Electronic Colloquium on Computational Complexity, Report No. 172 (2017)

From Laconic Zero-Knowledge to Public-Key Cryptography

Itay Berman* Akshay Degwekar* Ron D. Rothblum†
Prashant Nalini Vasudevan*

November 3, 2017

Abstract

Since its inception, public-key encryption (PKE) has been one of the main cornerstones of cryptography. A central goal in cryptographic research is to understand the foundations of public-key encryption and in particular, base its existence on a natural and generic complexity-theoretic assumption. An intriguing candidate for such an assumption is the existence of a cryptographically hard language $\mathcal{L} \in \mathsf{NP} \cap \mathsf{SZK}$.

In this work we prove that public-key encryption can be based on the foregoing assumption, as long as the (honest) prover in the zero-knowledge protocol is *efficient* and *laconic*. That is, messages that the prover sends should be efficiently computable (given the NP witness) and short (i.e., of sufficiently sub-logarithmic length). Actually, our result is stronger and only requires the protocol to be zero-knowledge for an *honest-verifier* and sound against computationally bounded cheating provers.

Languages in NP with such laconic zero-knowledge protocols are known from a variety of computational assumptions (e.g., Quadratic Residuocity, Decisional Diffie-Hellman, Learning with Errors, etc.). Thus, our main result can also be viewed as giving a unifying framework for constructing PKE which, in particular, captures many of the assumptions that were already known to yield PKE.

We also show several extensions of our result. First, that a certain weakening of our assumption on laconic zero-knowledge is actually *equivalent* to PKE, thereby giving a complexity-theoretic characterization of PKE. Second, a mild strengthening of our assumption also yields a (2-message) oblivious transfer protocol.

*MIT. Emails: {itayberm, akshayd, prashvas}@mit.edu.
†MIT and Northeastern University. Email: ronr@mit.edu.

ISSN 1433-8092

Contents

1 Introduction

Underlying symmetric key encryption is a centuries-old idea: *shared secrets enable secure communication*. This idea takes many forms: the Caeser cipher, the unconditionally secure one-time pads, fast heuristic constructions like AES, and a multitude of candidates based on the hardness of a variety of problems. The discovery of *public-key* encryption, by Diffie and Hellman [DH76] and Rivest, Shamir and Adleman [RSA78], was revolutionary as it gave us the ability to communicate securely without any shared secrets. Needless to say, this capability is one of the cornerstones of secure communication in today's online world.

As is typically the case in cryptography, we are currently very far from establishing the security of public-key cryptography unconditionally. Rather, to establish security, we rely on certain computational intractability assumptions. Despite four decades of extensive research, we currently only know constructions of public-key encryption from a handful of assumptions, most notably assumptions related to the hardness of factoring, finding discrete logarithms and computational problems related to lattices (as well as a few more exotic assumptions).

One of the central open problems in cryptography is to place public-key encryption on firmer complexity-theoretic grounding, ideally by constructing public-key encryption from the minimal assumption that one-way functions exist. Such a result seems well beyond current techniques, and by the celebrated result of Impagliazzo and Rudich [IR89] requires a non-blackbox approach. Given that, a basic question that we would like to resolve is the following:

From what general complexity-theoretic *assumptions can we construct public-key cryptography?*

Our motivation for asking this question is twofold. First, we seek to understand: Why is it the case that so few assumptions give us public-key encryption? What kind of "structured hardness" is required? Secondly, we hope that this understanding can guide the search for new concrete problems that yield public-key encryption.

1.1 Our Results

Our main result is a construction of a public-key encryption scheme from a general complexity-theoretic assumption: namely, the existence of a cryptographically hard language $\mathcal{L} \in \mathsf{NP}$ that has a *laconic* (honest-verifier) statistical zero-knowledge argument-system. We first discuss the notions mentioned above, and then proceed to state the main result more precisely (yet still informally).

By a cryptographically hard language we mean an NP language that is average-case hard with a solved instance generator.[1] A proof-system is *laconic* [GH98, GVW02] if the number of bits sent *from the prover to the verifier* is very small.[2] An argument-system is similar to an interactive proof, except that soundness is only required to hold against *computationally bounded* (i.e., polynomial time) cheating provers. Honest verifier zero-knowledge means that the *honest* verifier learns no more in the interaction than the fact that $x \in \mathcal{L}$ (i.e., the verifier can simulate the honest interaction by itself). Thus, our main result can be stated as follows:

[1] Loosely speaking, a solved-instance generator for an average-case hard language $\mathcal{L} \in \mathsf{NP}$ is an algorithm that generates samples $(x, w) \in \mathcal{R}_{\mathcal{L}}$ (where $\mathcal{R}_{\mathcal{L}}$ is the NP relation) and where x is distributed according to the average-case hard distribution restricted to YES instances.

[2] Laconic proof-systems with constant soundness and very short communication (e.g., just a single bit) are indeed known. As a matter of fact, many of the known hard problems that are known to yield public-key encryption schemes have such laconic SZK proof-systems (see Section 1.1.1 and Appendix A).

Theorem 1 (Informally Stated, see Theorem 3.6)**.** *Assume that there exists a cryptographically hard language $\mathcal{L} \in \mathsf{NP}$ with an r-round statistical honest-verifier zero-knowledge argument-system, with constant soundness, that satisfies the following two requirements:*

- **Efficient Prover:** *The honest prover strategy can be implemented in polynomial-time, given the* NP *witness.*[3]
- **Laconic Prover:** *The prover sends at most q bits in each of the r rounds, such that $r^2 \cdot q^3 = O(\log n)$, where n is the input length.*

*Then, there exists a public-key encryption (*PKE*) scheme.*

We emphasize that requiring only *honest-verifier* zero-knowledge (rather than full-fledged zero-knowledge) and *computational soundness* (i.e., an argument-system) weakens our assumption, and therefore only strengthens our main result. We also comment that we can handle provers that are less laconic (i.e., send longer messages) by assuming that the language $\mathcal{L}$ is sub-exponentially hard. Lastly, we remark the assumption in Theorem 1 may be viewed as a generalization of the notion of *hash proof systems* [CS02].[4] We discuss this point in more detail in Section 1.2.

1.1.1 Instantiations

Many concrete assumptions (which are already known to yield public-key encryption schemes) imply the conditions of Theorem 1. First, number-theoretic assumptions such as Quadratic Residuosity (QR) and Decisional Diffie-Hellman (DDH) can be shown to imply the existence of a cryptographically hard NP language with a laconic and efficient SZK argument-system and therefore satisfy the conditions of Theorem 1 (these and the other implications mentioned below are proven in Appendix A).

We can also capture assumptions related to lattices and random linear codes by slightly relaxing the conditions of Theorem 1. Specifically, Theorem 1 holds even if we relax the completeness, soundness and zero-knowledge conditions of the argument-system to hold only for *most* (but not necessarily all) of the instances (chosen from the average-case hard distribution). We call arguments with these weaker properties *average-case* SZK arguments.

It is not hard to see that *lossy encryption* [PVW08, BHY09] yields such an *average-case* laconic and efficient zero-knowledge argument-system. Recall that a PKE scheme is *lossy* if its public-keys are indistinguishable from so-called "lossy keys" such that a ciphertext generated using such a lossy key does not contain information about the underlying plaintext. Consider the following proof-system for the language consisting of all valid public-keys: given an allegedly valid public-key, the verifier sends to the prover an encryption of a random bit b and expects to get in response the value b. It is not hard to see that this protocol is a laconic and efficient average-case SZK argument-system.

Many concrete assumptions yield cryptographically hard languages with *average-case* laconic and efficient SZK arguments (whether via lossy encryption or directly). Most notably, Learning

[3] In the context of *argument-systems* (in contrast to general interactive proofs), the assumption that the honest prover is efficient goes without saying. Nevertheless, we wish to emphasize this point here.

[4] As a matter of fact, hash proof systems can be viewed as a special case of our assumption in which the (honest) prover is *deterministic* or, equivalently, sends only a single bit. In contrast, we handle arbitrary *randomized* provers (that are sufficiently laconic) and indeed most of the technical difficulty arises from handling this more general setting. See additional details in Section 1.2.

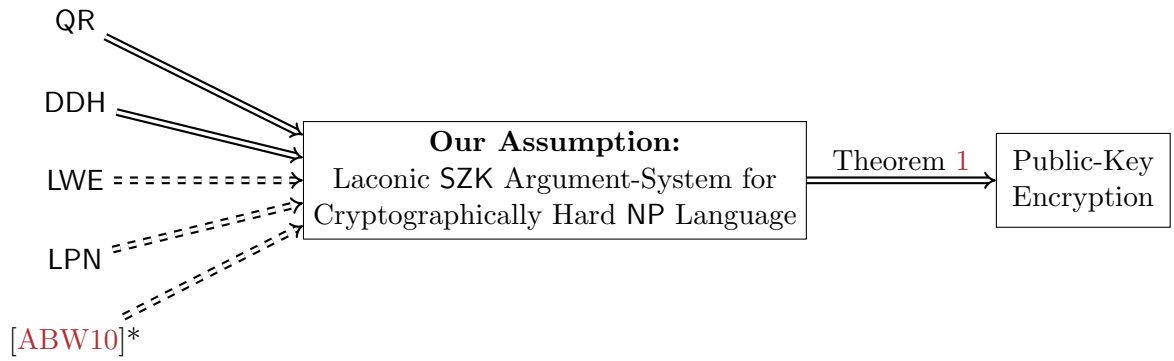

Figure 1: Instantiations of our assumption. Dashed arrows means that we only obtain average-case completeness, soundness and zero-knowledge. The (*) sign means that most, but not all, assumptions from [ABW10] imply our assumption.

