptyset) = \hat{f}_{x_{\leq j-1}}(\emptyset) + \hat{f}_{x_{\leq j-1}}(\{j\}) \cdot x_j$, (implied by Equation (1)) and a simple induction on j it follows that:

$$f(x) = \hat{f}_{x_{\leq j}}(\emptyset) + \sum_{i=j+1}^{n} \hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i,$$

which proves the lemma. □

As a corollary, the above lemma implies that the sum of Fourier coefficients (of restrictions of f) in absolute value is at least $|f(x)| - |\hat{f}(\emptyset)|$.

Corollary 3.4. *For every $x \in \{-1,1\}^n$, it holds that $\sum_{i=1}^{n} \left|\hat{f}_{x_{\leq i-1}}(\{i\})\right| \geq |f(x)| - |\hat{f}(\emptyset)|$.*

Proof. We have

$$\sum_{i=1}^{n} \left|\hat{f}_{x_{\leq i-1}}(\{i\})\right| = \sum_{i=1}^{n} \left|\hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i\right| \geq \left|\sum_{i=1}^{n} \hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i\right| = |f(x) - \hat{f}(\emptyset)| \geq |f(x)| - |\hat{f}(\emptyset)|$$

where both inequalities follow by the triangle inequality, and the second equality uses Lemma 3.3. □

3.2 The Boolean Case

A seminal result by Santha and Vazirani [SV86] shows that for every balanced Boolean function f (e.g., a candidate "extractor"), there exists a p-tampering source X that biases the output of f by at least p. We now present a strengthening of this result that additionally shows that if the function f is efficiently computable, then the source X could be an efficient p-tampering one (and only needs to use f as a black box). In the language of extractors, our result thus proves a strong impossibility result for deterministic randomness extraction from "efficient" Santha-Vazirani sources. Our proof of the generalized result is quite different (and in our eyes simpler) than classic proofs of the Santha-Vazirani theorem and may be of independent interest.

In fact, we present two different proofs. The first one achieves optimal bias p for balanced f, whereas the second uses an extremely simple "*mild greedy*" tampering algorithm that makes only a *single* query to f and achieves bias $p/3$ for balanced f.

Theorem 1.5 (Restated)**.** (Biasing Boolean Functions). *There exists an oracle machine* Tam *with input parameters n and $\varepsilon < 1$ that runs in time* $\mathrm{poly}(n/\varepsilon)$ *and for every $n \in N$ and $\varepsilon \in (0,1)$, every Boolean function $f : \{0,1\}^n \to \{-1,1\}$, and every $p < 1$, for $\mu = \mathsf{E}[f(U_n)]$ it holds that*

$$\mathsf{E}[f(U_n^{\mathsf{Tam}^f,p})] \geq \mu + p \cdot (1 - |\mu| - \varepsilon).$$

Proof of Theorem 1.5. Let us first present a proof with an inefficient tampering algorithm achieving bias $p \cdot (1 - |\mu|)$; next, we show how to make it efficient while not loosing much in bias. On input $x_{\leq i-1} = (x_1, \ldots, x_{i-1})$, Tam sets $x_i = \mathrm{sgn}(\hat{f}_{x_{\leq i-1}}(\{i\}))$. By Equation (1), $\hat{f}_{x_{\leq i-1}}(\{i\})$ corresponds to the change in expectation of $f_{x_{\leq i-1}}$ when setting the value of x_i. This amounts to greedily choosing the x_i that increases the expectation. Let $X = U_n^{\mathsf{Tam},p}$. By applying Lemma 3.3 and the linearity of expectations, we have

$$\mathsf{E}[f(X)] = \hat{f}(\emptyset) + \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\}) \cdot X_i\right] = \hat{f}(\emptyset) + \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\}) \cdot \mathsf{E}[X_i | X_{\leq i-1}]\right].$$

Since Tam tampers with the i'th bit with independent probability p, therefore

$$\mathsf{E}[X_i | X_{\leq i-1}] = p \cdot \mathrm{sgn}(\hat{f}_{X_{\leq i-1}}(\{i\})$$

and so it holds that

$$\mathsf{E}[f(X)] = \hat{f}(\emptyset) + p \cdot \sum_{i=1}^{n} \mathsf{E}\left[\left|\hat{f}_{X_{\leq i-1}}(\{i\})\right|\right] = \hat{f}(\emptyset) + p \cdot \mathsf{E}\left[\sum_{i=1}^{n} \left|\hat{f}_{X_{\leq i-1}}(\{i\})\right|\right] \geq \hat{f}(\emptyset) + p \cdot (1 - \hat{f}(\emptyset))$$

where the last inequality follows by Corollary 3.4.

Note that the above tampering algorithm Tam in general is not efficient since computing $\hat{f}_{x_{\leq i-1}}(\{i\})$ exactly may be hard. However, we show that Tam may approximate $\hat{f}_{x_{\leq i-1}}(\{i\})$ using $M = Theta(\frac{n^2}{\varepsilon^2} \cdot \log \frac{n}{\varepsilon})$ samples, and set x_i according to the sign of the approximation of $\hat{f}_{x_{\leq i-1}}(\{i\})$, while still inducing essentially the same bias. This clearly can be done efficiently given oracle access to f. As before, let $X = U_n^{\mathsf{Tam}^f,p}$ denote the corresponding p-tampering source. To lower bound $\mathsf{E}[f(X)]$, we note that the only difference from the previous case is that $\mathsf{Tam}(1^n, x_{\leq i-1})$ is no longer always outputting $\mathrm{sgn}(\hat{f}_{x_{\leq i-1}}(\{i\}))$. Nevertheless, we claim that for every $x_{<i}$ it holds that

$$\hat{f}_{x_{\leq i-1}}(\{i\}) \cdot \mathsf{E}[X_i | X_{\leq i-1} = x_{\leq i-1}] \geq p \cdot \left(|\hat{f}_{x_{\leq i-1}}(\{i\})| - \varepsilon/n\right)$$

since either (i) $|\hat{f}_{x_{\leq i-1}}(\{i\})| \geq \varepsilon/2n$ in which by a Chernoff bound Tam outputs $\mathrm{sgn}(\hat{f}_{x_{\leq i-1}}(\{i\}))$ with probability at least $1 - \varepsilon/2n$, or (ii) $|\hat{f}_{x_{\leq i-1}}(\{i\})| < \varepsilon/2n$ in which case the inequality holds no matter what Tam outputs since $|\mathsf{E}[X_i | X_{\leq i-1} = x_{\leq i-1}]| \leq p$. A lower bound on $\mathsf{E}[f(X)]$ then follows by the same analysis as before.

$$\mathsf{E}[f(X)] \geq \hat{f}(\emptyset) + p \cdot \sum_{i=1}^{n} \mathsf{E}\left[\left|\hat{f}_{X_{\leq i-1}}(\{i\})\right| - \varepsilon/n\right] \geq \mu + p \cdot (1 - |\hat{f}(\emptyset)| - \varepsilon).$$

□

Before presenting the second proof, we state the following lemma, which follows similarly to lemma 3.3, but instead it relies on a squared version of Equation (1).

Lemma 3.5. *For all* $x \in \{-1, 1\}^n$, $f(x)^2 = \hat{f}(\emptyset)^2 + \sum_{i=1}^{n} \left[\hat{f}_{x_{\leq i-1}}(\{i\})^2 + 2\hat{f}_{x_{\leq i-1}}(\emptyset)\hat{f}_{x_{\leq i-1}}(\{i\})x_i\right]$.

Proof. Squaring Equation (1), and recalling that $x_i^2 = 1$ since $x_i \in \{-1, 1\}$, for every f we obtain

$$\hat{f}_{x_1}(\emptyset)^2 = \hat{f}(\emptyset)^2 + \hat{f}(\{1\})^2 + 2\hat{f}(\emptyset) \cdot \hat{f}(\{1\}) \cdot x_1.$$

By expanding

$$\hat{f}_{x_{\leq j}}(\emptyset)^2 = \hat{f}_{x_{\leq j-1}}(\emptyset)^2 + \hat{f}_{x_{\leq j-1}}(\{j\})^2 + 2\hat{f}_{x_{\leq j-1}}(\emptyset) \cdot \hat{f}_{x_{\leq j-1}}(\{j\}) \cdot x_j$$

and using a simple induction over j it follows that

$$f(x)^2 = \hat{f}_{x_{\leq j}}(\emptyset)^2 + \sum_{i=j+1}^{n} \left(\hat{f}_{x_{\leq i-1}}(\{i\})^2 + 2\hat{f}_{x_{\leq i-1}}(\emptyset) \cdot \hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i. \right),$$

which proves the lemma. □

We continue to present the second proof, which uses a "*mild greedy*" tampering algorithm that makes only a *single* query to f and achieves bias $p/3$ for balanced f.

Theorem 3.6. *There exists an oracle machine* Tam *that makes a* single *query to its oracle such that for every* $n \in N$*, every Boolean function* $f : \{0,1\}^n \to \{-1,1\}$*, and every* $p < 1$*, for* $\mu = \mathsf{E}[f(U_n)]$ *it holds that*

$$\mathsf{E}[f(U_n^{\mathsf{Tam}^f,p})] \geq \mu + p \cdot (1 - \mu^2)/3.$$

Proof. We consider a *mild greedy tampering algorithm* MTam that on input $x_{\leq i-1} = (x_1, \ldots, x_{i-1})$, samples uniformly random $(x'_i, \ldots, x'_n) \leftarrow U_{n-i+1}$, queries $y = f(x_1, x_{i-1}, x'_i, \ldots, x'_n)$, and outputs $X_i = x'_i$ if $y = 1$ and $X_i = -x'_i$ if $y = -1$. Namely, MTam samples a random completion of $x_{\leq i-1}$ and output the sampled bit x'_i iff the sample evaluates to 1.

Interestingly, this simple mild greedy MTam implicitly "plays the first Fourier coefficient" in expectation in the sense that $\mathsf{E}[\mathsf{MTam}(x_{\leq i-1})] = \hat{f}_{x_{\leq i-1}}(\{i\})$.

Claim 3.7. *For every* $x_{\leq i-1}$*,* $\mathsf{E}[\mathsf{MTam}(x_{\leq i-1})] = \hat{f}_{x_{\leq i-1}}(\{i\})$.

Proof. Let $X_i = \mathsf{MTam}(x_{\leq i-1})$. We have $\mathsf{E}[X_i] = \mathsf{E}[f(x) \cdot x_i] = \hat{f}_{x_{\leq i-1}}(\{i\})$ where the second expectation is over the choice of $x \geq i$ from U_{n-i+1}. □

To analyze MTam, we derive two equalities analogous to that in the proof of Theorem 1.5, and the theorem follows by combining the two equalities. First, since MTam gets to tamper with bit i with independent probability p and $\mathsf{E}[\mathsf{MTam}(x_{\leq i-1})] = \hat{f}_{x_{\leq i-1}}(\{i\})$, by Claim 3.7, we have that $\mathsf{E}[X_i | X_{\leq i-1}] = p \cdot \hat{f}_{X_{\leq i-1}}(\{i\})$. Thus,

$$\mathsf{E}[f(X)] = \hat{f}(\emptyset) + p \cdot \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\})^2\right]. \tag{2}$$

Similarly, by applying Lemma 3.5 and the linearity of expectations, we have

$$\mathsf{E}[f(X)^2] = \hat{f}(\emptyset)^2 + \sum_{i=1}^{n} \left(\mathsf{E}[\hat{f}_{X_{\leq i-1}}(\{i\})^2]\right) + \sum_{i=1}^{n} \left(2\mathsf{E}[\hat{f}_{X_{\leq i-1}}(\emptyset) \cdot \hat{f}_{X_{\leq i-1}}(\{i\}) \cdot \mathsf{E}[X_i | X_{\leq i-1}]]\right).$$

Simplifying using the fact that f is Boolean, the trivial bound $|\hat{f}_{X_{\leq i-1}}(\emptyset)| \leq 1$, and $\mathsf{E}[X_i|X_{\leq i-1}] = p \cdot \hat{f}_{X_{\leq i-1}}(\{i\})$ gives

$$1 \leq \hat{f}(\emptyset)^2 + (1+2p) \cdot \sum_{i=1}^{n} \mathsf{E}[\hat{f}_{X_{\leq i-1}}(\{i\})^2]. \tag{3}$$

Plugging Equation (3) in Equation (2) yields

$$\mathsf{E}[f(X)] \geq \hat{f}(\emptyset) + \frac{p}{1+2p}\left(1 - \hat{f}(\emptyset)^2\right) \geq \mu + p \cdot \left(1 - \mu^2\right)/3,$$

which completes the proof of Theorem 3.6. □

3.3 Tampering with Bounded-Value Functions—The General Case

We further consider the more general case of tampering non-Boolean, bounded-value functions. We present an efficient tampering algorithm that biases the expectation of the function by an amount linear in the variance of the function.

Theorem 1.6 (Restated)**.** (Main Technical Theorem: Biasing Bounded-Value Functions). *There exists an efficient oracle machine* Tam *such that for every* $n \in N$*, every bounded-value function* $f : \{0,1\}^n \to [-1,1]$*, and every* $p < 1$*,*

$$\mathsf{E}[f(U_n^{\mathsf{Tam}^f,p})] \geq \mathsf{E}[f(U_n)] + \Omega(p \cdot \mathrm{Var}[f(U_n)]).$$

In particular, we will prove the above Theorem for constant $1/15.5$ as the constant in the $\Omega(\cdot)$ notation; we did not optimize this constant.

Remark 3.8 (The constant in Theorem 1.6 and the new potential function)**.** *In the original version [ACM+14] of this work, Theorem 1.6 was stated with constant* $1/5$*, but as it was pointed out to us by Mahloujifar [Mah18], the old proof using the original potential function did not hold to imply this constant. In this draft, we had used a different potential function from [MM17] whose attack follows the approach of this work and extends our results (of Theorem 1.6) to the blockwise setting. In particular, in this modification, Lemma 3.11 that provides the sufficient properties for the possible potential functions that can be used in the proof remains intact, however in this draft we instantiated* $g(x)$ *differently based on the similar potential function used in [MM17].*

We prove Theorem 1.5 using *mild greedy* tampering algorithm again. As before, we let MTam take a single sample, and make decision based on the outcome of the sample, but since f is not Boolean, we make randomized decision based on the function value on the sample. Specifically, on input $x_{\leq i-1} = (x_1, \ldots, x_{i-1})$:

- MTam samples random $(x'_i, \ldots, x'_n) \leftarrow U_{n-i+1}$, and computes $y = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$.
- MTam outputs $X_i = x'_i$ with probability $(1+y)/2$, and $X_i = -x'_i$ with probability $(1-y)/2$. Note that X_i has expectation $\mathsf{E}[X_i] = y \cdot x'_i$.

The following claim says that MTam "implicitly plays the first Fourier coefficient" in expectation.

Claim 3.9. *For every* $x_{\leq i-1} = (x_1, \ldots, x_{i-1}) \in \{-1,1\}^{i-1}$*,* $\mathsf{E}[\mathsf{MTam}(x_{\leq i-1})] = \hat{f}_{x_{\leq i-1}}(\{i\})$.