With Errors (LWE) [Reg05], Learning Parity with Noise (LPN) with small errors [Ale03] and most of the assumptions used by Applebaum *et al.* [ABW10] to construct PKE, all imply the existence of such languages.

Thus, Theorem 1 gives a common framework for constructing public-key encryption based on a variety of different intractability assumptions (all of which were already known to yield public-key encryption via a variety of somewhat ad hoc techniques), see also Fig. 1.

One notable hardness assumption that we do not know to imply our assumption (even the average-case variant) is *integer factorization* (and the related RSA assumption). We consider a further weakening of our assumption that captures also the factoring and RSA assumptions. As a matter of fact, we show that this further relaxed assumption is actually *equivalent* to the existence of a public-key encryption scheme. We discuss this in more detail in Section 1.1.3.

1.1.2 Perspective — From SZK-Hardness to Public-Key Encryption

As noted above, one of the central goals in cryptography is to base public-key encryption on a general notion of structured hardness. A natural candidate for such structure is the class SZK of statistical zero-knowledge proofs, since many of the assumptions that are known to yield public-key encryption have SZK proof-systems. Indeed, it is enticing to believe that the following conjecture holds:

Conjecture 2. *Assume that there exists a cryptographically-hard language $\mathcal{L} \in \mathsf{NP} \cap \mathsf{SZK}$. Then, there exists a public-key encryption scheme.*

(Here by SZK we refer to the class of languages having statistical zero-knowledge *proof-systems* rather than argument-systems as in Theorem 1. Assuming this additional structure only makes Conjecture 2 weaker and therefore easier to prove.)

Proving Conjecture 2 would be an outstanding breakthrough in cryptography. For instance, it would allow us to base public-key cryptography on the intractability of the discrete logarithm (DLOG) problem,[5] since (a decision problem equivalent to) DLOG has a perfect zero-knowledge

[5]Public-key schemes based on assumptions related to discrete log such as the decisional (or even computational) Diffie Hellman assumption are known to exist. Nevertheless, basing public-key encryption solely on the hardness of discrete log has been open since the original work of Diffie and Hellman [DH76].

proof-system[6] [GK93], or under the plausible quasi-polynomial average-case[7] hardness of the graph isomorphism problem (via the perfect zero-knowledge protocol of [GMW87]).

We view Theorem 1 as an initial step toward proving Conjecture 2. At first glance, it seems that Theorem 1 must be strengthened in *two* ways in order to establish Conjecture 2. Namely, we need to get rid of the requirements that the (honest) prover is (1) efficient and (2) laconic. However, it turns out that it suffices to remove only *one* of these restrictions, no matter which one, in order to obtain Conjecture 2. We discuss this next.

Handling Inefficient Provers. Sahai and Vadhan [SV03] showed a problem, called *statistical distance*, which is both (1) complete for SZK, and (2) has an extremely laconic honest-verifier statistical zero-knowledge proof in which the prover only sends a *single* bit (with constant soundness error). The immediate implication is that any SZK protocol can be compressed to one in which the prover sends only a single bit.

Unfortunately, the foregoing transformation does not seem to maintain the computational efficiency of the prover. Thus, removing the requirement that the prover is efficient from Theorem 1 (while maintaining the laconism requirement) would establish Conjecture 2.

Handling Non-Laconic Provers. Suppose that we managed to remove the laconism requirement from Theorem 1 and only required the prover to be efficient. It turns out that the latter would actually imply an even stronger result than Conjecture 2. Specifically, assuming only the existence of one-way functions, Haitner *et al.* [HNO+09] construct (non-laconic) statistical zero-knowledge *arguments* for any NP language, with an efficient prover. Thus, removing the laconism requirement from Theorem 1 would yield public-key encryption based merely on the existence of one-way functions.

In fact, even a weaker result would yield Conjecture 2. Suppose we could remove the laconism requirement from Theorem 1 while insisting that the proof-system has *statistical soundness* (rather than computational). Such a result would yield Conjecture 2 since Nguyen and Vadhan [NV06] showed that every language in $\mathsf{NP} \cap \mathsf{SZK}$ has an SZK protocol in which the prover is efficient (given the NP witness).

To summarize, removing the laconism requirement from Theorem 1, while still considering an argument-system, would yield public-key encryption from one-way functions (via [HNO+09]). On the other hand, removing the laconism requirement while insisting on statistical soundness would yield Conjecture 2 (via [NV06]). (Note that neither the [NV06] nor [HNO+09] proof-systems are laconic, so they too cannot be used directly together with Theorem 1 to prove Conjecture 2.)

1.1.3 Extensions

We also explore the effect of strengthening and weakening our assumption. A natural strengthening gives us oblivious transfer, and as mentioned above, a certain weakening yields a complete complexity-theoretic characterization of public-key encryption.

[6] That proof-system is actually laconic but it is unclear how to implement the prover efficiently.

[7] Graph isomorphism is in fact known to be solvable in polynomial-time for many natural distributions, and the recent breakthrough result of Babai [Bab16] gives a quasi-polynomial worst-case algorithm. Nevertheless, it is still plausible that Graph Isomorphism is average-case quasi-polynomially hard (for some efficiently samplable distribution).

A Complexity-Theoretic Characterization. The assumption from which we construct public-key encryption (see Theorem 1) requires some underlying hard *decision* problem. In many cryptographic settings, however, it seems more natural to consider hardness of *search* problems (e.g., integer factorization). Thus, we wish to explore the setting of laconic SZK arguments when only assuming the hardness of computing a witness for an instance sampled from a solved instance generator. Namely, an NP relation for which it is hard, given a random instance, to find a corresponding witness.

We introduce a notion of (computationally sound) proof-systems for such NP search problems, which we call *arguments of weak knowledge* (AoWK). Loosely speaking, this argument-system convinces the verifier that the prover with which it is interacting has at least some partial knowledge of some witness. Or in other words, no efficient cheating prover can convince the verifier to accept given *only* the input. We further say that an AoWK is *zero-knowledge* if the verifier learns nothing beyond the fact that the prover has the witness.

We show that Theorem 1 still holds under the weaker assumption that there is an efficient and laconic SZK-AoWK (with respect to some hard solved instance generator). Namely, the latter assumption implies the existence of PKE. Furthermore, we also show that the same assumption is also *implied* by any PKE scheme, thus establishing an equivalence between the two notions which also yields a certain complexity-theoretic characterization of public-key encryption.

Oblivious Transfer. *Oblivious Transfer* (OT) is a fundamental cryptographic primitive, which is complete for the construction of general secure multiparty computation (MPC) protocols [GMW87, Kil88]. We show that by making a slightly stronger assumption, Theorem 1 can extended to yield a (two-message) semi-honest OT protocol.

For our OT protocol, in addition to the conditions of Theorem 1, we need to further assume that there is a way to sample instances x such that it is hard to tell whether $x \in \mathcal{L}$ or $x \notin \mathcal{L}$ *even given the coins of the sampling algorithm*.[8] We refer to this property as *enhanced cryptographic hardness* in analogy to the notion of *enhanced* trapdoor permutations (see further discussion in Section 6.3).

1.2 Related Works

Cryptography and Hardness of SZK. Ostrovsky [Ost91] showed that the existence of a language in SZK with average-case hardness implies the existence of one-way functions. Our result can be interpreted as an extension of Ostrovsky's result: By assuming additional structure on the underlying SZK protocol, we construct a public-key encryption scheme. In fact, some of the ideas underlying our construction are inspired by Ostrovsky's one-way function.

Average-case SZK hardness also implies constant-round statistically hiding commitments [OV08], a primitive not implied by one-way functions in a black-box way [HHRS15]. Assuming the existence of an average-case hard language in a *subclass* of SZK (i.e., of languages having perfect randomized encodings), Applebaum and Raykov [AR16] construct Collision Resistant Hash functions.

In the other direction, some cryptographic primitives like homomorphic encryption [BL13], lossy encryption and PIR (computational private information retrieval) [LV16] imply the existence of

[8] In particular, the sampling algorithm that tosses a coin $b \in \{0,1\}$ and outputs $x \in \mathcal{L}$ if $b = 0$ and $x \notin \mathcal{L}$ otherwise does not satisfy the requirement (since the value of b reveals whether $x \in \mathcal{L}$).

average-case hard problems in SZK.[9] We also mention that many other primitives, such as one-way functions, public-key encryption and oblivious transfer do not imply the existence of average-case hard problems in SZK (under black-box reductions) [BDV16].

Hash Proof-Systems. Hash Proof-Systems, introduced by Cramer and Shoup [CS02], are a cryptographic primitive which, in a nutshell, can be described as a cryptographically hard language in NP with a one-round SZK protocol in which the honest prover is efficient given the NP witness and *deterministic* (and without loss of generality sends only a single bit). This is precisely what we assume for our main result except that we can handle *randomized* provers that send more bits of information (and the protocol can be multi-round). This special case of deterministic provers is significantly simpler to handle (and will serve as a warmup when describing our techniques). Our main technical contribution is handling arbitrary *randomized* provers.

Public-key encryption schemes have been shown to imply the existence of certain *weak* hash proof-systems [HLWW16]. Hash proof-systems were also shown in [GOVW12] to yield *resettable* statistical zero-knowledge proof-systems.

Laconic Provers. A study of interactive proofs in which the prover is laconic (i.e., transmits few bits to the verifier) was initiated by Goldreich and Håstad [GH98] and was further explored by Goldreich, Vadhan and Wigderson [GVW02]. These works focus on general interactive proofs (that are not necessarily zero-knowledge) and their main results are that laconic interactive proofs are much weaker than general (i.e., non-laconic) interactive proofs.

1.3 Techniques

To illustrate the techniques used, we sketch the proof of a slightly simplified version of Theorem 1. Specifically, we construct a PKE given a cryptographically hard language $\mathcal{L}$ with a *single-round* efficient-prover and laconic SZK argument-system (we shall briefly mention the effect of more rounds where it is most relevant). For simplicity, we also assume that the SZK protocol has *perfect* completeness and zero-knowledge. In the actual construction, given in the technical sections, we handle constant completeness error, negligible simulation error, and more rounds of interaction. Lastly, since we find the presentation more appealing, rather than presenting a public-key scheme, we construct a *single-round key-agreement* protocol.[10] Any such protocol can be easily transformed into a public-key encryption scheme.

Let $\mathcal{L} \in \mathsf{NP}$ be a cryptographically hard language with an SZK argument-system with prover $\mathbf{P}$, verifier $\mathbf{V}$ and simulator $\mathbf{S}$. We assume that the argument-system has perfect completeness, no simulation error and soundness error s, for some $s > 0$. Let $\mathsf{Y}_{\mathcal{L}}$ be a solved-instance generator for $\mathcal{L}$ producing samples of the form (x, w), where $x \in \mathcal{L}$ and w is a valid witness for x. The fact that $\mathcal{L}$ is cryptographically hard means that there exists a sampler $\mathsf{N}_{\mathcal{L}}$ that generates NO instances for $\mathcal{L}$ that are computationally indistinguishable from the YES instances generated by $\mathsf{Y}_{\mathcal{L}}$.

[9] On a somewhat related note, we mention that combining [BL13] with our result gives a construction of public-key encryption from symmetric-key additively homomorphic encryption. This was already shown in [Rot11] via a direct construction.

[10] Loosely speaking, a key agreement protocol allows Alice and Bob to agree on a common key that is *unpredictable* to an external observer that has wire tapped their communication lines.

Deterministic Prover. As a warmup, we assume first that the honest prover in the SZK argument-system is *deterministic*. As will be shown below, this case is significantly easier to handle than the general case, but it is a useful step toward our eventual protocol.

We construct a key-agreement protocol between Alice and Bob as follows. First Alice generates a solved instance-witness pair $(x, w) \leftarrow \mathsf{Y}_{\mathcal{L}}$. Alice then sends x across to Bob. Bob runs the simulator $\mathbf{S}(x)$ to generate a transcript (a', b', r'), where a' corresponds to the verifier's message, b' corresponds to the prover's message and r' correspond to the simulated random string for the verifier.[11] Bob sends the first message a' across to Alice. Bob then outputs the simulated second message b'. Alice uses the witness w to generate the prover's response b (i.e., the prover $\mathbf{P}$'s actual response given the message a' from the verifier) and outputs b. The protocol is also depicted in Fig. 2.

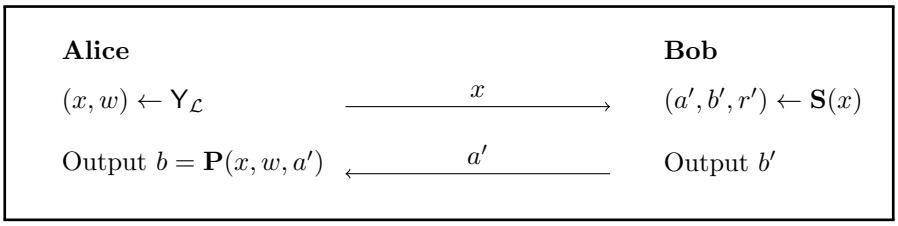

Figure 2: Key Agreement from Deterministic Provers

To argue that Fig. 2 constitutes a key-agreement protocol, we need to show that Alice and Bob output the same value, and that no efficient eavesdropper Eve (who only sees their messages) can predict this output with good probability.

That they agree on the same value follows from the fact that the prover is deterministic and the simulation is perfect. More specifically, since the simulation is perfect, the distribution of the simulated verifier's message a' is the same as that of the actual verifier's message; and now since the prover is deterministic, given (x, w, a'), the prover's response b, which is also Alice's output, is fixed. Since the simulation is perfect and $x \in \mathcal{L}$, if the simulator outputs (a', b', r'), then b', which is Bob's output, is necessarily equal to b.

Next, we show that any eavesdropper Eve who is able to guess Bob's output in the protocol can be used to break the cryptographic hardness of $\mathcal{L}$. Suppose Eve is able to guess Bob's output in the protocol with probability p. This means that given only x and a', where (a', b', r') is produced by the simulator $\mathbf{S}(x)$, Eve is able to find the message b':

$$\Pr_{\substack{(x,\cdot)\leftarrow\mathsf{Y}_{\mathcal{L}} \\ (a',b',r')\leftarrow\mathbf{S}(x)}} \left[b' = b'' \text{ where } b'' \leftarrow \mathrm{Eve}(x, a')\right] = p.$$

As the SZK argument has perfect completeness, and the simulation is also perfect, the transcripts produced by the simulator (on YES instances) are always accepted by the verifier. As Eve is able to produce the same prover messages as the simulator, her messages will also be accepted by the verifier. Namely,

$$\Pr_{\substack{(x,\cdot)\leftarrow\mathsf{Y}_{\mathcal{L}} \\ (a',b',r')\leftarrow\mathbf{S}(x)}} \left[\mathbf{V}(x, a', b''; r') = 1 \text{ where } b'' \leftarrow \mathrm{Eve}(x, a')\right] \geq p.$$

[11] Throughout this paper, we use the convention that primed symbols are for objects associated with a simulated (rather than real) execution of the protocol.

Again using the fact that the simulation is perfect, we can replace the simulated message a' and simulated coin tosses r' with a verifier message a and coins r generated by a real execution of the protocol:

$$\Pr_{\substack{(x,\cdot)\leftarrow \mathsf{Y}_{\mathcal{L}} \\ a\leftarrow \mathbf{V}(x;r)}} \left[\mathbf{V}(x,a,b'';r) = 1 \text{ where } b'' \leftarrow \mathrm{Eve}(x,a)\right] \geq p.$$

Recall that $\mathsf{N}_{\mathcal{L}}$ samples no-instances that are computationally indistinguishable from the YES instances generated by $\mathsf{Y}_{\mathcal{L}}$. If x had been a NO instance sampled using $\mathsf{N}_{\mathcal{L}}$, then the (computational) soundness of the SZK argument implies that the verifier would reject with probability $1 - s$:

$$\Pr_{\substack{x\leftarrow \mathsf{N}_{\mathcal{L}} \\ a\leftarrow \mathbf{V}(x;r)}} \left[\mathbf{V}(x,a,b'';r) = 1 \text{ where } b'' \leftarrow \mathrm{Eve}(x,a)\right] < s,$$

where s is the soundness error. If p is larger than s by a non-negligible amount, then we have a distinguisher, contradicting the cryptographic hardness of $\mathcal{L}$. So, no efficient eavesdropper can recover the agreed output value with probability noticeably more than s, the soundness error of the SZK argument.

Notice that so far we have only guaranteed that the probability of success of the eavesdropper is s, which may be as large as a constant (rather than negligible).[12] Nevertheless, using standard amplification techniques (specifically those of Holenstein and Renner [HR05]) we can compile the latter to a full-fledged key-agreement protocol.

Randomized Prover. So far we have handled deterministic provers. But what happens if the prover were *randomized*? Agreement is now in jeopardy as the prover's message b is no longer completely determined by the instance x and the verifier's message a. Specifically, after Alice receives the simulated verifier message a' from Bob, she still does not know the value of b' that Bob obtained from the simulator – if she ran $\mathbf{P}(x, w, a')$, she could get one of several possible b's, any of which could be the correct b'. Roughly speaking, Alice only has access to the *distribution* from which b' was sampled (but not to the specific value that was sampled).