Proof. Let $X_i = \mathsf{MTam}(x_{\leq i-1})$. We have:

$$\mathsf{E}[X_i] = \mathsf{E}[f(x) \cdot x_i] = \hat{f}_{x_{\leq i-1}}(\{i\})$$

where the second expectation is over the choice of $x \geq i$ from U_{n-i+1}. □

Let $X = U_n^{\mathsf{MTam}^f,p}$. Also, let the mean $\mathsf{E}[f(U_n)] = \mu$, the second moment $\mathsf{E}[f(U_n)^2] = \nu$, and the variance $\mathrm{Var}[f] = \sigma^2$ be denoted so. The analysis of the mild greedy algorithm MTam for the non-Boolean case is significantly more involved. We first follow an analogous step in the analysis of Boolean case to derive an inequality between $\mathsf{E}[f(X)]$ and $\mathsf{E}[f(X)^2]$, then rely on a potential function analysis to derive a second inequality relation between $\mathsf{E}[f(X)]$ and $\mathsf{E}[f(X)^2]$, and then derive a lower bound on $\mathsf{E}[f(X)]$ by combining the two. The two inequalities are stated in the following lemmas.

Lemma 3.10.

$$\mathsf{E}[f(X)] - \mu \geq \frac{p}{1+2p} \cdot \left(\mathsf{E}[f(X)^2] - \mu^2\right).$$

Lemma 3.11. *Let $g : \{-1,1\}^n \to [-1,1]$ be an arbitrary function. For every prefix $x_{\leq i} \in \{-1,1\}^i$, define a potential*

$$\Phi(x_{\leq i}) := \hat{f}_{x_{\leq i}}(\emptyset) + \frac{\hat{g}_{x_{\leq i}}(\emptyset)}{2} + \frac{\hat{g}_{x_{\leq i}}(\emptyset)^2}{4},$$

and let $\Phi := \Phi(x_{\leq 0})$. Then it holds that $\mathsf{E}[\Phi(X)] \geq \Phi$.

We first use the above two lemmas to derive Theorem 1.6.

Proof of Theorem 1.6 using Lemmas 3.10 and 3.11. In Lemma 3.11, we use $g(x) = (f(x) - \mu)^2/4$. Note that $g(x) \in [0,1] \subset [-1,+1]$. Let $\mu' = \mathsf{E}[f(X)]$ and $\beta = \mu' - \mu$ be the bias. Then, we have

$$\begin{aligned}
\mathsf{E}[\Phi(X_{\leq n})] &= \mathsf{E}[f(X)] + \mathsf{E}\left[\frac{1}{2} \cdot \frac{(f(X)-\mu)^2}{4} + \frac{1}{4} \cdot \frac{(f(X)-\mu)^4}{16}\right] \\
&< \mu' + \frac{3}{4} \cdot \mathsf{E}\left[\frac{(f(X)-\mu)^2}{4}\right] \\
&= \mu' + \frac{3}{16} \cdot \mathsf{E}\left[f(X)^2 + \mu^2 - 2\mu\mu'\right] \\
(\text{by } \beta = \mu' - \mu) \quad &= \mu' + \frac{3}{16} \cdot \mathsf{E}\left[f(X)^2 - \mu^2 - 2\mu\beta\right] \\
(\text{by Lemma 3.10}) \quad &= \mu' + \frac{3}{16} \cdot \left(\frac{\beta \cdot (1+2p)}{p} - 2\mu\beta\right) \\
&\leq \mu' + \frac{3\beta \cdot (1+2p)}{16p} + \frac{3}{8} \cdot \beta.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\mathsf{E}[\Phi(X_{\leq 0})] &= \mathsf{E}[f(U_n)] + \frac{1}{2} \cdot \mathsf{E}\left[\frac{(f(U_n)-\mu)^2}{4}\right] + \frac{1}{4} \cdot \mathsf{E}\left[\frac{(f(X)-\mu)^2}{4}\right]^2 \\
&= \mu + \frac{1}{8} \cdot \sigma^2 + \frac{1}{64}\sigma^4 > \mu + \frac{1}{8} \cdot \sigma^2.
\end{aligned}$$

Now, by using Lemma 3.11 (i.e., $\mathsf{E}[\Phi(X_{\leq 0})] \leq \mathsf{E}[\Phi(X_{\leq n})]$) and the above bounds we get:

$$\mu' + \frac{3\beta \cdot (1+2p)}{16p} + \frac{3}{8} \cdot \beta > \mu + \frac{1}{8} \cdot \sigma^2$$

and so,

$$\beta \cdot \left(1 + \frac{3(1+2p)}{16p} + \frac{3}{8}\right) > \frac{1}{8} \cdot \sigma^2.$$

Therefore,

$$(\mu' - \mu) \cdot (15.5) = \beta \cdot (15.5) > \beta \cdot \left(8p + \frac{3(1+2p)}{2} + 3p\right) > p \cdot \sigma^2$$

which proves Theorem 1.6. □

Now we prove Lemmas 3.10 and 3.11.

Proof of Lemma 3.10. By applying Lemma 3.3 and the linearity of expectations, we have

$$\mathsf{E}[f(X)] = \hat{f}(\emptyset) + \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\}) \cdot X_i\right] = \hat{f}(\emptyset) + \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\}) \cdot \mathsf{E}[X_i | X_{\leq i-1}]\right].$$

Since MTam gets to tamper with bit i with independent probability p and $\mathsf{E}[\mathsf{MTam}(x_{\leq i-1})] = \hat{f}_{x_{\leq i-1}}(\{i\})$, by Claim 3.9, we have that $\mathsf{E}[X_i | X_{\leq i-1}] = p \cdot \hat{f}_{X_{\leq i-1}}(\{i\})$. Thus,

$$\mathsf{E}[f(X)] = \hat{f}(\emptyset) + p \cdot \sum_{i=1}^{n} \mathsf{E}\left[\hat{f}_{X_{\leq i-1}}(\{i\})^2\right]. \tag{4}$$

Similarly, by applying Lemma 3.5 and the linearity of expectations, we have

$$\mathsf{E}[f(X)^2] = \hat{f}(\emptyset)^2 + \sum_{i=1}^{n} \left(\mathsf{E}[\hat{f}_{X_{\leq i-1}}(\{i\})^2]\right) + \sum_{i=1}^{n} \left(2\mathsf{E}[\hat{f}_{X_{\leq i-1}}(\emptyset) \cdot \hat{f}_{X_{\leq i-1}}(\{i\}) \cdot \mathsf{E}[X_i | X_{\leq i-1}]]\right).$$

Simplifying using the trivial bound $|\hat{f}_{X_{\leq i-1}}(\emptyset)| \leq 1$ and $\mathsf{E}[X_i | X_{\leq i-1}] = p \cdot \hat{f}_{X_{\leq i-1}}(\{i\})$ gives

$$\mathsf{E}[f(X)^2] \leq \hat{f}(\emptyset)^2 + (1+2p) \cdot \sum_{i=1}^{n} \mathsf{E}[\hat{f}_{X_{\leq i-1}}(\{i\})^2]. \tag{5}$$

The lemma follows by combining Equations (4) and (5):

$$\mathsf{E}[f(X)] \geq \hat{f}(\emptyset) + \frac{p}{1+2p}\left(\mathsf{E}[f(X)^2] - \hat{f}(\emptyset)^2\right) = \mu + \frac{p}{1+2p}\left(\mathsf{E}[f(X)^2] - \mu^2\right).$$

□

We now prove Lemma 3.11.

Proof of Lemma 3.11. We show that for every $x_{\leq i-1} \in \{-1,1\}^{i-1}$,

$$\mathsf{E}[\Phi(X_{\leq i}) \mid X_{\leq i-1} = x_{\leq i-1}] \geq \Phi(x_{\leq i-1}).$$

To simplify the notation, let $A = \hat{f}_{x_{\leq i-1}}(\emptyset), a = \hat{f}_{x_{\leq i-1}}(\{i\}), B = \hat{g}_{x_{\leq i-1}}(\emptyset)$, and $b = \hat{g}_{x_{\leq i-1}}(\{i\})$. Using this notation we have,

$$\Phi(x_{\leq i-1}) = A + B/2 + B^2/4.$$

Using Equation (1), we see that

$$\Phi(x_{\leq i}) = \hat{f}_{x_{\leq i}}(\emptyset) + \frac{\hat{g}_{x_{\leq i}}(\emptyset)}{2} + \frac{\hat{g}_{x_{\leq i}}(\emptyset)^2}{4} = (A + a \cdot x_i) + \frac{B + b \cdot x_i}{2} + \frac{(B + b \cdot x_i)^2}{4}.$$

Recall that in the tampering process, with probability $1-p$, X_i is uniformly random, and with probability p, $X_i = \mathsf{MTam}(x_{\leq i-1})$ equals 1 with probability $(1+\hat{f}_{x_{\leq i-1}}(\{i\}))/2 = (1+a)/2$. Namely, $\mathsf{E}[\Phi(X_{\leq i}) \mid X_{\leq i-1} = x_{\leq i-1}]$ is equal to

$$\frac{(1+pa)}{2}\left((A+a) + \frac{B+b}{2} + \frac{(B+b)^2}{4}\right) + \frac{(1-pa)}{2}\left((A-a) + \frac{B-b}{2} + \frac{(B-b)^2}{4}\right).$$

A calculation shows that

$$\mathsf{E}[\Phi(X_{\leq i}) \mid X_{\leq i-1} = x_{\leq i-1}] = \Phi(x_{\leq i-1}) + p \cdot \left(a^2 + \frac{(1+B)ab}{2} + \frac{b^2}{4}\right) + (1-p)\frac{b^2}{4}.$$

Note that since g is bounded, we have $|B| \leq 1$ and thus

$$a^2 + \frac{(1+B)ab}{2} + \frac{b^2}{4} \geq |a|^2 - |a| \cdot |b| + \frac{|b|^2}{4} = (|a| - |b|/2)^2 \geq 0.$$

Therefore, $\mathsf{E}[\Phi(X_{\leq i}) \mid X_{\leq i-1} = x_{\leq i-1}] \geq \Phi(x_{\leq i-1})$. Applying the inequality iteratively shows that

$$\mathsf{E}[\Phi(X)] = \mathsf{E}[\Phi(X_{\leq n})] \geq \mathsf{E}[\Phi(X_{\leq n-1})] \geq \cdots \geq \mathsf{E}[\Phi(X_{\leq 0})] = \Phi.$$

□

Lemma 3.11 now follows easily.

Proof of Lemma 3.11. By applying Lemma 3.11 with $g = f^2$ and noting that $\hat{g}(\emptyset) = \nu$, we have

$$\mathsf{E}[f(X)] + \frac{\mathsf{E}[f(X)^2]}{2} + \frac{\mathsf{E}[f(X)^2]^2}{4} \geq \mu + \frac{\nu}{2} + \frac{\nu^2}{4}.$$

□

3.3.1 Tampering with One Bit

In this subsection we study the effects of applying the greedy biasing attack of Theorem 1.5 in a scenario in which the tampering circuit Tam gets to tamper with only *one* bit of the input bits where the location of the tampered bit is selected at random. Intuitively, this is similar to a p-tampering attack with $p = 1/n$ where n is the number of input bits.

Theorem 3.12. *There exists an oracle machine* Tam *running in time* $\mathrm{poly}(n/\varepsilon)$ *such that for every* $n \in N$*, every bounded-value function* $f : \{0,1\}^n \to [-1,1]$*, and every* $p < 1$*,*

$$\mathsf{E}[f(X)] \geq \mathsf{E}[f(U_n)] + \frac{\mathrm{Var}[f(U_n)]}{2n} - \varepsilon.$$

where X *is the result of* Tam *tampering with a single randomly selected bit of the input bits* U_n*.*

Proof. We use the greedy tampering algorithm of Theorem 1.5. Similar to the proof of Theorem 1.5 we present a computationally unbounded tampering that achieves bias $\frac{p \cdot \mathrm{Var}[f(U_n)]}{n}$. This tampering attack could then be turned into an efficient one by using approximations of expectations while losing an additive ε term.

Recall that by Lemma 3.3 we have $f(x) = \mathsf{E}[f(U_n)] + \sum_{i=1}^{n} \hat{f}_{x_{\leq i-1}}(\{i\}) \cdot x_i.$, and that $\hat{f}_{x_{\leq i-1}}(\{i\})$ is the the increase in the expectation when we set x_i to be one.

Let $j \in [n]$ be the randomly selected index of the tampered bit. Recall therefore we have

$$\mathsf{E}[f(X)] = \mathsf{E}[f(U_n)] + |\hat{f}_{x_{\leq j-1}}(\{j\})|$$

where $x_{\leq j-1}$ is the first $i-1$ bits of X. However, since the j'th bit is the *only* tampered bit, $x_{\leq j-1}$ is uniformly distributed. We will show that $\mathsf{E}_{j \leftarrow [n], X}[|\hat{f}_{X_{\leq j-1}}(\{j\})|] \geq \mathrm{Var}[f(U_n)]/(2n)$ which is equivalent to $\mathsf{E}\left[\sum_{j \in [n]} |\hat{f}_{X_{\leq j-1}}(\{j\})|\right] \geq \mathrm{Var}[f(U_n)]/2$. The theorem then follows by linearity of expectations. Similarly to the proof of Theorem 1.5 it holds that $\mathsf{E}\left[\sum_{j \in [n]} |\hat{f}_{X_{\leq j-1}}(\{j\})|\right] \geq \mathsf{E}[|\mu - f(U_n)|]$. However, note that if $0 < \alpha < 2$ it holds that $\alpha \geq \alpha^2/2$, and therefore it holds that $\mathsf{E}[|\mu - f(X)|] \geq \mathsf{E}[|\mu - f(X)|^2]/2 = \mathrm{Var}[f]$.

□

4 Computational Splitting from Efficient Biasing

In this section we state and prove Lemma 1.8 formally.

The following definition formalizes the scenario in which one can distinguish between the functions f_0, f_1 by a tampering circuit T that performs a p-tampering attack against the inputs of f_0 and f_1 and at the end one applies a Boolean function f over the generated output.

Definition 4.1 (Distinguishing Functions)**.** *For any two functions* f_0, f_1 *mapping* $\{0,1\}^m$ *to* $\{0,1\}^*$*, any Boolean function* f*, tampering parameter* p*, and circuit* T*, we call* (f,T) *δ-distinguishing for* (f_0, f_1) *(under p-tampering attack) if* $\Pr[f(f_1(U_m^{T,p})) = 1] \geq \Pr[f(f_0(U_m^{T,p})) = 1] + \delta$*. In case* $(\mathbf{f}, \mathbf{T})$ *is a* distribution*, we call them δ-distinguishing if the same holds but also over the randomness of* (f,T)*; namely, if it holds that* $\mathsf{E}\delta_{\mathbf{f},\mathbf{T}} \geq \delta$ *where* $(\mathbf{f}, \mathbf{T})$ *is* $\delta_{f,T}$*-distinguishing.*

The definition above could be generalized beyond p-tampering attacks and can, e.g., , be directly adapted for tampering attackers that tamper with a randomly selected bit only.