Eve, however, has even less to work with than Alice; we can show, by an approach similar to (but more complex than) the one we used to show that no polynomial-time eavesdropper can guess b' in the deterministic prover case, that no polynomial-time algorithm can sample from any distribution that is close to the true distribution of b' for most x's and a''s.

We make use of this asymmetry between Alice and Eve in the knowledge of the *distribution* of b' (given x and a) to perform key agreement. We do so by going through an intermediate useful technical abstraction, which we call a *Trapdoor Pseudoentropy Generator*, that captures this asymmetry. We first construct such a generator, and then show how to use any such generator to do key agreement.

Trapdoor Pseudoentropy Generator. A distribution is said to possess *pseudoentropy* [HILL99] if it is computationally indistinguishable from another distribution that has higher entropy[13]. We

[12] This error can be made negligible by parallel repetition [BIN97] (recall that parallel repetition preserves *honest-verifier* zero-knowledge). Doing so however makes the prover's messages longer. While this is not an issue when dealing with deterministic provers, it will prove to be problematic in the general case of a randomized prover.

[13] By default, the measure of entropy employed is that of Shannon entropy. The Shannon entropy of a variable X given Y is defined as: $\mathrm{H}(X|Y) = \mathrm{E}_y\left[-\sum_x \Pr[X = x|y] \cdot \log(\Pr[X = x|y])\right]$.

will later claim that in the protocol in Fig. 2 (when used with a randomized prover), the distribution of b' has some pseudoentropy for the eavesdropper who sees only x and a'. In contrast, Alice, who knows the witness w, can *sample* from the distribution that b' was drawn from. This set of properties is what is captured by our notion of a trapdoor pseudoentropy generator.

A trapdoor pseudoentropy generator consists of three algorithms. The key generation algorithm KeyGen outputs a public and secret key pair $(\mathsf{pk}, \mathsf{sk})$. The *encoding*, given a public key pk, outputs a pair of strings (u, v), where we call u the public message and v the private message.[14] The *decoding* algorithm Dec, given as input the corresponding secret key and the public message u, outputs a value v'. These algorithms are required to satisfy the following properties (simplified here for convenience):

- **Correctness:** The distributions of v and v' are identical, given pk, sk, and u.
- **Pseudoentropy:** The distribution of v has some pseudoentropy given pk and u.

Correctness here only means that the secret key can be used to sample from the distribution of the private message v corresponding to the given public message u. This captures the weaker notion of agreement observed in the protocol earlier when Alice had sampling access to the distribution of Bob's output.

The pseudoentropy requirement says that without knowledge of the secret key, the private message v seems to have more entropy – it looks "more random" than it actually is. This is meant to capture the asymmetry of knowledge between Alice and Eve mentioned earlier.

Constructing a Trapdoor Pseudoentropy Generator. Our construction of a trapdoor pseudoentropy generator is described in Fig. 3. It is an adaptation of the earlier key exchange protocol for deterministic provers (from Fig. 2). The public key is an instance x in the language $\mathcal{L}$ and the corresponding secret key is a witness w for x – these are sampled using the solved-instance generator. To encode with public key x, the simulator from the SZK argument for $\mathcal{L}$ is run on x and the simulated verifier message a' is set to be the public message, while the simulated prover message b' is the private message. To decode given x, w and a', the actual prover is run with this instance, witness and verifier message, and the response it generates is output.

$\underline{\mathsf{KeyGen}}$	$\underline{\mathsf{Enc}(\mathsf{pk} = x)}$	$\underline{\mathsf{Dec}(\mathsf{pk} = x, \mathsf{sk} = w, u = a')}$
1. Sample $(x, w) \leftarrow \mathsf{Y}_{\mathcal{L}}$	1. Sample $(a', b', r) \leftarrow \mathbf{S}(x)$	1. Sample $v' \leftarrow \mathbf{P}(x, w, a')$
2. Output $(\mathsf{pk} = x, \mathsf{sk} = w)$	2. Output $(u = a', v = b')$	2. Output v'

Figure 3: Trapdoor Pseudoentropy Generator

Now we argue that this is a valid pseudoentropy generator. Since we will need to be somewhat precise, for the rest of this section, we introduce the jointly-distributed random variables X, A and B, where X represents the instance (sampled from $\mathsf{Y}_{\mathcal{L}}$), A represents the verifier's message (with respect to X), and B represents the prover's response (with respect to X and A). Note that since the simulation in the SZK argument is perfect, A and B represent the distributions of the messages output by the simulator as well.

[14] We refer to this procedure as an encoding algorithm because we think of the public message as an encoding of the private message.

The correctness of our construction follows from the perfect zero knowledge of the underlying SZK argument – the private message v produced by Enc here is the simulated prover's message b', while the output of Dec is the actual prover's response b with the same instance and verifier's message. Both of these have the same distribution, which corresponds to that of B conditioned on $X = x$ and $A = a'$.

In order to satisfy the pseudoentropy condition, the variable B needs to have some pseudoentropy given X and A. What we know, as mentioned earlier, is that B is *unpredictable* given X and A – that no polynomial-time algorithm, given x and a', can sample from a distribution close to that of the corresponding prover's message b. Towards this end, we will use a result of Vadhan and Zheng [VZ12], who give a tight equivalence between unpredictability and pseudoentropy. Applied to our case, their results say what we want – that the variable B has additional pseudoentropy $\log(1/s)$ given X and A, where s is the soundness error from the SZK argument. More precisely, there exists a variable C such that:

$$(X, A, B) \approx_c (X, A, C) \quad \text{and} \quad \mathrm{H}(C|X, A) > \mathrm{H}(B|X, A) + \log(1/s), \tag{1}$$

where the above expressions refer to *Shannon entropy*. The result of Vadhan and Zheng applies only when the variable B has a polynomial-sized domain, which holds since the proof-system is *laconic* (this is the first out of several places in which we use the laconism of the proof-system). The above shows that the construction in Fig. 3 is indeed a trapdoor pseudoentropy generator. Finally, and this will be crucial ahead, note that the private message produced by Enc is short (i.e., the same length as the prover's message in the SZK argument we started with).

In the case of an SZK protocol with r rounds, the above construction would be modified as follows. The encoder Enc samples a transcript from $\mathbf{S}(x)$, picks $i \in [r]$ at random, sets the public message u to be all the messages in the transcript upto the verifier's message in the i^{th} round, and the private message v to be the prover's message in the i^{th} of the transcript. The decoder Dec samples v' by running the prover on the partial transcript u to get the actual prover's response in the i^{th} round.[15] Zero knowledge ensures that v' and v are distributed identically, and unpredictability arguments similar to the ones above tell us that v' has pseudoentropy at least $\log(1/s)/r$.

From Laconic Trapdoor Pseudoentropy Generator to Key Agreement. Next, given a trapdoor pseudoentropy generator, such as the one in Fig. 3, we show how to construct a single-round key agreement protocol. We start with a pseudoentropy generator in which the public key is pk, the private key is sk, the public message is u, the private message is v, and the output of Dec is v'. The random variables corresponding to these are the same symbols in upper case. v and v' come from the distribution $V_{\mathsf{pk},u}$ (V conditioned on $PK = \mathsf{pk}$ and $U = u$), and V has additional pseudo-Shannon-entropy η given PK and U, where η can be thought of as a constant (η was $\log(1/s)$ in the foregoing construction).

In the key agreement protocol, first Alice samples a key pair $(\mathsf{pk}, \mathsf{sk})$ for the pseudoentropy generator and sends the public key pk to Bob. Bob runs $(u, v) \leftarrow \mathsf{Enc}(\mathsf{pk})$, keeps the private message v and sends the public message u to Alice. We would like for Alice and Bob to agree on the string v. In order for this to be possible, Bob needs to send more information to Alice so as to specify the specific v that was sampled from $V_{\mathsf{pk},u}$. A natural idea is for Bob to send, along with

[15] For simplicity, assume that the prover is stateless so it can be run on a partial transcript. In the actual proof we handle stateful provers as well.

the message u, a hash $h(v)$ of v, where h is a sampled from a pairwise independent hash function family $\mathcal{H}$.

Alice, on receiving the hash function h and the hash value $h(v)$, uses rejection sampling to find v. She can sample freely from the distribution $V_{\mathsf{pk},u}$ by running $\mathsf{Dec}(\mathsf{sk}, u)$ because she knows the secret key sk of the pseudoentropy generator and the public message u. She keeps drawing samples v' from $V_{\mathsf{pk},u}$, until she finds one that hashes to $h(v)$. Note that this brute force search is only feasible if the number of strings in the support of V is small, which is the case if the number of bits in v is small – considering the big picture, this is one of the reasons we want the the prover from the SZK argument to be laconic.

The main question now is how to set the length of the hash function. On the one hand, having a long hash helps agreement, as more information is revealed to Alice about v. On the other hand, security demands a short hash that does not leak "too much" information about v.

For agreement, roughly speaking, if the hash length were more than the *max-entropy*[16] of V given PK and U, which we denote by $\mathrm{H}_{\max}(V|PK, U)$, then the set of possible prover responses is being hashed to a set of comparable size, so with good probability, the hash value $h(v)$ will have a unique pre-image, which Alice can identify.