Remark 4.2. *Note that in Definition 4.1 above, the tampering circuit* T *could be fully described independently of* f_0, f_1*, or it could be described as an oracle algorithm* T^{f_0,f_1} *relative to* (f_0, f_1)*. However, when the circuit* T *receives the random bits* U_n*, it does not know which one of* f_0, f_1 *will be executed over the sequence of tampered bits. One can think of an alternative version of Definition 4.1 in which* two *tampering circuits* T_0, T_1 *together with a detecting function* f *aim to distinguish* f_0 *from* f_0 *in which* T_0 *tampers with with the inputs of* f_0 *and* T_1 *tampers with the inputs of* f_1*.*

Now we describe Lemma 1.8 formally.

Lemma 4.3 (Lemma 1.8, formalized)**.** *Let f_0 and f_1 be two efficient functions from $\{0,1\}^m$ to $\{0,1\}^{\mathrm{poly}(\kappa)}$ where $m \le \mathrm{poly}(\kappa)$ and κ is the security parameter*[4] *and $\mathrm{H}_\infty(f_b(U_m)) > t$ for both $b \in \{0,1\}$ and some $t \ge 10$. Let $\varepsilon = \Pr[f_0(U_n) \neq f_1(U_n)]$, and let $p < 1$ be an arbitrary tampering parameter. There is a PPT $A(\kappa, m)$ that* only *depends on κ, m and outputs a function $\mathbf{f}$ and a tampering oracle-aided circuit $\mathbf{T}$ such that $(\mathbf{f}, \mathbf{T}^{(f_0,f_1)})$ δ-distinguish (f_0, f_1) where $\delta \ge \Omega(\varepsilon \cdot p - 2^{-t/2})$. Moreover, in case the tampering model allows T to tamper with only one randomly selected bit, then $\delta \ge \Omega(\varepsilon/n - 2^{-t/2})$, and the distribution of the sequence of input bits remains uniformly random even after the tampering (with one bit).*

Remark 4.4. *Note that the adversary of Lemma 4.3 generates the tampering oracle-aided circuit T without the knowledge of the f_0, f_1, however, the circuit T will have access to* both *of these functions while tampering with the inputs. But the tampered randomness will be fed to f_b for an unknown $b \in \{0,1\}$. This setting is useful in scenarios in which the functions f_0, f_1 are not known to the attacker during the generation of the tampering virus, but will be known during the tampering attack. An example is the tampering attack of Section 5 for* private-key *encryption schemes in which the functions f_0, f_1 are the functions whose decryption depends on the private key and take randomness as input and encrypt, in order, $b = 0$ and $b = 1$.*

In the rest of this section, we will prove Lemma 4.3.

Proof of Lemma 4.3. We first prove the lemma for the case of p-tampering attacks and then will describe how to adapt it to the case of tampering with one randomly selected bit.

Notation. In all cases below, when f is clear from the context, for $b \in \{0,1\}$ let $\widetilde{f}_b(\cdot) = f(f_b(\cdot))$ and $\mu_b = \mathsf{E}[\widetilde{f}_b(U_n)]$. When f_0, f_1 are clear from the context let $g_f(x) = \widetilde{f}_1(x) - \widetilde{f}_0(x)$ and let $\mu = \mathsf{E}[g_f(U_m)]$. When clear from the context, we use μ'_b to denote the expected value of $\widetilde{f}_b(\cdot)$ and μ' to denote the expected value of $g_f(\cdot)$ both under a tampering attack. Let $\perp$ be a trivial tampering circuit that does not change the input. In the following, without loss of generality we will assume that $p < 1/10$. We will also use the notation $(1-f)(\cdot) = 1 - f(\cdot)$. Let ℓ be an upper-bound on the length of $f_0(\{0,1\}^m)$ and $f_1(\{0,1\}^m)$.

Description of A. A will sample $f \leftarrow \mathcal{F}$ for a family of pairwise independent functions $\mathcal{F}$ mapping $\{0,1\}^\ell$ to $\{0,1\}$, and then outputs tampering circuit T of Theorem 1.6 such that the p-tampering (oracle) circuit T^{g_f} biases g_f towards $+1$ (by $\Omega(\mathrm{Var}[g_f(U_m)] \cdot p)$). Recall that T is an oracle algorithm that has access to f_0 and f_1 as subroutines and thus it could compute g_f on any input of its choice. For simplicity, in the following we will simply write T instead of T^{g_f} or $T^{(f_0,f_1)}$.

In the rest of the proof we will prove the properties of the algorithm A as needed in Lemma 4.3. First observe that a p-tampering algorithm T that generates input distribution $U_n^{T,p}$ makes f δ-distinguish (f_0, f_1) if it biases $g_f(x) = f(f_1(x)) - f(f_0(x))$ which, in general, is *not* a Boolean function and takes values in $\{-1, 0, +1\}$.

Claim 4.5. *(f, T) is δ-distinguishing for (f_0, f_1) iff $\mathsf{E}[g_f(U_n^{T,p})] \ge \delta$ where $g_f(x) = \widetilde{f}_1(x) - \widetilde{f}_0(x)$.*

[4] The input length m could potentially be much smaller than the security parameter κ.

Proof. The proof follows by a simple application of the linearity of expectation:

$$\mu_1 - \mu_0 = \Pr[\widetilde{f}_1(U_m^{T,p}) = 1] - \Pr[\widetilde{f}_0(U_m^{T,p}) = 1] = \mathsf{E}[\widetilde{f}_1(U_m^{T,p})] - \mathsf{E}[\widetilde{f}_0(U_m^{T,p})] = \mathsf{E}[g_f(U_m^{T,p})] = \mu.$$

□

Therefore, the goal of the tampering circuit T is to bias the bounded function $g\colon \{0,1\}^m \mapsto \{-1,0,+1\}$ towards 1. However, this is not possible in general because f_0 and f_1 could be the same (or almost the same) functions, in which case $g_f(\cdot)$ is almost always zero. Here, we will use the fact that $\Pr[f_0(U_n) \neq f_1(U_n)] = \varepsilon$.

Definition 4.6. *For any function $g\colon \{0,1\}^m \mapsto \{-1,0,+1\}$, let $val(g) = \Pr[g_f(U_m) \in \{+1,-1\}]$.*

Claim 4.7. *Let $\mathcal{F}$ be a family of pairwise independent functions mapping $\mathrm{Supp}(f_0(\{0,1\}^m)) \cup \mathrm{Supp}(f_1(\{0,1\}^m))$ to $\{0,1\}$. Then $\mathsf{E}_{f\leftarrow\mathcal{F}}[val(g_f)] \geq \varepsilon/2$.*

Proof. We have:

$$\mathsf{E}_{f\leftarrow\mathcal{F}}[val(g_f)] = \mathsf{E}_{f\leftarrow\mathcal{F}}[\mathsf{P}[\widetilde{f}_0(U_m) \neq \widetilde{f}_1(U_m)]] =$$
$$\mathsf{E}[\mathsf{P}_{f\leftarrow\mathcal{F}}[\widetilde{f}_0(U_m) \neq \widetilde{f}_1(U_m)]].$$

But note that whenever $f_0(r) = f_1(r)$ it holds that $\mathsf{P}_{f\leftarrow\mathcal{F}}[\widetilde{f}_0(r) \neq \widetilde{f}_1(r)] = 0$, and whenever $f_0(r) \neq f_1(r)$ it holds that $\mathsf{P}_{f\leftarrow\mathcal{F}}[\widetilde{f}_0(r) \neq \widetilde{f}_1(r)] = 1/2$ (due to the pairwise independence of $f \leftarrow \mathcal{F}$). Therefore, it holds that

$$\mathsf{E}[\mathsf{P}_{f\leftarrow\mathcal{F}}[\widetilde{f}_0(U_m) \neq \widetilde{f}_1(U_m)] = \mathsf{P}_{r\leftarrow U_m}[f_0(r) \neq f_1(r)] \cdot (1/2) = \varepsilon/2.$$

□

In the following we will always assume that f is being sampled from the family of pairwise independent functions $\mathcal{F}$ described above.

Claim 4.8. *If $val(g) \geq \delta$. and $\mathsf{E}[g(U_m)] < 1/10$ then $\mathrm{Var}[g(U_m)] \geq 4\delta/5$.*

Proof. Recall that $\mathrm{Var}[X] = \mathsf{E}[(X - \mathsf{E}[X])^2]$. By the first property, with probability at least δ over the choice of r it holds that $g(r) \in \{-1,1\}$. For such r, by the second property, the value of $|g(r) - \mathsf{E}[g]|$ is at least $9/10$. Therefore, it holds that $\mathrm{Var}_r[g(\cdot)] \geq \delta \cdot (9/10)^2 = 4\delta/5$. □

We will use the following well-known fact followed by the left-over hash lemma.

Lemma 4.9. *Suppose X is a random variable defined over $\{0,1\}^\ell$ and $\mathrm{H}_\infty(X) \geq t$. Let $\mathcal{F} = \{f_s \mid s \in \{0,1\}^{2\ell}\}$ be a family of pairwise independent functions indexed by $s \in \{0,1\}^{2\ell}$ that map $\{0,1\}^\ell$ to $\{0,1\}$. Then the statistical distance between $(s, f_s(X))_{s\leftarrow U_{2\ell}}$ and $(U_{2\ell+1})$ is at most $O(1/2^t)$.*

Note that the goal of A is to generate the tampering circuit T without the knowledge of f_0, f_1 while, at the same time, this circuit T (which does access f_0, f_1 in a black-box way) will increase μ' (i.e., the value of μ under a p-tampering attack). To do this we rely on the high entropy of the two sources and use the fact that for such sources (when we sample $f \leftarrow \mathcal{F}$ from a family of pairwise independent functions) f is always *far* from being a good distinguishing function for (f_0, f_1) (without tampering). The formal argument follows.

By Lemma 4.9 and an averaging argument, with probability $1 - O(2^{-t/2})$ over the choice of $f \leftarrow \mathcal{F}$, it holds that $|\mu| \leq 2^{-t/2}$. Now suppose the sampled f is such that it leads to $|\mu| \leq 2^{-t/2} < 1$. We call such f a good sampled function. Since we are assuming $t \geq 10$ it follows that $2^{-t/2} < 1/10$, and so by Claim 4.8 we have $\mathrm{Var}[g_f] \geq 4 \cdot val(g_f)/5$, and so (f, T) is already ρ_f-distinguishing (f_0, f_1) for

$$\rho = (4 \cdot val(g_f)/5) \cdot (p/2) - 2^{-t/2} = 2 \cdot val(g) \cdot p/5 - 2^{-t/2}.$$

Claim 4.7 showed that $\mathsf{E}[val(\mathcal{F})] \geq \varepsilon/5$, and since there is $O(2^{-t/2})$ probability that the sampled f is not good, it follows that the sampled (f, T) are $\delta_{f,T}$-distinguishing (f_0, f_1) where $\mathsf{E}_{(f,T)\leftarrow A}[\delta_{f,T}] \geq 2p \cdot \varepsilon/25 - O(2^{-t/2})$.

Tampering with One Randomly Selected Bit. To prove Lemma 4.3 for the case of tampering with one randomly selected bit, all we have to do is to use the tampering algorithm of Theorem 3.12 rather than that of Theorem 1.6. The $\Omega(\varepsilon/n - 2^{-t/2})$ distinguishability advantage follows similarly to the proof for the case of p-tampering, but it still remains to show that the distribution of the tampered bits (despite being tampered with) remains uniform over $\{0,1\}^n$. The reason, roughly speaking, is that the tampering direction is also selected at random and that randomness depends on the selection of the pairwise independent function f.

More formally, without loss of generality we assume that for every $f \in \mathcal{F}$ complement function $(1 - f)$ is also in $\mathcal{F}$, and that these two functions are sampled with equal probabilities.[5] We also defer the selection of the function f to *after* selecting the index $j \in [n]$ of the tampered with as well fixing as the randomness used by the tampering circuit T. The greedy tampering of Theorem 3.12 proceeds by approximating whether setting x_i will increase or decrease the final expectation. It is easy to see that the decision by the tampering T would flip if we choose $(1 - f)$ instead of f. Therefore, the final distribution X after the tampering is as follows: a random index j is selected, then x_j is tampered with a random value (depending on whether f or $(1 - f)$ is chosen). This way, the distribution of X remains uniformly random over $\{0,1\}^n$. □

5 Impossibility of Tamper Resilient Cryptographic Primitives

In this section we show how to use Lemma 4.3 and the arguments given in [DOPS04] to derive impossibility results regarding tamper resilient encryption, commitments, secure two party computation, and zero-knowledge proofs.

5.1 Encryption

In this subsection we prove that no CPA secure (private-key or public-key) encryption scheme would remain secure in the presence of a tampering attack. For these applications we follow the way the "splitting lemma" (see Lemma 1.7) is employed in [DOPS04] but instead we use our computationally efficient version of this tool (i.e., our Lemma 4.3). We formalize and prove the result for the case of public-key encryption for which the tampering occurs *only* during the encryption. We then sketch the argument for the private-key setting. At the end we show that an almost identical proof holds for private-key encryption schemes as well. We start by formally defining the notion of tamper resilient encryption.

[5] This could be achieved, e.g., , by switching to choosing $(1 - f)$ with provability 0.5 whenever f is sampled from $\mathcal{F}$. This modification gives us the desired property while preserving the pairwise independence of $\mathcal{F}$.

Definition 5.1. *We call a public-key or private-key encryption scheme* p-tamper-resilient secure*, if for* every $\mathrm{poly}(n)$*-sized adversary* Adv *there exists a negligible function* $\mathrm{negl}(n)$ *such that for every sequence* $\{(x_0^n \neq x_1^n)\}_n$ *of pair of messages of equal (polynomial) length (i.e.,* $|x_0^n| = |x_1^n| = \mathrm{poly}(n)$*)* Adv *can win in the game below with probability at most* $1/2 + \mathrm{negl}(n)$ *where* n *is the security parameter.*

1. *A pair of keys* $(\mathsf{sk}, \mathsf{pk})$ *are generated and* Adv *receives the public-key* pk*. In case the scheme is private-key, the adversary receives no key.*
2. Adv *generates a* p*-tampering circuit* T *and* $\{x_0^n, x_1^n\}$*. If the scheme is public-key, these choices can depend on the public key* pk*.*
3. T *acts on* r_E*, the uniform randomness of the encryption, as a* p*-tampering circuit (see Definition 3.1) and transforms it into* r_E^T*. In case the scheme is private-key* T *also gets access to the private key.*[6]
4. *The message* $x_b \in \{x_0^n, x_1^n\}$ *is chosen at random and* $c = \mathrm{Enc}_{\mathsf{pk}}(x_b, r_E^T)$ *is sent to* Adv*.*
5. Adv *receives* c*, outputs a guess* b'*, and wins if* $b = b'$*.*

$\mathsf{P}[\text{Adv } \textit{wins}] - 1/2$ *is also called the* advantage *of* Adv*.*

Let us make a few remarks regarding Definition 5.1 follow.

- We do not allow the adversary to tamper with the randomness of the key-generation phase. One reason is that key-generation is run only once and is easier to protect than the encryption phase which might be executed many times. Note that this restriction only makes our negative result stronger.
- We require the adversary to generate the tampering circuit *without* the knowledge of the exact selected message x_b (even though this message finally will be present in the infected encrypting device). The reason is that the randomness r_E may be generated ahead of time and the message gets selected afterwards. Again, this restriction only makes our impossibility result stronger.

Why Accessing the Secret Key? Note that in Definition 5.1 we did not allow T to access the private key if the scheme is public-key, while we did allow T to access the private key if the scheme is private-key. This difference is justified since in a private-key encryption scheme the private-key should be stored and used in the encrypting device, and thus could be accessed by the tampering circuit, while in a public-key encryption scheme this is not necessarily the case. Moreover, there are private-key encryption schemes that remain tamper-resilient (against tampering with encryption's randomness) if the tampering circuit does not access the private key. E.g., consider the following scheme in which the message space is $\mathcal{M} = \{0,1\}^m$, the key length is k, and R is a PRG with a k-bit key, domain size n and image size m. To encrypt any message x, the encrypting algorithm chooses a random seed $r \leftarrow \{0,1\}^n$ and takes $\mathrm{Enc}(\mathsf{key}, x, r) = (r, R_{\mathsf{key}}(r) \oplus x)$ where $\oplus$ is the bit-wise "exclusive or" operation. It is easy to see that **(1)** this scheme is multi-message secure, and **(2)** as long as $p < 1 - \omega(\log n)/n$ the p-tampering source $U_n^{T,p}$ (as a result of p-tampering attack by T) still has $\omega(\log n)$ min-entropy and that suffices for multi-message security of the scheme under a p-tampering attack (to encryption).

[6] If the scheme was public-key this would not be necessary as the whole description of T could depend on pk.

One-Message vs Multi-Message Security. Note that if a private-key encryption scheme is 1-message secure (i.e., the encryption of all messages in the message space are indistinguishable) then one should not anticipate a similar attack to that of Theorem 5.2 to hold, simply because the one-time pad encryption has a *deterministic* encryption. However, as we will show below, if the private-key encryption scheme is even *two*-message secure (i.e., the adversary proposes two *pairs* of messages and one pair gets encrypted message by message and is returned to him) then it should have enough randomness in its encryption phase to let a tampering attack (to the randomness of encryption) break its security.

The following is the formalized variant of Theorem 1.1.

Theorem 5.2 (No Tamper-Resilient Public-Key or Private-Key Encryption)**.** *Let* Π *be a CPA-secure public-key or private-key encryption scheme for message space* $\{0,1\}$ *and completeness* $1-\mathrm{negl}(n)$*. For every* $p \geq 1/\operatorname{poly}(n)$*, there is an efficient* p*-tampering adversary that breaks* Π *(according to Definition 5.1) with advantage* $\Omega(p)-\mathrm{negl}(n)$*.*

Proof. We will use Lemma 4.3. In the following let ek be the *encryption* key. This would be the public-key in case of public-key encryption and the secret (symmetric) key in the case of private-key encryption. According to Definition 5.1 the tampering circuit T generated by the adversary does have access to the encryption key ek, either because the scheme is public-key and pk is known by the adversary before generating pk, or that the scheme is private-key and the tampering circuit T gets to read sk during the tampering attack.

Suppose the encryption randomness r_E has $m = \operatorname{poly}(n)$ bits. Recall that the adversary $\textsc{Adv}$ generates a tampering circuit T that tampers with the encryption randomness r_E of the challenger and changes its distribution from U_m into a p-tampering source $U_m^{T,p}$, receives the cipher text c that is encrypted under the tampered randomness, and it has to guess the index b of the encrypted message with probability $1/2+\Omega(p)$.

The adversary will use $x_0^n = 0, x_1^n = 1$ and wishes to distinguish between the two functions: $f_0(r) = \mathrm{Enc}_{\mathsf{ek}}(0,r)$ and $f_1(r) = \mathrm{Enc}_{\mathsf{ek}}(0,r)$. By the completeness of the encryption scheme, with overwhelming probability over the choice of ek, it holds that $\Pr[f_0(U_m) \neq f_1(U_m)] = \varepsilon \geq 1 - \mathrm{negl}(n) > 1/2$.

We first prove that the encryptions of $0,1$ have enough min-entropy, and then we will apply Lemma 4.3 to derive the attack.

Lemma 5.3 (Ciphertext Entropy)**.** *For any CPA secure private-key or public-key encryption scheme for messages* $m = \{0,1\}$*, it holds that with probability* $1-\mathrm{negl}(n)$ *over the choice of the encryption key* ek *it holds that* $\mathrm{H}_\infty(\mathbf{c}_b) \geq \omega(\log n)$ *for both* $b \in \{0,1\}$ *where* $\mathbf{c}_b$ *is the random variable denoting the encryption of* b *over the randomness of the encryption.*

Proof. Suppose on the contrary that with probability $1/\operatorname{poly}(n)$ over the choice of the key key, there is a message $b \in \{0,1\}$ and a ciphertext c' such that $\mathsf{P}[\mathbf{c}_b = c'] = 1/\operatorname{poly}(n)$. Then one can break the CPA security of Π with advantage $\geq 1/\operatorname{poly}(n)$ as follows. Given the challenge ciphertext $c \leftarrow \mathbf{c}_b$, the adversary obtains c_0 as a fresh encryption of 0, either by encrypting itself if the scheme is public-key, or by asking it to be encrypted if the scheme is CPA-secure. Then the adversary outputs outputs 0 iff $c_0 = c$.

The attack succeeds with advantage $\geq 1/\operatorname{poly}(n)$ because of the following. If the challenger chooses $b = 1$, then by the $1-\mathrm{negl}(n)$ completeness of Π, the probability that $c_1 = c$ is only $\mathrm{negl}(n)$, and therefore the probability of adversary outputting 0 is $\mathrm{negl}(n)$. On the other hand,

if the challenger chooses $b = 1$ (based on the assumption above) with probability $1/\operatorname{poly}(n)$ the two encryptions of zero equal to c'. This means that the bit output by the adversary will indeed distinguish between the two encrypted bits. □

Because of Lemma 5.3, the tampering adversary can use the attack of Lemma 4.3 for $t = \omega(n)$. Thus, the adversary is able to efficiently generate an efficient function f and an efficient tampering circuit T such that by using T in its tampering attack and outputting $f(c)$ it will win in the security game of Definition 5.1 with probability $1/2 + \Omega(p) - \operatorname{negl}(n)$. □

5.2 Tamper-Resilient Commitments and Secure Computation

In this section we use the argument used in [DOPS04] for the case of commitments holds (in the computational setting) also for a weaker variant of commitments which we call "semi-honest" commitments. Then we use the impossibility of tamper-resilient semi-honest commitments to also rule out tamper-resilient secure two-party computation.

Suppose f is a two-input finite function, Alice holds an input x, Bob holds an input y and they want to jointly compute $f(x, y)$ in a way that *only Bob* receives $f(x, y)$. In this section we show that any such finite function is either *trivial* and has a deterministic secure protocol, or that any efficient protocol to compute $f(x, y)$ is vulnerable to tampering attacks. We restrict ourselves to the case that only one party receives the output of f, and extending our result to the setting that both parties get outputs remains as an interesting open question.

We use the following semi-honest (weak) definition for the security of SFE protocols. This makes our negative result only stronger.

Definition 5.4 ((Weak) Semi-Honest SFE)**.** *Suppose* Π *is a two-party protocol in which Alice and Bob receive* 1^n *as common input, Alice gets input* $x \in \mathcal{X}$ *and Bob gets input* $y \in \mathcal{Y}$*. We call* Π *a secure function to compute a two-input function* f *with input sets* $\mathcal{X}$ *and* $\mathcal{Y}$ *if the following hold:*

- *At the end of the interaction Bob receives* $f(x, y)$ *with probability* $1 - \operatorname{negl}(n)$*.*
- *For any pair of inputs for Bob* $y_1 \neq y_2 \in \mathcal{Y}$ *and input* x *for Alice, if Bob uses* y_b *for a random* $b \in \{0, 1\}$ *to interact with Alice who uses* x*, after an honest execution, Alice cannot guess* b *with probability more than* $1/2 + \operatorname{negl}(n)$*.*
- *For any pair of inputs* $x_1 \neq x_2$ *for Alice and* y *for Bob, if* $f(x_1, y) = f(x_2, y)$*, at the end of the interaction, an honest execution of Bob using* y *cannot guess with probability more than* $1/2 + \operatorname{negl}(n)$ *the randomly chosen input of Alice* x_b*.*

Definition 5.5 (Tamper-Resilient (Weak) Semi-Honest SFE)**.** *We call* Π *a secure protocol for* f *against* p*-tampering adversaries, if the conditions stated in Definition 5.4 hold even if the party (e.g., Alice) who tries to guess the randomly chosen input of the other party (e.g., Bob's input) is allowed to perform a* p*-tampering attack along the execution of the protocol. Namely, if Alice generates a tampering circuit* Tam *which transforms the uniform randomness of Bob into* $U^{\mathsf{Tam}}_{\operatorname{poly}(n)}$ *and then interacts with Bob, she still should not be able to guess Bob's input which is chosen as* $y \leftarrow \{y_1, y_2\}$ *with probability more than* $1/2 + \operatorname{negl}(n)$*. The same holds if Bob tries to guess Alice's input* $x \leftarrow \{x_1, x_2\}$ *when he uses an input* y *such that* $f(x_1, y) = f(x_2, y)$ *even if he gets to perform a* p*-tampering attack over Alice's randomness.*

Following [BMM99] we say that f has an *insecure minor* if there are $x_1 \neq x_2 \in \mathcal{X}, y_1 \neq y_2 \in \mathcal{Y}$ such that $f(x_1, y_1) \neq f(x_2, y_1)$ but $f(x_1, y_2) = f(x_2, y_2)$. It was shown in [BMM99] that if f does not have an insecure minor, then there is a deterministic single message protocol to compute f which is secure even against malicious parties. Here we show that if f has an insecure minor, then it cannot have a tamper-resilient secure protocol.

Theorem 5.6 (Impossibility of Tamper-Resilient SFE)**.** *Suppose f is a two-input function with an insecure minor. Then for every $p > 1/\operatorname{poly}(n)$, there is no protocol to compute f securely against p-tampering adversaries (according to Definition 5.5).*

Proof Outline. To prove Theorem 5.6, roughly speaking, we show that computing f can be interpreted as a weak form of commitment scheme that cannot be tamper-resilient. Here we show that the impossibility of tamper-resilient commitments applies even to a weak form of commitment schemes that is implied by any secure (asymmetric) SFE. Note that here we are not referring to a general black-box reduction from (our variant of weakly secure) commitments to secure SFE, since such transformations might not preserve tamper-resilience, but in fact our reduction is more direct in the sense that any attack to our weakly-secure commitment scheme can be transformed back to an attack against the SFE scheme.

Definition 5.7 (Semi-Honest Commitments)**.** *We call (S, R) a semi-honest bit-commitment if:*

- **Completeness.** *Similar to standard commitments.*
- **Semi-Honest Hiding.** *We call (S, R) $(1 - \varepsilon)$ semi-honest hiding if an honest execution of R cannot guess a randomly chosen committed bit b by the end of the commitment phase with a probability more than ε. More formally, such receiver is modeled by $R = (R_1, R_2)$ where R_1 runs the receiver's protocol honestly and then R_2 receives the view of R_1 and outputs a the output of R. We simply call (S, R) semi-honest hiding if it is $(1 - \operatorname{negl}(n))$ semi-honest hiding.*
- **Semi-Honest Binding.** *We call (S, R) $(1 - \varepsilon)$ semi-honest binding, if the probability that (in an honest execution), the randomness r_S is accepted as decommitment to* both *of $\{0, 1\}$ is at most ε. Namely, with probability at least $1 - \varepsilon$ over the choice of (r_S, r_R) it holds that $\tau(0, r_S, r_R) \neq \tau(1, r_S, r_R)$ where $\tau(b, r_S, r_R)$ is the transcript of the commitment phase. We call (S, R) simply semi-honest binding if it is $(1 - \operatorname{negl}(n))$ semi-honest binding.*

The following construction shows how to get a commitment scheme from any protocol that computes a function f with an insecure minor. We will show that if the protocol to compute f if semi-honest secure, then so is the commitment scheme.

Construction 5.8. *Suppose $\Pi = (A, B)$ is a two party protocol for computing a function f with an insecure minor of: $f(x_1, y_1) \neq f(x_2, y_1)$, $f(x_1, y_2) = f(x_2, y_2)$. We construct a bit commitment scheme $\Sigma_\Pi = (S, R)$ as follows.*

- *For an input $b \in \{0, 1\}$ given to the sender, the commitment phase consists of an execution of Π to compute $f(x_b, y_2)$ where S emulates $A(x_b)$ and R emulates $B(y_2)$.*
- *In the decommitment phase, S reveals b and sends the randomness she used to emulate $A(x_b)$ to R as the decommitment string.*

Theorem 5.9. *If Π is a weak semi-honest secure computation for f (with insecure minor), then the commitment scheme Σ_Π of Construction 5.8 is semi-honest secure.*

Proof. The completeness of Σ_Π is immediate. The hiding of Σ_Π also follows immediately from the security of Π for Alice (and the fact that the receiver is honestly executing the protocol as well).

In the following we prove the semi-honest binding of Σ_Π. Suppose for sake of contradiction that with probability $\varepsilon > 1/\operatorname{poly}(n)$ over the choice of r_S, r_R, it holds that using both inputs $b \in \{0,1\}$ leads to the *same* transcript: $\tau(0, r_S, r_R) = \tau(1, r_S, r_R)$. This event which we denote by E can be efficiently verified to hold by the sender (who is emulating the execution of Alice).

Now suppose, in a different game, the receiver emulates the execution of $B(y_2)$ instead of $B(y_1)$. In this case, by the security of the protocol Π for Bob, the event E should happen with probability at least $\varepsilon' = \varepsilon - \operatorname{negl}(n) > 1/\operatorname{poly}(n)$, or otherwise Alice can distinguish Bob's inputs by using either of the inputs $\{x_1, x_2\}$, executing the protocol with Bob, and checking wether the event E holds or not. This contradicts the completeness of the protocol Π, because with non-negligible probability ε', both inputs x_1, x_2 lead to the same exact transcripts and therefore the same outputs for Bob, but with $1 - \operatorname{negl}(n)$ probability the outputs should be different. □

Before proving Theorem 5.6 we show that the commitment messages have enough min-entropy.

Lemma 5.10 (Commitment's Entropy)**.** *For any semi-honest secure bit commitment scheme Π, it holds that with probability $1 - \operatorname{negl}(n)$ over the choice of the receiver's randomness r_R it holds that $\mathrm{H}_\infty(\mathbf{c}_b) \geq \omega(\log n)$ for both $b \in \{0,1\}$ where $\mathbf{c}_b$ is the random variable (over the randomness of the sender) denoting the transcript of commitment for message b.*

Proof. The proof is similar to that of Lemma 5.3.