For security we would like to argue, using the Leftover Hash Lemma, that to any eavesdropper $h(v)$ looks uniformly random given (pk, u, h). This would be true if the hash length were less than the *min-entropy*[17] of V given PK and U, which we denote by $\mathrm{H}_{\min}(V|PK, U)$. Unfortunately, both of the above conditions cannot hold simultaneously because the min-entropy is upper-bounded by the max-entropy.

The crucial observation at this point is that Eve is computationally bounded. Hence, a *computational* analogue of high min-entropy, which we will call *pseudo-min-entropy*, would suffice for security. Concretely, consider a random variable C such that (PK, U, C) is computationally indistinguishable from (PK, U, V). Furthermore, suppose that the min-entropy of C given PK and U is considerably larger than the hash length. We can then use the Leftover Hash Lemma to argue that $h(V)$ looks uniform to efficient eavesdroppers:

$$(PK, U, h, h(V)) \approx_c (PK, U, h, h(C)) \approx_s (PK, U, h, R)$$

where R is the uniform distribution over the range of h.

The benefit of this observation is that, since C is only required to be computationally close and not statistically close to V, the min-entropy of C given PK and U could be much larger than that of V given PK and U. And if we can find a C such that $\mathrm{H}_{\min}(C|PK, U)$ is sufficiently larger than $\mathrm{H}_{\max}(V|PK, U)$, then we will indeed be able to choose a hash length that is both large enough for agreement and small enough for security.

Also notice that for the agreement to work, it is not necessary for the hash length to be larger than the max-entropy of V (given PK and U) itself – instead, if there was another variable D such that (PK, U, D) is *statistically* close to (PK, U, V), and also Alice is somehow able to sample from D given $PK = \mathsf{pk}$ and $U = u$, then it is sufficient for the hash to be longer than $\mathrm{H}_{\max}(D|PK, U)$. Given such a variable, Bob will operate as he did earlier, but Alice can assume that he is actually sampling from $D_{\mathsf{pk},u}$ instead of $V_{\mathsf{pk},u}$, and since these two distributions are close most of the time, the probability of Alice's subsequent computation going wrong is small. This helps us because now

[16] The max entropy corresponds to the logarithm of the support size. The *conditional* max entropy of a random variable X given Y is defined as: $\mathrm{H}_{\max}(X|Y) = \max_y \log(|\operatorname{Supp}(X|Y = y)|)$.

[17] The min-entropy of a variable X given Y is defined as: $\mathrm{H}_{\min}(X|Y) = -\log(\max_{x,y} \Pr[X = x|Y = y])$.

we might be able to find such a D that has lower max-entropy given PK and U than V, and then $\mathrm{H}_{\min}(C|PK,U)$ would only have to be larger than this.

Following these observations, we set ourselves the following objective: find variables C and D such that:

$$(PK,U,D) \approx_s (PK,U,V) \approx_c (PK,U,C) \quad \text{and} \quad \mathrm{H}_{\max}(D|PK,U) < \mathrm{H}_{\min}(C|PK,U) \tag{2}$$

What we do know about V is that it has some pseudo-Shannon-entropy given PK and U. That is, there is a variable C such that:

$$(PK,U,V) \approx_c (PK,U,C) \quad \text{and} \quad \mathrm{H}(C|PK,U) > \mathrm{H}(V|PK,U) + \eta \tag{3}$$

The rest of our construction deals with using this pseudo-Shannon-entropy to achieve the objectives above. This we do using a technique from Information Theory dating back to Shannon [Sha48] which is often referred to in the cryptography literature as *flattening of distributions*, which we describe next. We note that this technique has found use in cryptography before [HILL99, GV99, SV03].

Flattening and Typical Sets. The central idea here is that if we start with a distribution that has Shannon entropy ξ and repeat it k times, then the new distribution is close to being uniform on a set whose size is roughly $2^{k\xi}$. This set is called the *typical set*; it consists of all elements whose probability is close to $2^{-k\xi}$.

In our case, consider the distribution (PK^k, U^k, V^k), which is the k-fold product repetition of (PK,U,V). Roughly speaking, we define the *typical set* of V^k conditioned on any $(\mathbf{pk},\mathbf{u})$ in the support[18] of (PK^k,U^k) as follows[19]:

$$\mathcal{T}_{V^k|\mathbf{pk},\mathbf{u}} = \left\{ \mathbf{v} : \Pr\left[V^k = \mathbf{v} \middle| (PK^k,U^k) = (\mathbf{pk},\mathbf{u})\right] \approx 2^{-k\,\mathrm{H}(V|PK,U)} \right\}$$

Considering the typical set is useful for several reasons. On the one hand, the typical set is quite small (roughly $2^{k\,\mathrm{H}(V|PK,U)}$) in size, which means that any distribution supported within it has somewhat low max-entropy. On the other hand, there is an upper bound on the probability of any element that occurs in it, which could be useful in lower bounding min-entropy, which is what we want to do.

The most important property of the typical set it that it contains most of the probability mass of the conditional repeated distribution. That is, for most $(\mathbf{pk},\mathbf{u},\mathbf{v})$ sampled from (PK^k,U^k,V^k), it holds that $\mathbf{v}$ lies in the typical set conditioned on $(\mathbf{pk},\mathbf{u})$; quantitatively, Holenstein and Renner [HR11] show the following:

$$\Pr_{(\mathbf{pk},\mathbf{u},\mathbf{v})\leftarrow(PK^k,U^k,V^k)}\left[\mathbf{v} \notin \mathcal{T}_{V^k|\mathbf{pk},\mathbf{u}}\right] < 2^{-\Omega(k/q^2)} \tag{4}$$

where q is the number of bits in each sample from V. Recall that in our earlier construction of the trapdoor pseudoentropy generator, this corresponds to the length of the prover's message in the SZK argument we started with. We want the above quantity to be quite small, which requires that $k \gg q^2$. This is one of the considerations in our ultimate choice of parameters, and is another reason we want the prover's messages to not be too long.

[18] The support of (PK^k,U^k) consists of vectors with k elements. We represent vectors by bold symbols, e.g., $\mathbf{v}$.

[19] The actual definition quantifies how different from 2^{-kH} the probability is allowed to be.

Back to PKE Construction. We shall use the above facts to now show that V^k has pseudo-min-entropy given PK^k and U^k. Let C be the random variable from the expression (3) above that we used to show that V has pseudo-Shannon-entropy. After repetition, we have that:

$$(PK^k, U^k, V^k) \approx_c (PK^k, U^k, C^k) \quad \text{and} \quad \mathrm{H}(C^k|PK^k, U^k) = k \cdot \mathrm{H}(C|PK, U) > k \cdot (\mathrm{H}(V|PK, U) + \eta).$$

Next, consider the variable C' that is obtained by restricting, for each $\mathbf{pk}$ and $\mathbf{u}$, the variable C^k to its typical set conditioned on $(\mathbf{pk}, \mathbf{u})$. By applying the bound of Holenstein and Renner (4) with an appropriate choice of k, we infer that:

$$(PK^k, U^k, C^k) \approx_s (PK^k, U^k, C').$$

Further, the upper bound on the probabilities of elements in the typical set tells us that C' has high min-entropy[20] given PK^k and U^k:

$$\mathrm{H}_{\min}(C'|PK^k, U^k) \approx \mathrm{H}(C^k|PK^k, U^k) \geq k \cdot (\mathrm{H}(V|PK, U) + \eta).$$

Putting the above few expressions together tells us that V^k has some pseudo-min-entropy given PK^k and U^k, which is in fact somewhat more than its Shannon entropy:

$$(PK^k, U^k, V^k) \approx_c (PK^k, U^k, C') \quad \text{and} \quad \mathrm{H}_{\min}(C'|PK^k, U^k) \gtrsim \mathrm{H}(V^k|PK^k, U^k) + k \cdot \eta. \tag{5}$$

This satisfies our objective of getting a variable – V^k here – that has high pseudo-min-entropy (given PK^k and U^k). Our goal is now to find another variable that is statistically close to V^k given PK^k and U^k, and also has small max-entropy given PK^k and U^k. We do this using the same approach as above. Consider the variable V' that is constructed from V^k in the same way C' was from C^k – for each $(\mathbf{pk}, \mathbf{u})$, restrict V^k to its typical set conditioned on $(\mathbf{pk}, \mathbf{u})$. Again, bound (4) tells us that the new distribution is close to the old one. And also, because of the upper bound on the size of the typical set, we have an upper bound on the max-entropy[21] of V' given PK^k and U^k.

$$(PK^k, U^k, V^k) \approx_s (PK^k, U^k, V') \quad \text{and} \quad \mathrm{H}_{\max}(V'|PK^k, U^k) \lesssim \mathrm{H}(V^k|PK^k, U^k). \tag{6}$$

Putting together expressions (5) and (6), we find that the relationship we want between these entropies of C' and V' is indeed satisfied:

$$\mathrm{H}_{\min}(C'|PK^k, U^k) \gtrsim \mathrm{H}_{\max}(V'|PK^k, U^k) + k \cdot \eta.$$

To summarize, we manage to meet the conditions of expression (2) with respect to (PK^k, U^k, V^k) (instead of (PK, U, V)) with C' taking the role of C and V' taking the role of D. We can now finally fix the length of our hash – call it ℓ – to be between $\mathrm{H}_{\max}(V'|PK^k, U^k)$ and $\mathrm{H}_{\min}(C'|PK^k, U^k)$, which can be done by setting it to a value between $\mathrm{H}(V^k|PK^k, U^k)$ and $\mathrm{H}(V^k|PK^k, U^k) + k\eta$ for an appropriate k, and emulate the earlier protocol. We will be able to use the Leftover Hash Lemma as desired to argue security and use the low max-entropy of V' to argue agreement.