Suppose on the contrary that with probability $1/\operatorname{poly}(n)$ over the choice of the receiver's randomness r_R, there is a message $b \in \{0,1\}$ and a transcript c' such that $\mathsf{P}[\mathbf{c}_b = c'] \geq 1/\operatorname{poly}(n)$. Then one can break the semi-honest security of Π with advantage $\geq 1/\operatorname{poly}(n)$ as follows. Given the transcript $c \leftarrow \mathbf{c}_b$, the semi-honest receiver adversary $R = (R_1, R_2)$ obtains c_0 by running the honest sender's protocol (with fresh randomness) against the *same* randomness r_R of the receiver R_1. The receiver R_2 outputs outputs 0 iff $c_0 = c$.

The attack succeeds with advantage $\geq 1/\operatorname{poly}(n)$ because of the following. If the sender chooses $b = 1$, then by the security of Π against semi-honest senders the probability that $c_1 = c$ is only $\operatorname{negl}(n)$, and therefore the probability of adversary outputting 0 is $\operatorname{negl}(n)$. On the other hand, if the sender chooses $b = 1$, with probability $\geq 1/\operatorname{poly}(n)$ the two transcripts of for $b = 0$ will be equal to c'. This means that the bit output by the adversary will indeed distinguish between the two committed bits. □

Finally we will prove Theorem 5.6 using Lemma 5.10 and Lemma 4.3.

Proof of Theorem 5.6. By the reduction of Construction 5.8 and Theorem 5.9 it is sufficient to rule out "tamper-resilient" semi-honest commitments where tamper-resilient semi-honest commitments are naturally defined by requiring semi-honest binding and hiding properties to hold even if a party performs an $(1/\operatorname{poly}(n))$-tampering attack to randomness of the other party.

For a fixed r_R let $f_b(r_S)$ be the transcript of the scheme where sender uses (b, r_S) and the receiver uses r_R. By Lemma 5.10, the entropy of the transcript will be $t \geq \omega(n)$. So a tampering receiver attacker A can use the attack of Lemma 4.3 to generate (f, T) and have T tamper with

the randomness of the sender in a way that by applying $f(\cdot)$ to the transcript A can distinguish between $b = 0$ and $b = 1$ with advantage $\Omega(p) - \text{negl}(n)$. □

Remark 5.11. *Our tampering attacks against encryption, commitment, and secure computation schemes described in subsections above all use a single invocation of the computational splitting algorithm of Lemma 1.8. Therefore, if the tampering model allows the tampering attacker to tamper with a single uniformly selected bit of randomness, the tampering attack has the extra property that the tampered random bits remain completely random! Therefore, no statistical test on the tampered random bits could reveal the fact that they are tampered with.*

5.3 Impossibility of Tamper-Resilient Zero-Knowledge for NP

In this section we show that no interactive proof system (with an efficient prover) for languages in $\mathsf{NP} \setminus \mathsf{BPP}$ can be zero-knowledge against a tampering verifier. For simplicity, in the following we assume that the prover's randomness and the common input are both of length n (otherwise we pad the shorter one). First we give an explicit definition for tamper-resilient zero-knowledge.

Definition 5.12 (Tamper-Resilient Zero-Knowledge)**.** *Suppose (P, V) is an interactive proof system for language L where x (of length n) is the common input and $y \in \{0,1\}^{\text{poly}(n)}$ is prover's private input. A p-tampering verifier V^* at the beginning of the interaction generates a tampering circuit T that gets y as auxiliary input and transforms the uniform source of private randomness U_n of the prover into a p-tampering source $U_n^{T,p}$. We call such proof system α-tamper-resilient zero-knowledge against p-tampering, if for every p-tampering PPT verifier V^*, there exists a simulator* Sim *such for every sequence of triples (x, y, z) of: the common input x, prover's private input y, the following two ensembles are $\alpha(|x|)$-indistinguishable:*

$$\{\textit{Output of } V^* \textit{ in } \langle V^*, P(y)\rangle(x)\}_{x\in L, y} \textit{ and } \{\text{Sim}(x)\}_{x\in L}.$$

Remark 5.13. *Note that in the definition above we did not give auxiliary input to the verifier, even though such definitions are standard. However, an auxiliary input zero knowledge is indeed a* stronger *form of zero knowledge, and the fact that our main result about zero knowledge if* negative *it only makes our result stronger. We also chose to simulate the* output *of the malicious verifier rather than its view, but these two are well-known to be equivalent.*

Theorem 5.14 (Impossibility of Zero-Knowledge for NP)**.** *Suppose there exists an efficient prover zero-knowledge proof system Π for $L \in \mathsf{NP}$ with negligible completeness and soundless errors. Then Π cannot be $o(p)$-tamper-resilient zero-knowledge against p-tampering verifiers for $p > 1/\text{poly}(n)$ unless $L \in \mathsf{BPP}$.*

Comparison to the setting of [DOPS04]. As noted in the introduction, we cannot simply follow the footsteps of [DOPS04] and use our computational splitting lemma (Lemma 4.3) to derive Theorem 5.14 above because of how the definition of zero-knowledge proofs conceptually differs between the computationally efficient and information theoretic (imperfect) sources of randomness. The difference is that in [DOPS04] the simulation is *universal* and should handle all the (bad) sources of randomness (with high min-entropy of certain forms), while here the simulator implicitly *knows* the bad source of randomness because it can access the code of the malicious verifier.

In the rest of this section we will prove Theorem 5.14. At a high level, the proof is as follows.

Intuition. Fix input x and a verifier randomness r_V. Then either of the following holds:

1. If the prover is (almost) deterministic, we show how to use the simulator of the zero-knowledge protocol to decide whether $x \in L$ or not with non-negligible advantage. The proof of this step improves upon a result by Goldreich and Oren [GO94] who showed that the min-entropy of τ is positive.

2. If the prover is using sufficient randomness in the transcript to convey the proof in zero-knowledge manner: We show that in this case the messages of the system (now generated by the prover, since r_V is already fixed) have min-entropy at least $\omega(\log n)$. Relying on the high min-entropy of the messages and using ideas behind the proof of Lemma 4.3 we still show the existence of a tampering verifier who can learn nontrivial information about any of the bits of the witness by applying a Boolean function f to the transcript τ generated by a tampered r_S. Since the view of such tampering verifier should be simulated by the efficient simulator $\text{SIM}(\cdot)$, by running $\text{SIM}(x)$ enough number of times we can learn y_i for every i and obtain y.

We start by formally describing our extension of the result of Goldreich and Oren [GO94], which corresponds to the first step above. Note that here we only rely on the zero-knowledge property of the proof system (regardless of the tampering).

Theorem 5.15 (Message-Entropy of Zero-Knowledge Proofs)**.** *Suppose (P, V) is an interactive proof system for a language L with negligible completeness and soundness errors. Then there is a PPT algorithm A such that the following properties hold.*

- *A takes as input x and 1^K and runs in time $\text{poly}(|x|, K)$.*
- *If A accepts x, it implies $x \in L$ up to $\text{negl}(n)$ error: $\mathsf{P}[A(x) = 1 \text{ and } x \notin L] \leq \text{negl}(n)$.*
- *Suppose in addition that (P, V) is zero-knowledge, then either of the following holds for $x \in L$:*
  1. *$A(1^K, x)$ accepts x with probability at least $1/\text{poly}(K, n)$, or*
  2. *with probability $1 - 1/K$ over the choice of r_V, it holds that the min-entropy of the transcript $\tau = \langle V, P\rangle(x)$ conditioned on r_V is at least: $\text{H}_\infty(\tau \mid r_V) \geq \log(K)$.*

Interpretation. Theorem 5.15 implies that if the entropy of the messages coming from the prover in a zero-knowledge protocol is $O(\log n)$, then by taking $K = \text{poly}(n)$ large enough $A(1^k, x)$ can decide $x \in L$ with $1/\text{poly}(n)$ advantage (i.e., it accepts $x \in L$ with $1/\text{poly}(n)$ probability and accepts any $x \notin L$ with $\text{negl}(n)$ probability). This improves over the result of Goldreich and Oren [GO94] that obtained the same conclusion based on assumption that the prover is *deterministic*.

The following theorem corresponds to the second step of the proof of Theorem 5.14.

Theorem 5.16 (Signaling the Witness Bits)**.** *Suppose (P, V) is an efficient-prover $o(p)$-tamper-resilient zero-knowledge proof system against p-tampering adversaries for a language L and $p = 1/\text{poly}(n)$. There exists an efficient algorithm B_p such that the following holds for all $x \in L$: If with probability $1 - 1/K$ over the choice of r_V, the min-entropy of the transcript $\tau = \langle V, P\rangle(x)$ conditioned on r_V is $\text{H}_\infty(\tau \mid r_V) \geq \log K$ and that $K = \omega(1/p^2)$, then with probability $1 - \text{negl}(n)$ the output of B_p is a valid witness y for $x \in L$.*

Proof of Theorem 5.14. We first prove Theorem 5.14 using Theorems 5.15 and 5.16. We present an efficient algorithm C that decides membership of $x \in L$ for the language $L \in \mathsf{NP}$ in probabilistic polynomial time by relying on efficient algorithms A and B of Theorems 5.15 and 5.16.

Algorithm $C_p(x)$.

1. Take $K = \omega(1/p^2)$ to be some $\mathrm{poly}(n)$ (which is possible since $p = 1/\mathrm{poly}(n)$).
2. Run algorithm $B_p(x)$ to get some NP-witness y. Output $x \in L$ if y is an acceptable witness.
3. Otherwise, run the algorithm $A(1^K, x)$ and output whatever A decides about $x \in L$.

Soundness of C. We first show that if $x \notin L$, C accepts x with $\mathrm{negl}(n)$ probability. The reason is that since we verify the extracted "witness" y. Therefore if $x \notin L$, no such witness can exist and pass the verification and so we would not accept x in Step 2. On the other hand, Theorem 5.15 asserts that the probability that if $x \notin L$, the algorithm A accept x with negligible probability.

Completeness of C. There are two possibilities.

1. First suppose that with probability $1 - 1/K$ over the choice of r_V, the min-entropy of the transcript $\tau = \langle V, P\rangle(x)$ conditioned on r_V is at least: $\mathrm{H}_\infty(\tau \mid r_V) \geq \log K$. In this case, the algorithm B will extract the witness with probability $1 - \mathrm{negl}(n)$ and so C accepts x with probability $1 - \mathrm{negl}(n)$.
2. Otherwise, by the properties of the algorithm A specified in Theorem 5.15, x will be accepted with probability at least $1/\mathrm{poly}(n, K) \geq 1/\mathrm{poly}(n)$.

□

In the following we will prove Theorems 5.15 and 5.16.

5.3.1 Proof of Theorem 5.15

Our proof, at a high level, follows the approach of Goldreich and Oren [GO94], who showed that zero-knowledge with *deterministic* provers is impossible unless $L \in \mathsf{BPP}$. Namely, one starts by assuming that the prover is deterministic for *all* $x \in L$ and derive $L \in \mathsf{BPP}$. In our Theorem 5.15, however, we conclude an instance-dependent statement; namely, for every $x \in L$, *either* there is "sufficient" entropy in the messages, or that we can decide the membership of x in L efficiently.

Below, first, we give a comparison between the approach of [GO94] and our approach for the weaker claim (than Theorem 5.15) that the entropy of the prover messages in case of $x \in L$ cannot *always* be $O(\log n)$. Then will then prove Theorem 5.15 formally.

The Approach of [GO94]. Suppose the prover is deterministic. This means that for every prefix of the transcript of the interaction between the prover and the verifier $p_1, v_1, \ldots, p_{i-1}, v_{i-1}$, the next message p_i of the prover is determined. This fact can be used, together with the existence of the simulator $\textsc{Sim}$, to efficiently generate an accepting transcript τ whenever $x \in L$, in a way that the same procedure does *not* generate an accepting transcript whenever $x \notin L$. The main ideas of [GO94] to obtain both properties simultaneously are as follows:

1. At a high level, in the process of generating τ we are executing the verifier V against some fixed (simulated) prover strategy P^*. This way, the soundness condition guarantees that this will not lead to an accept for $x \notin L$ unless with negligible probability.

2. The (simulated) prover P^* needs to behave close to the honest prover if $x \in L$ to generate an accepting τ. For this, we use the simulator SIM to get the prover messages p_i *one by one*. To get the i'th message p_i, we use the simulator over a verifier who knows the previously generated partial transcript $p_1, v_1, \ldots, p_{i-1}$ as auxiliary input, and sends v_{i-1} as the next answer according to the algorithm of the honest verifier. This way the value p'_i will be indeed the same as the actual p_i (that the honest prover would return) with $1-\text{negl}(n)$ probability, or otherwise these two answers would be distinguishable which contradicts the zero-knowledge property (note that p_i is fixed can be known to the distinguishing circuit).

Extensions. When the prover is randomized, we cannot conclude that the generated message p'_i is necessarily the same as p_i with high probability because the prover's messages (even conditioned on the previous messages $p_1, p_2, \ldots, p_{i-1}$) could be different in every new execution of the protocol against the same fixed honest verifier. Thus, instead we follow the following approach.

1. **How to Generate Messages:** Instead of executing the simulator once for getting every message p'_{i+1}, we repeatedly execute the simulator SIM up to some $\text{poly}(n)$ times till we get the *same* the partial transcript $(p_1, v_1 \ldots, p_{i-1}, v_i)$ as generated previously, and only then we look at the simulated value p'_i (which we hope to be the same as p_i with good probability).

2. **Analysis:** The challenging part is to show that if there is at least one transcript $\tau = (p_1, v_1, \ldots, p_m, v_m)$ that appears with $\alpha \geq 1/\text{poly}(n)$ probability, then we will generate τ through the process above with non-negligible probability. A naive analysis would use the fact that in every step, the provability of obtaining p_i is at least $\beta_i > \alpha > 1/\text{poly}(n)$. This simple analysis does not work because $(1/\text{poly}(n))^{\text{poly}(n)}$ could be negligible (but note that this analysis in fact works for constant number of rounds because $(1/\text{poly}(n))^{O(1)} \geq 1/\text{poly}(n)$). For arbitrary polynomial number of rounds we need a sharper analysis to show that if we obtain p_i with probability β_i, the product $\prod \beta_i$ is also non-negligible. We do so by decomposing α into $\alpha = \alpha_1 \cdot \alpha_2 \ldots$ where α_i is the probability of p_i being the honest message conditioned on the previously generated transcript. We show that for every i, $\beta_i/\alpha_i \approx 1 + \varepsilon$ for arbitrary small $\varepsilon = 1/\text{poly}(n)$ and thus $\beta = \prod_i \beta_i \approx \prod_i \alpha_i = \alpha$.

The Formal Proof. Let $\tau = Q(x, r_V)$ be the random variable denoting the transcript of the protocol when the common input is x, and the verifier has random coins fixed to r_V. Since the verifier acts deterministically once x and r_V are fixed, the distribution of Q is a deterministic function of the random coins of the prover. For an implicitly fixed r_V, we further let $Q_i(x, \tau_{i-1})$ denote the distribution of the message sent by the prover in the i^{th} round, given that $\tau_{i-1} = p_1, v_1, \ldots, p_{i-1}, v_{i-1}$ is the set of messages exchanged in the previous rounds.