The final key agreement protocol from a trapdoor pseudoentropy generator is presented in Fig. 4.

[20] $\mathrm{H}_{\min}(C'|PK^k, U^k)$ could actually be slightly less than the approximate lower bound presented here because there is some slack allowed in the definition of the typical set – it can contain elements whose probabilities are slightly larger than $2^{-k\,\mathrm{H}(C|PK,U)}$. We need to pick this slack carefully – if it is too large, C' loses its min-entropy, and if it is too small the typical set also becomes too small and the bound in (4), which actually depends on this slack, becomes meaningless. This is another constraint on our choice of parameters.

[21] The same caveats as in Footnote 20 regarding the min-entropy of C' apply here as well.

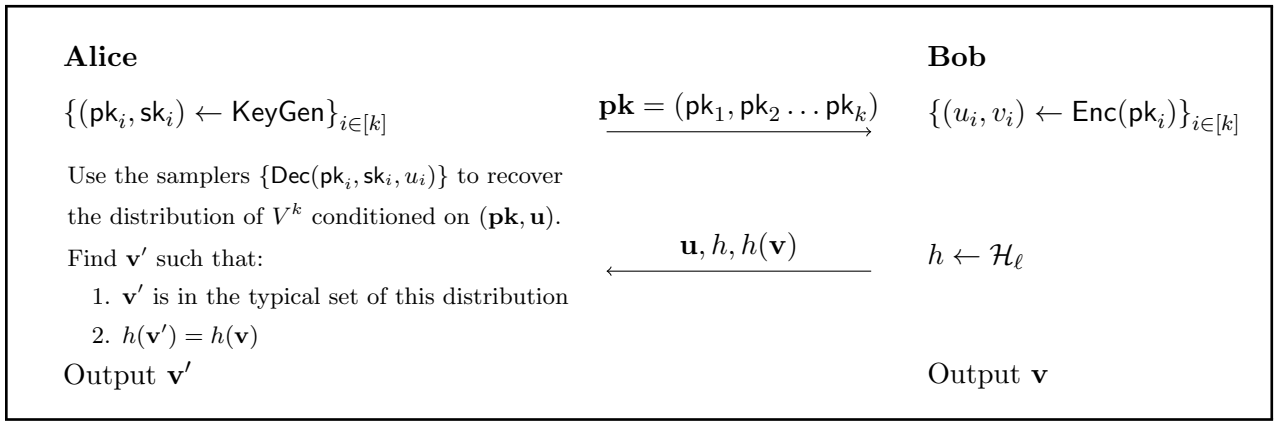

Figure 4: Key Agreement from Trapdoor Pseudoentropy Generator

How Laconic? To examine how long the prover's message can be, lets recall the restrictions of our construction. First, we need both parties to be efficient. While Bob is clearly efficient, Alice performs an exhaustive search over the domain of possible prover messages. The size of this domain is $2^{q\cdot k}$ because the parties repeat the underlying protocol k times and the length of each prover's message is q bits. For Alice to be efficient, this domain has to be polynomial-sized, requiring that $q \cdot k = O(\log n)$, where n is the input length. Second, we need that the concentration bound for the typical set (Eq. (4)) to be meaningful; that is, we need k/q^2 to be at least a constant. Together, these imply that q^3 needs to be $O(\log n)$. Lastly, this setting of parameters also suffices for the [VZ12] result that we used in Eq. (1).

1.4 Organization

In Section 2 we describe notions from cryptography and information theory that we need. In Section 3 we formally describe and state our assumption and our main theorem. In Section 4 we define and construct a trapdoor pseudoentropy generator. In Section 5 we use the latter to construct a public-key encryption. In Section 6 we describe various extensions: that certain relaxations of our assumption also yield public-key encryption, that a mild strengthening of our assumptions yields a single-round oblivious transfer protocol, and that many concrete assumptions used to construct public-key encryption in the past also imply our assumptions.

2 Preliminaries

In this section we recall notions from cryptography and information theory that will be used throughout this work.

Notation and Conventions. We use lowercase letters for values, uppercase for random variables, uppercase calligraphic letters (e.g., $\mathcal{U}$) to denote sets, boldface for vectors (e.g., $\mathbf{x}$), and uppercase sans-serif (e.g., A) for algorithms (i.e., Turing Machines). All logarithms considered here are in base two. Given a probabilistic polynomial-time algorithm A, we let $\mathsf{A}(x;r)$ be an execution of A on input x given randomness r. We let poly denote the set all polynomials. A function

$\nu\colon \mathbb{N} \to [0,1]$ is *negligible*, denoted $\nu(n) = \mathsf{negl}(n)$, if $\nu(n) < 1/p(n)$ for every $p \in \mathsf{poly}$ and large enough n.

Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X. Similarly, given a finite set $\mathcal{S}$, we let $s \leftarrow \mathcal{S}$ denote that s is selected according to the uniform distribution on $\mathcal{S}$. We adopt the convention that when the same random variable occurs several times in an expression, all occurrences refer to a single sample. For example, $\Pr[f(X) = X]$ is defined to be the probability that when $x \leftarrow X$, we have $f(x) = x$. We write U_n to denote the random variable distributed uniformly over $\{0,1\}^n$. The support of a distribution D over a finite set $\mathcal{U}$, denoted $\mathrm{Supp}(D)$, is defined as $\{u \in \mathcal{U} : D(u) > 0\}$. The *statistical distance* of two distributions P and Q over a finite set $\mathcal{U}$, denoted as $\mathrm{SD}(P,Q)$, is defined as $\max_{\mathcal{S}\subseteq\mathcal{U}} |P(\mathcal{S}) - Q(\mathcal{S})| = \frac{1}{2}\sum_{u\in\mathcal{U}} |P(u) - Q(u)|$. The *data-processing inequality for statistical distance* states that for any randomized procedure F, it holds that $\mathrm{SD}(F(P), F(Q)) \leq \mathrm{SD}(P,Q)$.

The *k-fold product repetition* of a random variable X is the random variable X^k such that for every $\mathbf{x} = (x_1, \ldots, x_k)$, it holds that $\Pr[X^k = \mathbf{x}] = \prod_{i=1}^{k} \Pr[X = x_i]$. For sake of notational convenience, for jointly distributed random variables $(X_1, \ldots, X_\ell)$, we use $(X_1^k, \ldots, X_\ell^k)$ to denote the k-fold product repetition $(X_1, \ldots, X_\ell)^k$.

2.1 Public Key Encryption

In this section, we recall the definition of semantic-security [GM84] for public-key encryption (PKE). We shall restrict our attention to bit-encryption schemes (i.e., schemes in which only single bit messages are encrypted) and note that the latter implies full-fledged public-key encryption (c.f., [Gol09]).

Our definition includes parameters α and β which correspond, respectively, to the correctness and security errors.

Definition 2.1 (Public Key Encryption)**.** *An α-correct β-secure public key encryption scheme is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Gen}(1^\lambda)$ outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$, the encryption algorithm $\mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)$ outputs a ciphertext ct (given the message $\sigma \in \{0,1\}$ and the public key pk) and the decryption algorithm $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ returns a decrypted message (given the secret key sk and the ciphertext ct). The scheme satisfies the following properties:*

- **Correctness:** *For all sufficiently large $\lambda \in \mathbb{N}$:*

$$\Pr\Big[\mathsf{Dec}\big(1^\lambda, \mathsf{sk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\big) = \sigma\Big] > \frac{1+\alpha(\lambda)}{2},$$

  where the probability is over $\sigma \leftarrow \{0,1\}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and the randomness of Enc and Dec.

- **Semantic Security:** *For every probabilistic polynomial-time adversary A and sufficiently large $\lambda \in \mathbb{N}$, it holds that*

$$\Pr\Big[\mathsf{A}\big(1^\lambda, \mathsf{pk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\big) = \sigma\Big] < \frac{1+\beta(\lambda)}{2},$$

  where the above probability is over $\sigma \leftarrow \{0,1\}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and the randomness of Enc and A.

If a scheme is α-correct β-secure for some constants $\alpha > \beta > 0$, then we say the scheme is a weak public-key encryption scheme. *If such a scheme is $(1 - 1/\lambda^c)$-correct $(1/\lambda^c)$-secure for every $c > 0$, we say that it is a* semantically secure public-key encryption scheme.