In the following we assume that the proof system (P, V) has m rounds.

Observe that the message sent by the verifier in the i^{th} round is a (poly-time computable) deterministic function f of x, r_V, and the messages sent by the prover in the previous rounds. We can use this fact to write Q in terms of Q_i, as follows. Let $\tau = p_1, v_1, \ldots, p_m, v_m$ be a transcript of the protocol consistent with x and r_V. Then, assuming that τ_i is the partial transcript up to and including round i, it holds that:

$$\mathsf{P}[Q(x, r_V) = \tau] = \mathsf{P}[Q_1(x, \tau_0) = p_1] \cdot \cdots \cdot \mathsf{P}[Q_m(x, \tau_{m-1}) = p_m].$$

Our algorithm A will be such that for $K = \text{poly}(n)$ and $x \in L$, if for a $1/K$ fraction of r_V, the min-entropy of $Q(x, r_V)$ is $\leq \log K$, then $A(1^K, x)$ will output "accept" with probability $\geq 1/16K^2$.

Suppose for a $1/K$ fraction of r_V, there is some corresponding "heavy" message transcript $\tau^* = \tau^*(x, r_V)$ that occurs with probability $\geq 1/K$; we call such r_V *special*.

Our approach will be to produce an efficient prover strategy P^* that, when interacting with V on x with such a special r_V, will produce the corresponding τ^* as its transcript with probability $\geq 1/8K$ (without knowing r_V and τ^* in advance).

First, assuming that we have such a prover strategy P^*, we claim that we can use P^* to create the required A. We observe that for fixed $x \in S$ (of large enough length $n = |x|$), not more than $1/2$ fraction of the special r_V can have their corresponding τ^* to be a rejecting transcript, because otherwise the honest verifier will reject on common input x with probability $1/2K > \text{negl}(n)$, violating the $1 - \text{negl}(n)$ completeness. Thus a uniformly chosen r_V will be both special and have an accepting τ^* with probability $> 1/2K$.

Now we examine what will happen when we simulate (P^*, V) on common input $x \in S$, with a uniformly chosen r_V. With probability $> 1/2K$, the random coins of V will have a corresponding τ^* that is an accepting transcript, and conditioned on getting such an r_V, the output of (P^*, V) will be τ^* with probability $\geq 1/8K$. Thus, with probability $\geq 1/16K^2$, (P^*, V) will be accepting. Further, notice that when the common input is $x \notin L$, then the output of (P^*, V) can be accepting with probability at most $\text{negl}(n)$, or otherwise the $1 - \text{negl}(n)$ soundness would be violated. Therefore, we can simply define A to be the machine simulating (P^*, V) on common input x.

It remains to show how to construct P^*. Define V^* to be the verifier strategy that takes a partial message transcript as its auxiliary input z, sends its first messages according to z, and then aborts. By the auxiliary input zero-knowledge, there exists an efficient simulator S for V^*.

The malicious prover P^* (defined based on S), over the common input x, works as follows.

1. Let $\hat{\tau} = \emptyset$.
2. For $i = 1$ to k :
   (a) Run $S(x, \hat{\tau})$ repeatedly, until you find a transcript τ whose first $i - 1$ messages match with $\hat{\tau}$. If no such message was found in $K(\log K)$ runs, abort.
   (b) Send p_i according to τ, and receive v_i from the verifier.
   (c) $\hat{p_i} = p_i$, $\hat{v_i} = v_i$

Assuming that the randomness of the verifier, r_V, is special, ideally we want that in each iteration i, P^* sends the i^{th} prover message such that it matches τ^* for x and r_V of the verifier it is interacting with.

To actually analyze the behavior of P^*, let assume that P^* and V have each sent all messages correctly corresponding to τ^* for all iterations up to the t^{th} iteration (i.e., $\hat{\tau} = \tau^*_{t-1}$). We will calculate the probability that P^* then sends p^*_t on the t^{th} iteration. Then, since V acts deterministically given x, r_V and the prover messages, V will also send v^*_t on this iteration.

Our analysis will initially assume that we have access to an oracle simulator $O(x, z)$, that produces transcripts identically distributed to a real interaction between (P, V^*). We will estimate the success probability for each iteration when we are using this oracle. Then, we will argue that replacing the oracle O with our simulator S gives at most a negligible loss.

Using the Ideal Oracle. We now give the analysis when we use O. First we analyze the probability that our oracle simulator finds a transcript matching the messages sent so far. Since the verifier V^* will send messages according to its auxiliary input, the verifier messages up to step t will always match v_i^* (after this, V^* will abort, but it turns out we don't care about the $(t+1)^{\text{th}}$ and later messages in this transcript). So we only need to find the probability that the transcript has matching prover messages. Since O is a perfect simulator oracle, the first $t-1$ prover messages of $O(x, \tau_{t-1}^*)$ will be distributed exactly as the random variables Q_i for $i \leq t-1$. Thus the probability that a transcript produced by $O(x, \tau_{t-1}^*)$ matches the first $t-1$ messages of τ^* is given by

$$\prod_{i=1}^{t-1} \mathsf{P}[Q_i(x, \tau_{i-1}^*) = p_i^*] \geq \prod_{i=1}^{k} \mathsf{P}[Q_i(x, \tau_{i-1}^*) = p_i^*] \geq \frac{1}{K}.$$

Thus, repeating $K(\log K)$ times will give us a probability $\geq 1-(1-1/K)^{K(\log K)}$ of producing a matching transcript, which is $\geq 1-1/K$. Given that we have produced a matching transcript, the probability that the t^{th} message of this transcript is p_t^* is given by $\geq \mathsf{P}[Q_t(x, \tau_{t-i}^*) = p_t^*]$. Therefore the overall probability that P^* using O sends p_t^* in the t^{th} iteration, given that it sent the correct messages in the previous iterations, is $(1-1/K) \cdot \mathsf{P}[Q_t(x, \tau_{t-i}^*) = p_t^*]$.

Using the Simulator. Now we replace the ideal oracle O by using the simulator S. We argue that P^* using S must succeed with probability at least $(1-1/K) \cdot (1-1/K) \cdot \mathsf{P}[Q_t(x, \tau_{t-i}^*) = p_t^*]$. Otherwise, we can create a distinguisher $\mathcal{D}$ to distinguish between $\mathsf{View}_{V^*}\langle P, V^*(\tau_{t-1}^*)\rangle(x)$ and $S(x, \tau_{t-1}^*)$ as follows: Simulate the t^{th} iteration of P^* and outputs 1 whenever the message sent is m_i^* (where 1 corresponds to guessing that the distribution is $\mathsf{View}_{V^*}\langle P, V^*(\tau_{t-1}^*)\rangle(x)$). Then $\mathcal{D}$ distinguishes between $\mathsf{View}_{V^*}\langle P, V^*(\tau_{t-1}^*)\rangle(x)$ and $S(x, \tau_{t-1}^*)$ with advantage at least

$$\frac{1}{K} \cdot (1 - \frac{1}{K}) \cdot \mathsf{P}[Q_t(x, \tau_{t-i}^*) = p_t^*] \geq \frac{1}{K} \cdot (1 - \frac{1}{K}) \cdot \frac{1}{K}$$

contradicting the fact that the protocol is zero-knowledge.

Finally note that the probability that every message sent by P^* using S matches τ^* is at least:

$$\prod_{t=1}^{k} (1 - \frac{1}{K})^2 \cdot \mathsf{P}[Q_t(x, \tau_{t-i}^*) = p_t^*] = (1 - \frac{1}{K})^{2K} \cdot \mathsf{P}[Q(x, r_V) = \tau^*] > \frac{1}{8K}.$$

The last inequality holds for large enough K because $(1-1/K)^{2K} \approx 1/e^2$ for large enough K.

Proof of Theorem 5.16. In the following we use $\{-1, +1\}$ (instead of $\{0, 1\}$) to represent the bits of the witness y. In what follows we will describe a malicious verifier V_i^* which is indexed by $i \leq \text{poly}(n)$. We will show how to use a simulator for the $o(p)$-tamper-resilient zero knowledge to obtain an efficient way to guess y_i with overwhelming probability where y_i is the i'th bit of the witness given to the prover. Then, by enumerating i we can construct all of y with high probability.

The description of tampering verifier V_i^*. The verifier V_i^* starts by choosing r_V at random as the randomness of the verifier. Then it runs the attacker of Lemma 4.3 to obtain $(\mathbf{f}, \mathbf{T})$ for the following two functions f_0, f_1. The function f_b takes prover's randomness as input and outputs the transcript τ of the interaction of the verifier (with randomness r_V) with the prover (with the designated randomness) using witness y^b where y is the same as y except possibly on the i'th bit in

which y^b is fixed to be b. In other words, it is as if the prover will flip the i'th bit of y if it is not b and then interacts with the verifier using a randomness r_P (which is the input) and then the transcript denotes the output of f_b. After sampling $(f, T) \leftarrow (\mathbf{f}, \mathbf{T})$ from the corresponding distribution of Lemma 4.3 for functions f_0, f_1 described above V_i^* sends T as the p-tampering circuit to tamper with prover's randomness, and outputs $f(\tau)$ as its final output bit.

Output of V_i^* correlates with y_i. Now by the properties of Lemma 4.3, if with probability $1 - 1/K$ over the choice of r_V, the min-entropy of the transcript $\tau = \langle V, P\rangle(x)$ conditioned on r_V is $\mathrm{H}_\infty(\tau \mid r_V) \geq \log K$ and that $K = \omega(1/p^2)$, then we can conclude that with probability $1 - o(p)$ over the choice of r_V, the output bit of V_i^* distinguishes f_0 from f_1 by advantage $\Omega(p) - \sqrt{1/K} = \Omega(p) - o(p) = \Omega(p)$. Therefore, the $o(p)$ error over the choice of r_P cannot affect this bound and the final output bit of V_i^* will indeed distinguish f_0 from f_1 by advantage $\Omega(p)$.

Using simulator of V_i^* to reveal y_i. Since the output of V_i^* is just one bit, the simulator should indeed simulate this bit *statistically* well. This means that the simulator $\textsc{Sim}$ for V_i^* gets x as input and generates an output bit that distinguishes which bit b the i'th bit of the witness y_i is taking, and reveals this value by advantage $\Omega(p)$. Therefore, the output bit of simulator (executed over x) could be used to guess y_i with probability $1/2 + \Omega(p)$. By repeating the execution of the simulator and taking majority one can obtain y_i with probability $1 - \mathrm{negl}(n)$. □

6 Achieving Tamper Resilience Using Pseudorandomness

We present our positive results on tamper-resilient cryptography in this section. We show that assuming the existence of pseudorandom generators (PRGs), which is implied by the existence of one-way functions [HILL99], a wide range of cryptographic primitives, including signatures, identification schemes, witness-hiding protocols, as well as encryption schemes with a "weak" notion of semantic security (see Definition 6.9) can be made resilient to p-tampering for $p = n^{-\alpha}$, where n is the security parameter and $\alpha > 0$ is an arbitrary constant. Our construction can be extended in a straightforward way to achieve resilience to p-tampering with $p = \log^{-c}(n)$ for some constant c, assuming the existence of PRGs with sub-exponential security.

Using Pseudorandomness with Short Seeds. We obtain our positive results by a generic transformation that converts a secure implementation P of a primitive $\mathcal{P}$ to one that is secure even in the presence of p-tampering attacks. Our transformation is in fact very simple: given an implementation P with standard security, we convert it to $\overline{P}$ that generates a short random seed x with length $s \leq 1/p$, and then uses a PRG $G : \{0,1\}^s \to \{0,1\}^{\mathrm{poly}(n)}$ to generate a pseudorandom string $G(x)$. Finally, it emulates P with $G(x)$ as the randomness. Note that by doing so, P only needs to use s random bits to generate a seed x for G, and so, on average, only 1 bit of the randomness gets tampered with during the p-tampering attack.

First note that when we use the PRG G over a seed s of length $1/p$ and use $G(s)$ instead of the truly random bits, the scheme $\overline{P}$ (typically) remains secure if P was originally secure due to the pseudorandomness of $G(s)$.[7] Also, since the min-entropy of the tampered s is at least $\lg[((1+p)/2)^{1/p}]$, any event E involving a system that executes P, would happen with a tampered

[7] More formally, for this, we would need the event E that the security is broken to be efficiently recognizable.

s with probability at most $\varepsilon \cdot ((1+p)/2)^{1/p}/2^{1/p} < \varepsilon \cdot e$ where ε is the probability of E happening in the original (un-tampered) game. Therefore, if originally we had $\varepsilon = \text{negl}(n)$, the probability of adversary winning remains negligible $e\varepsilon = \text{negl}(n)$.

Our transformation applies to natural primitives with a security game that can be captured by a "threshold-t falsifiable game" with $t = 0$. A threshold-t falsifiable game Π is simply a game between an efficient challenger C and an adversary A such that C outputs accept or reject after the interaction with A, and the game Π is secure if for every efficient adversary A, C outputs accept with probability at most $t(n)+\text{negl}(n)$. Threshold-0 falsifiable games, in general, capture primitives with security defined as hardness of searching secrets. As mentioned, this includes signature schemes, witness-hiding protocols, and identification schemes.

Our result about tamper-resilient identification schemes might seem skeptical at first because most known identification schemes are based on zero-knowledge protocols, and we have demonstrated earlier that zero-knowledge is impossible in the presence of tampering. However, tamper-resilient identification schemes are still possible to obtain because identification schemes only rely on the weaker property of *witness hiding* (as opposed to zero-knowledge property), and in this section we show that the witness hiding property can be preserved under tampering.

Beyond Threshold-0 Primitives. The above mentioned idea of using pseudorandomness can only be applied to threshold-0 primitives since it relies on the fact that adversary's original winning probability is at most negligible. We also show how to obtain tamper-resilient implementations of cryptographic primitives even when the security game is not threshold-0, but here we assume the number of honest parties to be at least 2. This result, on a high level, is obtained by one honest party Alice helping the other honest party Bob extract pure randomness from its tampered randomness $\widehat{X_B}$. To do this, Alice sends random string $\widehat{X_A}$ (which might also be a tampered string) to Bob, and Bob applies a two-source extractor to get pure randomness $R_B = \mathsf{Ext}(\widehat{X_A}, \widehat{X_B})$. In case $\widehat{X_A}$ is sent over a public channel observed by the adversary, we would need a *strong* two-source extractor [DO03] such that R_B remains uniformly random even conditioned on the value of $\widehat{X_A}$. This idea can be applied to obtain private truly random seeds where we have two conditions: **(1)** $p < \sqrt{2} - 1$; this is required to guarantee that each of $\widehat{X_A}, \widehat{X_B}$ have sufficient min-entropy required by the known two-source extractors, and that **(2)** the honest parties know the set of honest parties. For the more general case of any constant $p < 1$, and where the (at least two) honest parties do not know each other, we rely on the network extractor paradigm of [KLR09] to obtain private *pseudorandom* seeds under (non-standard) computational assumptions.