For some parameters of α and β, weak public-key encryption schemes can be amplified to semantically secure public-key encryption schemes

Theorem 2.2 ([HR05, Theorem 6])**.** *Let α and β be constants such that $\alpha^2 > \beta$. If there exists an α-correct β-secure public-key encryption scheme, then there exists a semantically secure public-key encryption scheme.*

2.2 Universal Hashing

Universal hash functions are used extensively in complexity theory and cryptography.

Definition 2.3 (Universal Hash Function)**.** *A family of functions $\mathcal{H} = \{h : [N] \to [M]\}$ is* Universal *if for every distinct $x_1, x_2 \in [N]$, it holds that*

$$\Pr_{h \leftarrow \mathcal{H}}\left[h(x_1) = h(x_2)\right] = \frac{1}{M}.$$

Fact 2.4 (c.f. [Vad12, Theorem 3.26])**.** *For every $n, m \in \mathbb{N}$, there exists a family of universal hash functions $\mathcal{H}_{n,m} = \{h \colon \{0,1\}^n \to \{0,1\}^m\}$ where a random function from $\mathcal{H}_{n,m}$ can be selected using $\max(m, n) + m$ bits, and given a description of $h \in \mathcal{H}_{n.m}$ and $x \in \{0,1\}^n$, the value $h(x)$ can be evaluated in time* $\mathsf{poly}(n, m)$.

2.3 Entropy and Divergence

Concepts from information theory, including various notions of entropy, play a pivotal role in this paper.

Definition 2.5 (Shannon, Conditional, and Min Entropies)**.** *Let X be a random variable taking values in a discrete alphabet $\mathcal{X}$. The* entropy *of X is defined as*

$$\mathrm{H}(X) = \sum_{x \in \mathrm{Supp}(X)} \Pr[X = x] \cdot \log\left(\frac{1}{\Pr[X = x]}\right) = \mathop{\mathrm{E}}_{x \leftarrow X}\left[\log\left(\frac{1}{\Pr[X = x]}\right)\right].$$

Let Y be a random variable taking values in a discrete alphabet $\mathcal{Y}$, which is jointly distributed with X. The conditional entropy *of X given Y is defined as*

$$\mathrm{H}(X|Y) = \mathop{\mathrm{E}}_{y \leftarrow Y}[\mathrm{H}(X|Y = y)] = \mathop{\mathrm{E}}_{(x,y) \leftarrow (X,Y)}\left[\log\left(\frac{1}{\Pr[X = x \mid Y = y]}\right)\right].$$

Finally, the min-entropy *of X is defined as*

$$\mathrm{H}_\infty(X) = \min_{x \in \mathcal{X}} \log\left(\frac{1}{\Pr[X = x]}\right)$$

We recall some basic facts about entropy.

Fact 2.6 (Chain Rule for Entropy)**.** *For any jointly distributed random variables X and Y it holds that* $\mathrm{H}(X,Y) = \mathrm{H}(X) + \mathrm{H}(Y|X)$.

Fact 2.7 (Conditioning does not Increase Entropy)**.** *For any jointly distributed random variables X and Y it holds that* $\mathrm{H}(X|Y) \leq \mathrm{H}(X)$.

Fact 2.8 ([Vad99, Fact 3.3.9])**.** *For any two random variables X and Y, taking values in $\mathcal{U}$, it holds that*

$$|\mathrm{H}(X) - \mathrm{H}(Y)| \leq \log(|\mathcal{U}|) \cdot \gamma + h(\gamma),$$

where $\gamma = \mathrm{SD}(X,Y)$ and $h(p) = p \cdot \log(1/p) + (1-p) \cdot \log(1/(1-p))$ is the binary entropy function.

Fact 2.9. *For any three jointly distributed random variables X, Y and Z, such that X and Y take values in $\mathcal{U}$, it holds that*

$$|\mathrm{H}(X|Z) - \mathrm{H}(Y|Z)| \leq \log(|\mathcal{U}|) \cdot \gamma + h(\gamma),$$

where $\gamma = \mathrm{SD}((X,Z),(Y,Z))$.

Proof. Let $\gamma_z = \mathrm{SD}((X|Z=z),(Y|Z=z))$. It holds that

$$\begin{aligned}
\mathrm{H}(X|Z) - \mathrm{H}(Y|Z) &= \mathop{\mathrm{E}}_{z \leftarrow Z}[\mathrm{H}(X|Z=z) - \mathrm{H}(Y|Z=z)] \\
&\leq \mathop{\mathrm{E}}_{z \leftarrow Z}[\log(|U|) \cdot \gamma_z + h(\gamma_z)] \\
&= \log(|U|) \cdot \mathop{\mathrm{E}}_{z \leftarrow Z}[\gamma_z] + \mathop{\mathrm{E}}_{z \leftarrow Z}[h(\gamma_z)] \\
&\leq \log(|U|) \cdot \mathop{\mathrm{E}}_{z \leftarrow Z}[\gamma_z] + h\left(\mathop{\mathrm{E}}_{z \leftarrow Z}[\gamma_z]\right) \\
&= \log(|U|) \cdot \gamma + h(\gamma),
\end{aligned}$$

where the first inequality follows from Fact 2.8, the second inequality follows from Jensen's inequality on the concave function $h(\cdot)$ (i.e., the binary entropy function), and the last equality follows from the fact that $\mathrm{E}_{z \leftarrow Z}[\gamma_z] = \gamma$. We can bound $\mathrm{H}(Y|Z) - \mathrm{H}(X|Z)$ similarly and Fact 2.9 follows. □

We use *Divergence* to measure "distance" between distributions (or random variables).

Definition 2.10 (Divergence (aka Kullback-Leibler divergence, aka relative entropy))**.** *Let X and Y be random variables taking values in a discrete alphabet $\mathcal{X}$. The* divergence *from X to Y is defined as*

$$\mathrm{KL}(X||Y) = \sum_{x \in \mathcal{X}} \Pr[X=x] \cdot \log\left(\frac{\Pr[X=x]}{\Pr[Y=x]}\right),$$

or ∞ if $\Pr[X=x] > 0 = \Pr[Y=x]$ for some $x \in \mathcal{X}$, with the convention that $0 \log \frac{p}{0} = 0$.

For $p, q \in [0,1]$, the binary divergence *from p to q is defined as $\mathrm{KL}(p||q) = \mathrm{KL}(X||Y)$, for $X \sim \text{Bernoulli}(p)$ and $Y \sim \text{Bernoulli}(q)$.*

The following facts are well-known:

Fact 2.11 (Chain Rule for Divergence, see [PW16, Theorem 2.2(4)])**.** *It holds that*

$$\begin{aligned}\mathrm{KL}(X_1, X_2, \ldots, X_k || Y_1, Y_2, \ldots, Y_k) &= \sum_{i=1}^{k} \mathop{\mathrm{E}}_{x \leftarrow X^{i-1}} [\mathrm{KL}(X_i|_{X^{i-1}=x} || Y_i|_{Y^{i-1}=x})] \\ &= \sum_{i=1}^{k} \mathop{\mathrm{E}}_{X^{i-1}} [\mathrm{KL}(X_i|_{X^{i-1}} || Y_i|_{X^{i-1}})],\end{aligned}$$

where $X^i = (X_1, \ldots, X_i)$ *and* $Y^i = (Y_1, \ldots, Y_i)$*.*

Fact 2.12 (Data-Processing for Divergence, see [PW16, Theorem 2.2(6)])**.** *For any (possibly randomized) process* P*, it holds that*

$$\mathrm{KL}(\mathsf{P}(X) || \mathsf{P}(Y)) \leq \mathrm{KL}(X||Y).$$

The following two facts relate to the *binary* divergence function as defined above. Using the fact that the mapping $(p, q) \mapsto \mathrm{KL}(p||q)$ is convex (see [PW16, Theorem 4.1]) and its minimum is attained at the line $p = q$ it follows that:

Fact 2.13. *For every* $1 \geq p \geq p' \geq q' \geq q \geq 0$*, it holds that*

$$\mathrm{KL}\big(p' \big|\big| q'\big) \leq \mathrm{KL}(p||q),$$

and equality holds iff $p = p'$ *and* $q = q'$*.*

Finally, we note that slightly increasing the second argument for the binary divergence function does not change its value by much.