6.1 Tamper Resilient Signatures

As a concrete example, we will first show how to achieve tampering resilient signatures. The arguments for the other primitives will be indeed similar.

Definition 6.1 (Many-Time Signatures)**.** *A many-time signature scheme* $P = (\text{Gen}, \text{Sign}, \text{Ver})$ *is* secure *if every PPT adversary* A *wins in the following game* $\Pi = (A, C)$ *with negligible probability:*

1. *C executes* $\text{Gen}(1^n)$ *to generate signing key* sk *and verification key* vk*, and sends* vk *to A.*
2. *For* $i \in [\text{poly}(n)]$*, A sends a message* m_i *to C and receives* $\sigma_i = \text{Sign}_{\mathsf{sk}}(m_i)$ *from C.*
3. *A generates and sends* (m, σ) *to C.*

4. C accepts (i.e., A wins) if $m \neq m_i$ for every i, and $\mathrm{Ver}_{\mathsf{vk}}(m, \sigma) = 1$.

We say that any A breaks the security of P if he wins with non-negligible probability.

Recall that, assuming the existence of OWFs, we know how to create many-time secure signature schemes that are deterministic in their signing and verification phases. In fact, any many-time signature scheme that uses randomness in the signing phase can be converted into one that is deterministic, by using PRFs. This could be done by generating the seed for the PRF in the key-generation phase, and adding it as part of the secret key. Then, whenever a message m is to be signed, we apply the PRF to m, and use the result as the randomness needed to sign.

The definition of p-tamper-resilient signatures follows from Definition 6.1 and asserting that the security holds against p-tampering adversaries.

Definition 6.2 (Tamper Resilient Signature)**.** *A many-time signature scheme is p-tamper-resilient if it remains secure according to Definition 6.1, even if the adversary, at the beginning of the game, generates a p-tampering circuit Tam that transforms the uniform randomness $U_{\mathrm{poly}(n)}$ of the challenger (i.e., the randomness for key-generation and signing) into a $U^{\mathsf{Tam},p}_{\mathrm{poly}(n)}$ and the challenger uses $U^{\mathsf{Tam},p}_{\mathrm{poly}(n)}$ in its interaction.*

Theorem 6.3. *Let $\alpha \in (0,1)$ be a constant. If there exists a many-time secure signature scheme, then there exists a many-time $(n^{-\alpha})$-tamper-resilient signature scheme.*

Proof. Let $P = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Ver})$ be a secure signature scheme. W.l.o.g., we assume that P is deterministic except in Gen. Suppose further that $\mathrm{Gen}(1^n)$ uses $m(n)$ bits of randomness. Let $G : \{0,1\}^s \to \{0,1\}^{m(n)}$ be a pseudorandom generator with $s = O(n^\alpha)$. (Recall that secure signature schemes imply OWFs [IL89], which imply PRGs with arbitrary polynomial length output.) We define $\overline{P} = (\overline{\mathrm{Gen}}, \mathrm{Sign}, \mathrm{Ver})$, where $\overline{\mathrm{Gen}}(1^n)$ works as follows: $\overline{\mathrm{Gen}}$ generates a random seed X of length $s = n^\alpha$, and then computes $r = G(X)$. It then emulates Gen using r as random coins. In the following by $\overline{\Pi}$ we refer to the new scheme and by $\overline{C}$ we refer to the new challenger of the multi-message security game of $\overline{\Pi}$.

Roughly speaking, Theorem 6.3 is proved by observing that **(1)** P remains secure even if the randomness of P is generated from the pseudorandom source $G(X)$, and **(2)** after tampering, the seed X used by $\overline{P}$ still has $s - O(1)$ bits of min-entropy.

The Formal Proof. Suppose that there exists an efficient adversary $\overline{A}$ that breaks the scheme $\overline{\Pi}$ by winning the game against $\overline{C}$ with non-negligible probability. Namely, there exists a non-negligible ε such that for infinitely many n,

$$\mathsf{P}[\langle \overline{A}, \overline{C} \rangle(1^n) = 1] \geq \varepsilon(n).$$

By an averaging argument and fixing coins of A, we can assume w.l.o.g. that $\overline{A}$ is deterministic and thus the tampering circuit Tam is fixed. Observe that in this case, the interaction $\langle \overline{A}, \overline{C} \rangle$ in $\overline{\Pi}$ is equivalent to the interaction $\langle A, C[G(U_s^{\mathsf{Tam}})] \rangle$ in Π, where the notation $\langle A, C[D] \rangle$ means the output of simulating $\langle A, C \rangle$ with C's coins drawn from distribution D, and A is simply $\overline{A}$ without sending the tampering circuit. Thus, we have

$$\mathsf{P}[\langle A, C[G(U_s^{\mathsf{Tam}})] \rangle = 1] \geq \varepsilon.$$

Note that since U_s^{Tam} is a p-tampering source over $s = 1/p$ bits, it has min-entropy $s - \log e$. This can be seen by observing that for every $r \in \{0,1\}^s$, $\mathsf{P}[U_s^{\mathsf{Tam}} = r] \leq ((1+p)/2)^s \leq e^{p \cdot s}/2^s = e/2^s$.

This means that if we use the randomness U_s rather than U_s^{Tam} the probability of any event would decrease at most by a factor of e. Namely, A is a successful attacker breaking the scheme with probability $\varepsilon(n)/e > 1/\operatorname{poly}(n)$. But, this contradicts the security of the scheme P and completes the proof of Theorem 6.3. □

Our proofs of security for identification schemes and witness hiding protocols follow the same lines as those of our result about signature schemes. Thus we will only discuss the specific modifications we make in each case.

6.2 Identification Schemes

An identification scheme is a protocol that allows Alice to prove her identity to Charlie, and a malicious Bob cannot impersonate Alice even if he gets to see Alice proving her identity polynomially many times. Each identity can be represented by some $\alpha \in \{0,1\}^n$ and there exists a "public-file" that couples every identity α by some secretly generated public information $I(s, \alpha)$. The secret s (which is chosen at random) is given to the person with identity α. The public record $(\alpha, I(s, \alpha))$ will be accessed by both the "prover" (of the identity) and the "verifier" during identification.

Definition 6.4. *An identification scheme $\Pi = (I, P, V)$ has three efficient components: The algorithm I takes as input $(\alpha \in \{0,1\}^n, s \in \{0,1\}^{\operatorname{poly}(n)})$ and outputs $v \in \{0,1\}^{\operatorname{poly}(n)}$ and (P, V) form an interactive proof system. We also demand the following properties.*

- **Completeness:** $\mathsf{P}[\langle P(s), V\rangle(\alpha, I(s,\alpha)) = 1] = 1$ *holds for every* $\alpha \in \{0,1\}^n, s \in \{0,1\}^{\operatorname{poly}(n)}$.
- **Soundness:** *For every* $\alpha \in \{0,1\}^n$, *every* $\operatorname{poly}(n)$*-sized (non-uniform) interactive adversary A wins in the following game with* $\operatorname{negl}(n)$ *probability.*
  1. *For* $s \leftarrow \{0,1\}^{\operatorname{poly}(n)}$, *$A$ interacts with $P(s)$ polynomially many times.*
  2. *Then A (while keeping its internal state) interacts with V on common input $(\alpha, I(s,\alpha))$.*
  3. *A wins if V accepts the interaction.*

Remark 6.5 (Identification from Signatures)**.** *Many-times secure signature schemes can be used to obtain identification schemes as follows. The information generation algorithm I, for every identity α generates a pair of signing key s and verification key v. (Note that the signing key can always be assumed to be a uniformly sampled string.) The verifier V, given the public record (α, v) sends a uniformly chosen random message m to P who signs m using the signing key s, and sends the signature σ back to V. Finally V accepts if σ is a valid signature of m. It is easy to see that the many-time security of the signature scheme implies the soundness of the constructed identification scheme. Further, if the signature scheme has a deterministic* Sign *algorithm, then the identification scheme has a deterministic prover P (but the verifier is randomized).*

Although secure signature schemes imply secure identification schemes according to Remark 6.5, this transformation does not preserve the tamper resilience by definition. The reason is that, even though it is known that there are signature schemes with deterministic verification, this identification scheme uses some extra randomness which could also be tampered with by the adversary.

Theorem 6.6. *If there exists a secure identification scheme, then there exists a p-tamper-resilient identification scheme, with $p = n^{-\alpha}$, for any constant $\alpha > 0$.*

Proof. The existence of a secure identification scheme implies the existence of OWFs [IL89], which in turn implies the existence of a secure signature scheme Σ with a deterministic signing algorithm. The signature scheme Σ can be used to obtain an identification scheme Π with a deterministic prover P according to the construction described in Remark 6.5.

Since the prover P in the obtained identification scheme is deterministic, the adversary A can only usefully tamper with the randomness of the information generation algorithm I and the verifier V. I uses its randomness in the initial *generation* phase, and V uses its randomness in the final *verification* phase. Since in the security game of Π there is only a *single* verification performed, the number of random bits needed for the information generation and the final verification is an a priori fixed polynomial. Therefore, we can apply the same idea that we used to make signature schemes tamper resilient. Namely, we modify I and V *both* so that each of them uses only a short random seed of length $O(1/p)$ and applies a PRG to expand it to the right size $\text{poly}(n)$. The proof of tamper-resilience of the new scheme is identical to that of Theorem 6.3.

□

6.3 Witness Hiding Protocols

Definition 6.7 (Witness Hiding Protocols)**.** *Suppose R is an* NP *relation. Namely, there is an efficient algorithm that accepts (x, w) when $|w| = \text{poly}(|x|)$ iff $(x, w) \in R$. Suppose* Gen *is a randomized sampling procedure that given 1^n (and enough random bits) runs in time $\text{poly}(n)$ and outputs some (x, w) of length $|x| = n$. A proof system (P, V) for the relation R is* one-time witness hiding *w.r.t. the sampling procedure* Gen *if, for all PPT adversaries A, A wins in the following game with negligible probability:*

1. $\text{Gen}(1^n)$ *generates $(x, w) \in R$.*
2. *A interacts in $\langle P(w), A\rangle(x)$.*
3. *A outputs w', and wins if $(x, w') \in R$.*

(P, V) is called q-time witness hiding (resp. witness hiding) if A is allowed to participate in $q = \text{poly}(n)$ (resp. any polynomial) number of interactions for the same (x, w) before outputting w'.

All variants of witness hiding defined in Definition 6.7 can be defined under tampering attacks.

Theorem 6.8. *Any (q-times) witness hiding protocol (P, V) for some relation R can be converted into a stateless (q-times) p-tamper-resilient witness-hiding protocol, with $p = n^{-\alpha}$, for any constant $\alpha > 0$. Achieving p-tamper resilience witness-hiding (with unbounded number of repetitions) is also possible assuming that the prover is allowed to keep internal state between repetitions.*

Proof. First note that any witness hiding protocol implies the existence of one-way functions as follows: Given input r, use r as randomness and using $\text{Gen}(\cdot)$ sample $(x, w) \in R$ and output $x = f(r)$. Inverting $f(r)$ for a random r implies breaking the witness hiding of the corresponding protocol even without participating in the interactive phase. Again, we can again use the existence of OWFs to get PRGs [HILL99].

To achieve tamper resilience under a single execution of the interactive proof we can use the same exact trick as we did for signature schemes: $\overline{\text{Gen}}$ uses a "short" random seed s of length $1/p$, and applies a PRG to make it of sufficiently long to run Gen. The prover also does the same thing; i.e., it tosses $1/p$ many coins and expands them to the right length using a PRG. In the following we focus on the case of sequential repetition.

A Priori Bounded Number of Repetitions. If the number of sequential repetitions is an a priori bounded polynomial $q = \text{poly}(n)$, then we can still take the same approach as that of the single-repetition case. Namely, one can think of the security game with a fixed number $q = \text{poly}(n)$ of repetitions as a game with a fixed time-complexity that needs a fixed $\ell = \text{poly}(n)$ number of random bits used by the challenger $C = (\text{Gen}, P_1, P_2, \ldots, P_q)$. Thus, by using a PRG we can expand an initial $1/p$ number of true random bits to ℓ bits and apply the same analysis.

Unbounded Number of Repetitions. At first sight, it seems that our general technique of using PRGs does not work if we go through an unbounded number of repetitions (i.e., a polynomial that is chosen by the adversary). But, if the prover can keep internal state, we can use a PRF as a way to obtain a PRG with an unbounded number of output bits. (Note that if g is a PRG, for any polynomial ℓ, the function $G(s) = [g_s(1), \ldots, g_s(\ell)]$ is a PRG.) Thus, again, we start by using $1/p$ truly random bits s, and use a PRG g with key s and compute it over $g_s(1), g_s(2), \ldots$ to obtain enough number of pseudorandom bits that is needed for the execution of the repetitions.

To analyze the tamper resilience of the scheme above, we apply the same analysis presented for the previous cases. All we have to do additionally is to first *fix* the adversary, which fixes the number of its sequential repetitions, and then apply the same analysis as before. This was not previously needed because, e.g., in case of signatures, even though the adversary's complexity was not fixed before the analysis, we had an upper-bound on the number of random bits needed by the challenger, but here we get this upper-bound after fixing the adversary's running time. □

6.4 Weak Semantic Security

Here we present the following new definition as a relaxation of the semantic security and prove our positive result about the possibility of tamper resilient encryption under this relaxed definition.

Definition 6.9. *A public-key or private-key encryption scheme* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is called* $(1+\delta)$-weakly semantically-secure *if for every* $\text{poly}(n)$*-sized (non-uniform) adversary* $\textsc{Adv}$*, arbitrary distribution* $\mathcal{X}$ *over* $\mathcal{M}$ *and arbitrary functions* $I, f \colon \mathcal{M} \mapsto \{0,1\}^{\text{poly}(n)}$ *there exists a PPT simulator* $\textsc{Sim}$ *such that:*

$$\mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Adv}(I(m), \text{Enc}(m)) = f(m)] \leq (1+\delta) \cdot \mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Sim}(I(m)) = f(m)] + \text{negl}(n).$$

The probabilities above are also over the generation of the encryption key. Note that $(\text{Gen}, \text{Enc}, \text{Dec})$ *is* semantically-secure *if it is* 1*-weakly semantically secure.*

Definition 6.10 (Tamper Resilient Weakly Semantically Secure Encryption)**.** *We call an encryption scheme* Π tamper-resilient weakly semantically secure *if for every* $p = 1/\text{poly}(n)$ *(where* n *is the security parameter)* Π *is secure under Definition 6.9 even when the adversary* $\textsc{Adv}$ *is able to generate a* p*-tampering circuit* Tam *that modifies the randomness of the key generation and encryption algorithms into* p*-tampering sources.*

We prove the following positive result about the possibility of achieving weakly semantically secure encryptions under Definition 6.9. Our result implies that when we consider the weak semantic security according to Definition 6.9 for functions $f(m)$ which cannot be computed by more than $\mathrm{negl}(n)$ probability given the encryption of m, one can always preserve this property (of f remaining "hard" to compute given the encryptions) even in the presence of $p = \frac{1}{\mathrm{poly}(n)}$-tampering attacks.

Theorem 6.11. *Suppose there exits a semantically secure (public-key or private-key) encryption scheme, and $p = n^{-\alpha}$ for a constant $\alpha > 0$. Then for every $0 < \beta < \alpha$, there exists a $(1 + O(n^{-\beta}))$ weakly semantically-secure encryption scheme that is secure against p-tampering adversaries.*

Proof. We follow the same paradigm of previous sections but with even smaller PRG seeds. The existence of any semantically secure encryption scheme implies the existence of OWFs, which in turn implies the existence of PRGs with arbitrary polynomial stretch [HILL99]. Thus, for every constant $\gamma > 0$ we can use two seeds of length $n^\gamma/2$ to get enough number of pseudorandom bits to use in Gen and Enc. To get the $(1 + O(n^{-\beta}))$ weakly semantically secure scheme $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec})$ we set $\gamma = \alpha - \beta$ and then modify the Gen and Enc algorithms (and call them $\mathrm{Gen}', \mathrm{Enc}'$) to use a truly random seed of length $n^\gamma/2$ and stretch it to the necessary number of bits.

First we claim that if $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ was semantically secure, $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec})$ remains semantically secure. This is nontrivial because the distribution $\mathcal{X}$ is not necessarily efficiently samplable and f, I are not necessarily efficiently computable. We use the classical result of [GM84] that a public key scheme is semantically secure iff it is CPA secure. A CPA secure is defined based on a security game in which the adversary, given the public key, chooses two messages m_0, m_1, receives the encryption of m_b for $b \leftarrow \{0, 1\}$ and wins if he guesses b correctly with probability $1/2 + 1/\mathrm{poly}(n)$. It is easy to see that the CPA property is preserved when we use pseudorandom bits in Gen, because otherwise the security game itself can be turned into a distinguisher against the PRG that is used to expand the pseudorandom bits used in Gen.

Now we study the security of $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec})$ in the presence of $n^{-\alpha}$-tampering attacks. During a $n^{-\alpha}$-tampering attack, every bits of the PRG seed of length n^γ (applied to two seeds of length $n^\gamma/2$) is tampered with by the adversary only with probability $n^{-\alpha}$. Therefore, the probability of any seed being the result of the tampering is at most $(1 + n^{-\alpha}/2)^{n^\gamma} \leq e^{n^{-\beta}}/2^{n^\gamma}$. Similarly to the argument in the analysis of the tamper-resilient signatures, we conclude that any event E which an efficient adversary could make it happen without the tampering, would happen now (with the tampering) with probability at most

$$e^{(n^{-\beta})} \cdot \mathsf{P}[E \text{ without tampering}] \leq (1 + O(n^{-\beta})) \cdot \mathsf{P}[E \text{ without tampering}].$$

We conclude Theorem 6.11 by considering the event E to be the event that the adversary is computing $f(m)$ correctly. In the original semantically secure scheme it holds that

$$\begin{aligned} &\mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Adv}(I(m), \mathrm{Enc}(m)) = f(m)] \\ &\leq \mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Sim}(I(m)) = f(m)] + \mathrm{negl}(n) \end{aligned}$$

which implies that in the new scheme $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec})$ it holds that

$$\begin{aligned} &\mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Adv}(I(m), \mathrm{Enc}(m)) = f(m)] \\ &\leq (1 + O(n^{-\beta})) \cdot (\mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Sim}(I(m)) = f(m)] + \mathrm{negl}(n)) \\ &\leq \mathrm{negl}(n) + (1 + O(n^{-\beta})) \cdot \mathsf{P}_{m \leftarrow \mathcal{X}}[\textsc{Sim}(I(m)) = f(m)]. \end{aligned}$$

□

6.5 Generalization to Threshold-0 Primitives

Our tamper-resilient constructions for Signatures, Identification Schemes, Weakly-Semantically Secure Encryptions, etc, above clearly suggest a general approach that can be applied to a wide range of primitives where the adversary's job is to make the challenger accept with non-negligible probability. Here we describe the abstract properties of the primitives that makes this approach applicable.

Any natural cryptographic primitive $\mathcal{P}$ can be viewed as a set of interactive algorithms with some completeness properties imposed on them. For example an encryption scheme has three components of $(\text{Gen}, \text{Enc}, \text{Dec})$. In the security game of $\mathcal{P}$ some of these components are executed on the "challenger" side and their randomness is prone to online tampering. There might be several "instantiations" of each of these components by the challenger during its interaction with the adversary. For example in the security game of multi-message signatures, the signing algorithm might be executed an unbounded polynomial number of times depending on adversary's choice. If an executed component can keep internal state across different instantiations, we still consider this a single instantiation. For example, when a tampering verifier interacts with a prover sequentially in $\text{poly}(n)$ repetitions while the prover keeps an internal state (as it was the case in witness-hiding protocols in Section 6.3), we consider this a single instantiation.

In order to apply the pseudorandomness-based approach of previous sections to a primitive and make its implementation resilient to p-tampering for $p = 1/\text{poly}(n)$, it is sufficient to have both of the following two conditions:

1. The threshold of the security game is 0 (i.e., adversary needs to with with $1/\text{poly}(n)$).
2. The number of randomized components of the primitive $\mathcal{P}$ which are instantiated during the security game are at most $n^{-\beta}$ for a constant $\beta < \alpha$ where $p = n^{-\alpha}$.

When we have the above two conditions, all we have to do is to use pseudorandomness to execute the components of $\mathcal{P}$ as follows: each component uses a seed s of length n^γ as the key to a PRF g where $\gamma = \alpha - \beta$ and uses $g_s(1), g_s(2), \ldots$ to get its "random" bits during its execution. Note that, for any fixed polynomial t, the sequence $[g_s(1), g_s(2), \ldots, g_s(t)]$ is pseudorandom. So after fixing the complexity of the adversary Adv who interacts with the challenger, each instantiated component of $\mathcal{P}$ is using a PRG that stretches a seed s of length n^γ to the needed number of random bits. Each of these seeds could be p-tampered with by the adversary in an instantiation during the security game, and so their joint min-entropy loss (compared to a truly random seed) is at most $O(n^\beta \cdot n^\gamma \cdot n^{-\alpha}) = O(1)$. Therefore, any event E that the original (non-tampering) adversary could make it happen with probability ρ, will happen in the security game with the tampering attack with probability at most $2^{O(1)}\rho$, which remains $\text{negl}(n)$ assuming that ρ was already negligible.

6.6 Beyond Threshold-0 Primitives

Here we describe a general method that allows us to make a cryptographic primitive tamper-resilient to p-tampering attacks for any constant $p < 1$, as long as there are two honest parties involved in the primitive (that know each other). We start by describing a solution for the task of key agreement (which is a primitive in which both parties are honest and the adversary is a passive

eavesdropper), and then describe how it can be generalized to more settings by relying the work of [KLR09] at the cost of non-standard computational assumptions.

6.6.1 Tamper-Resilient Key Agreement

A tampering adversary attacking a key agreement protocol $\widehat{\Pi}$ between Alice and Bob sends tampering circuits Tam_A and Tam_B to Alice and Bob where each tampering circuit tampers with the randomness of the corresponding party. In this section we prove the following theorem.

Theorem 6.12. *Suppose* Π *is a secure key agreement protocol. Then for every constant* $p < \sqrt{2}-1$ *there is another key agreement protocol* $\widehat{\Pi}$ *which is secure against* p*-tampering adversaries.*

Proof. We show how Alice and Bob can start from $2n$ bits of tampered random bits and obtain $\Omega(n)$ bits X_A, X_B each in a way that (X_A, X_B) are (jointly) statistically close to uniform.

Alice and Bob divide their original $2n$-bit random strings into two equal parts and call them (X_A^1, X_A^2) and (X_B^1, X_B^2) where the bits in X_A^1 (resp. X_B^1) are tossed before the bits in X_A^2 (resp. X_B^2). Suppose the tampered values of these random seeds are $(\widehat{X}_A^2, \widehat{X}_A^2), (\widehat{X}_B^2, \widehat{X}_B^2)$. We use the following lemma which is implied by Theorem 1 of [DO03].

Lemma 6.13 (Strong Two-Source Extractors)**.** *Suppose* X *and* Y *are two random variables defined over* $\{0,1\}^n$ *with min-entropy* $\alpha \cdot n$ *and* $\alpha > 1/2$*, then there is an efficient function* $\mathsf{Ext}\colon \{0,1\}^{2n} \mapsto \{0,1\}^m$ *(independent of* X, Y*) for* $m = \Omega(n)$ *such that* $(\mathsf{Ext}(X,Y), Y)$ *is* $\mathrm{negl}(n)$*-close to* (U_m, Y)*.*

Description of $\widehat{\Pi}$: Alice (resp. Bob) will send $\widehat{X}_A^1$ (resp. $\widehat{X}_B^1$) to the other party. Then Alice (resp. Bob) will apply the strong extractor of Lemma 6.13 and gets $X_A = \mathsf{Ext}(\widehat{X}_A^2, \widehat{X}_B^1)$ (resp. $X_B = \mathsf{Ext}(\widehat{X}_B^2, \widehat{X}_A^1)$). Then they use (X_A, X_B) as their randomness and execute Π. Note that By taking n large enough, X_A, X_B would be long enough to execute Π.

Since the tampering circuits Tam_A and Tam_B cannot communicate, it follows that the random variables $\widehat{X}_A^2$ and $\widehat{X}_B^1$ are independent. Since, $p < \sqrt{2}-1$, the min-entropy of the "source" $\widehat{X}_A^2$ is at least α for $\alpha > 1/2$ even conditioned on the publicly revealed message $\widehat{X}_A^1$, simply because $\widehat{X}_A^2$ was tampered with *after* $\widehat{X}_A^1$. Therefore, the extracted X_A will remain statistically close to a secret U_m for $m = \Omega(n)$ in eyes of the adversary, even conditioned on the information sent over the public channel. The same argument holds for the uniformity of X_B. Thus, Alice and Bob can use the (statistically close to) pure random seeds X_A, X_B to run the protocol Π securely.

□

Remark 6.14 (Synchronization Issue)**.** *A subtle point here is that Alice and Bob should toss coin and obtain* $(\widehat{X}_A^2, \widehat{X}_A^1)$ *and* $(\widehat{X}_B^2, \widehat{X}_B^1)$ before *sending* $\widehat{X}_A^1$ *and* $\widehat{X}_B^1$ *to each other. Otherwise, if say Alice tosses coin after receiving* $\widehat{X}_B^1$*, then the tampering circuit* Tam_A *could tamper* X_A^2 *into* $\widehat{X}_A^2$ with the knowledge *of* $\widehat{X}_B^1$ *which makes* $\widehat{X}_B^1$ *and* $\widehat{X}_A^2$ *dependent and thus we cannot apply Lemma 6.13.*

References

[ACM+14] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *International Cryptology Conference*, pages 462–479. Springer, 2014. 14

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009. 1

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance – a cautionary note. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 1–11, November 1996. 1

[Ale96] One Aleph. Smashing the stack for fun and profit. *http://www.shmoo.com/phrack/Phrack49/p49-14*, 1996. 1

[BDL97] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997. 1, 6

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 501–510. IEEE, 2010. 1

[BMM99] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In *Annual International Cryptology Conference–CRYPTO*, pages 80–97. Springer, 1999. 25

[BPR14] Mihir Bellare, Kenneth G Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In *Advances in Cryptology–CRYPTO 2014*, pages 1–19. Springer, 2014. 4

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual international cryptology conference–CRYPTO*, pages 513–525. Springer, 1997. 1

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 235–244. ACM, 2000. 2

[CKM11] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 740–758. Springer, 2011. 1, 6

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2010. 1

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 511–520. IEEE, 2010. 1

[DO03] Yevgeniy Dodis and Roberto Oliveira. On extracting private randomness over a public channel. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 252–263. Springer, 2003. 34, 41

[DOPS04] Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im) possibility of cryptography with imperfect randomness. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 196–205. IEEE, 2004. 5, 8, 21, 24, 27

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 293–302. IEEE, 2008. 1

[DPW18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *Journal of the ACM (JACM)*, 65(4):20, 2018. 1, 6

[DSK12] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits against constant-rate tampering. In *Advances in Cryptology–CRYPTO 2012*, pages 533–551. Springer, 2012. 1

[FB09] Ariel J. Feldman and Josh Benaloh. On subliminal channels in encrypt-on-cast voting systems. In *Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections*, EVT/WOTE'09, pages 12–12, Berkeley, CA, USA, 2009. USENIX Association. 4

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *International Colloquium on Automata, Languages, and Programming*, pages 391–402. Springer, 2011. 1, 6

[Fry00] Niklas Frykholm. Countermeasures against buffer overflow attacks. *RSA Tech Note*, pages 1–9, 2000. 1

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, 2004. 1, 3, 6

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 39

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994. 8, 9, 28, 29

[GR12] Shafi Goldwasser and Guy Rothblum. How to compute in the presence of leakage. In *IEEE Symposium on Foundations of Computer Science*, 2012. 1

[HDWH12] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium*, August 2012. 2

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 33, 37, 39

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989. 4, 35, 37

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006. 1, 6

[KK08] Seny Kamara and Jonathan Katz. How to encrypt with a malicious random number generator. In *Fast Software Encryption*, pages 303–315. Springer, 2008. 5

[KKS11] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *Annual Cryptology Conference*, pages 373–390. Springer, 2011. 1, 6

[KLR09] Yael Tauman Kalai, Xin Li, and Anup Rao. 2-source extractors under computational assumptions and cryptography with defective randomness. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 617–626. IEEE, 2009. 34, 41

[KLR12] Yael Kalai, Allison Lewko, and Anup Rao. Formulas resilient to short-circuit errors. In *IEEE Symposium on Foundations of Computer Science*, 2012. 1

[LHA+12a] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 626–642. Springer, 2012. 2

[LHA+12b] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. http://eprint.iacr.org/. 2

[LL10] Feng-Hao Liu and Anna Lysyanskaya. Algorithmic tamper-proof security under probing attacks. In *International Conference on Security and Cryptography for Networks*, pages 106–120. Springer, 2010. 1

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Crypto*, 2012. 1, 6

[Mah18] Saeed Mahloujifar. Private communication. 2018. 14

[MM17] Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017. 14

[MR04] Micali and Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography Conference (TCC), LNCS*, volume 1, 2004. 1

[PB04] Jonathan D. Pincus and Brandon Baker. Beyond stack smashing: Recent advances in exploiting buffer overruns. *IEEE Security & Privacy*, 2(4):20–27, 2004. 1

[Rot12] Guy N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, 2012. 1

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978. 1

[Sim94] Gustavus J Simmons. Subliminal channels; past and present. *Transactions on Emerging Telecommunications Technologies*, 5(4):459–474, 1994. 4

[SV86] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. 3, 10, 11

[YY96] Young and Yung. The dark side of 'black-box' cryptography, or: Should we trust capstone? In *CRYPTO: Proceedings of Crypto*, 1996. 4