Fact 2.14. *For every* $p, q \in (0, 1)$ *and* $\gamma \in [0, 1 - q)$*, it holds that*

$$\mathrm{KL}(p||q) - \mathrm{KL}(p||q + \gamma) \leq 2 \cdot \frac{p}{q} \cdot \gamma.$$

Proof. Straightforward calculations show that

$$\begin{aligned}\mathrm{KL}(p||q) - \mathrm{KL}(p||q+\gamma) &= p \cdot \log\left(\frac{p}{q}\right) + (1-p) \cdot \log\left(\frac{1-p}{1-q}\right) \\ &\quad - p \cdot \log\left(\frac{p}{q+\gamma}\right) - (1-p) \cdot \log\left(\frac{1-p}{1-(q+\gamma)}\right) \\ &= p \cdot \log\left(\frac{q+\gamma}{q}\right) + (1-p) \cdot \log\left(\frac{1-q-\gamma}{1-q}\right) \\ &\leq p \cdot \log\left(1 + \frac{\gamma}{q}\right) \\ &\leq 2 \cdot \frac{p}{q} \cdot \gamma,\end{aligned}$$

where the first inequality follows by removing negative terms and the second since $\log(1 + x) \leq 2x$ for any $x \geq 0$. □

2.4 Pseudoentropy

Intuitively, a distribution Y has pseudoentropy if there exists a high-entropy distribution Z that is indistinguishable from Y. We are interested in conditional pseudoentropy, which refers to a joint distribution (X, Y), and stipulates the existence of a distribution Z, jointly distributed with (X, Y), such that (1) $Z|X$ has high (conditional) entropy, and (2) (X, Y) is computationally indistinguishable from (X, Z). When considering non-uniform adversaries, the intuition translates directly into a definition. In the uniform case, however, the definition is slightly more complicated. Since, we prefer to show *uniform* reductions, we present that definition, and discuss why the additional complications immediately after the definition.

Definition 2.15 (Conditional Pseudoentropy (c.f. [VZ12, Definition 2.12]))**.** *Let $t = t(\lambda) \in \mathbb{N}$, $\varepsilon = \varepsilon(\lambda) \in [0,1]$ and $m = m(\lambda) \geq 0$. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$. We say that Y has (t, ε)* conditional pseudoentropy at least *m* given *X if for every oracle-aided probabilistic algorithm* A *that on input $(1^\lambda, x, y)$ runs in time $t(\lambda)$, there is a sequence of random variables $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$ over $\mathcal{Y}_\lambda$, jointly distributed with X, Y, such that the following hold for large enough $\lambda \in \mathbb{N}$:*

1. $\mathrm{H}(Z_\lambda|X_\lambda) \geq m(\lambda)$;
2. *Let $O_{X'_\lambda,Y'_\lambda,Z'_\lambda}$ denote an oracle that returns random independent samples from $(X'_\lambda, Y'_\lambda, Z'_\lambda)$ when queried, where $(X'_\lambda, Y'_\lambda, Z'_\lambda)$ are identically distributed as $(X_\lambda, Y_\lambda, Z_\lambda)$. Then, it holds that*

$$\left|\Pr\left[\mathsf{A}^{O_{X'_\lambda,Y'_\lambda,Z'_\lambda}}(1^\lambda, X_\lambda, Y_\lambda) = 1\right] - \Pr\left[\mathsf{A}^{O_{X'_\lambda,Y'_\lambda,Z'_\lambda}}(1^\lambda, X_\lambda, Z_\lambda) = 1\right]\right| \leq \varepsilon(\lambda),$$

where the above probabilities are over $X_\lambda, Y_\lambda, Z_\lambda$, the random coins of A and the samples generated by $O_{X'_\lambda,Y'_\lambda,Z'_\lambda}$.

We say that Y has conditional pseudoentropy at least *m* given *X if for every constant $c > 0$, Y has $(\lambda^c, 1/\lambda^c)$ conditional pseudoentropy at least $m - 1/\lambda^c$ given X.*

The reason that the oracle samples from (X', Y', Z') (and not (X, Y, Z)) is to emphasize its samples are independent from the inputs given to the distinguisher A (namely (X, Y, Z)). The reason to give the distinguisher oracle access to samples from the distributions is to ensure that repetition preserves pseudoentropy. Intuitively, assume that X is indistinguishable from Y and $\mathrm{H}(Y) \geq m$, namely X has pseudoentropy at least m. We would like that if (X_1, X_2) are independent copies of X, they would have pseudoentropy at least $2m$. Namely we would like that (X_1, X_2) are indistinguishable from (Y_1, Y_2). However, to prove this one must have the ability to sample from X and Y. See [VZ12] for additional discussion. Lastly, following [VZ12], Definition 2.15 allows the distribution Z to depend on the adversary A.

We will use the fact that if (X, Y) are statistically close to $(\tilde{X}, \tilde{Y})$ and Y has high conditional pseudoentropy given X, then also $\tilde{Y}$ has high conditional pseudoentropy given $\tilde{X}$.

Proposition 2.16. *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$, $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$, $\tilde{X} = \{\tilde{X}_\lambda\}_{\lambda\in\mathbb{N}}$, $\tilde{Y} = \{\tilde{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$ and the same for $\tilde{X}_\lambda$ and $\tilde{Y}_\lambda$.*

Assume $\log(|\mathcal{X}_\lambda| \cdot |\mathcal{Y}_\lambda|) = \mathsf{poly}(\lambda)$, $\mathrm{SD}\Big((X_\lambda, Y_\lambda), (\tilde{X}_\lambda, \tilde{Y}_\lambda)\Big) \leq \mathsf{negl}(\lambda)$ and Y has conditional pseudoentropy at least $m = m(\lambda)$ given X. Then also $\tilde{Y}$ has conditional pseudoentropy at least $m = m(\lambda)$ given $\tilde{X}$.

Since the proof of this statement is not completely straightforward, we give it in full here.

Proof. Assume toward a contradiction that $\tilde{Y}$ does not have conditional pseudoentropy at least $m = m(\lambda)$ given $\tilde{X}$. Namely that there exists $\tilde{c} > 0$ and an algorithm A running in time $\lambda^{\tilde{c}}$ such that for every sequence of random variables $\tilde{Z} = \{\tilde{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ there exists an infinite index-set $\Lambda \subseteq \mathbb{N}$ such that for every $\lambda \in \Lambda$, if $\mathrm{H}(\tilde{Z}_\lambda|\tilde{X}_\lambda) \geq m(\lambda) - 1/\lambda^{\tilde{c}}$ then

$$\left|\Pr\left[\mathsf{A}^{O_{\tilde{X}'_\lambda,\tilde{Y}'_\lambda,\tilde{Z}'_\lambda}}(1^\lambda, \tilde{X}_\lambda, \tilde{Y}_\lambda) = 1\right] - \Pr\left[\mathsf{A}^{O_{\tilde{X}'_\lambda,\tilde{Y}'_\lambda,\tilde{Z}'_\lambda}}(1^\lambda, \tilde{X}_\lambda, \tilde{Z}_\lambda) = 1\right]\right| > \frac{1}{\lambda^{\tilde{c}}}. \tag{7}$$

We use A to break the conditional pseudoentropy of Y given X.

Fix $c > \tilde{c}$ to be determined by the analysis and let $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of random variables jointly distibuted with X, Y such that $\mathrm{H}(Z_\lambda|X_\lambda) \geq m(\lambda) - 1/\lambda^c$, for large enough $\lambda \in \mathbb{N}$. Let $F_\lambda(x, y)$ be the random process that samples from Z_λ conditioned on $X_\lambda = x$ and $Y_\lambda = y$. Namely $(X_\lambda, Y_\lambda, Z_\lambda) \equiv (X_\lambda, Y_\lambda, F_\lambda(X_\lambda, Y_\lambda))$. Let $\tilde{Z} = \{\tilde{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of random variables jointly distributed with $\tilde{X}, \tilde{Y}$ such that $\tilde{Z}_\lambda \equiv F_\lambda(\tilde{X}_\lambda, \tilde{Y}_\lambda)$. Fix large enough $\lambda \in \mathbb{N}$ (which we omit from the notation). It holds that

$$\mathrm{SD}\Big((X, Y, Z), (\tilde{X}, \tilde{Y}, \tilde{Z})\Big) \leq \mathrm{SD}\Big((X, Y), (\tilde{X}, \tilde{Y})\Big) = \mathsf{negl}(\lambda),$$

where the inequality follows from data-processing inequality for statistical distance (on the random process that takes (x, y) and outputs $(x, y, F(x, y))$).

Using the above, we argue that the entropy of $\tilde{Z}|\tilde{X}$ is high. Using Facts 2.6 and 2.8 it holds that

$$\begin{aligned}
\left|\mathrm{H}(\tilde{Z}|\tilde{X}) - \mathrm{H}(Z|X)\right| &\leq \left|\mathrm{H}(\tilde{X}, \tilde{Z}) - \mathrm{H}(X, Z)\right| + \left|\mathrm{H}(\tilde{X}) - \mathrm{H}(X)\right| \\
&\leq \log(|\mathcal{X}| \cdot |\mathcal{Y}|) \cdot \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) + \log(|\mathcal{X}|) \cdot \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) \\
&= \mathsf{negl}(\lambda),
\end{aligned}$$

where the second inequality follows since $h(p) \leq 2\sqrt{p}$ for every $p \in [0, 1]$ and the equality follows from the assumption on the sizes of $\mathcal{X}$ and $\mathcal{Y}$. It follows that

$$\mathrm{H}(\tilde{Z}|\tilde{X}) \geq m - 1/\lambda^c - \mathsf{negl}(\lambda) \geq m - 1/\lambda^{\tilde{c}}.$$

Hence, Eq. (7) holds with respect to $\tilde{Z}$.

Using again that $\mathrm{SD}\Big((X, Y, Z), (\tilde{X}, \tilde{Y}, \tilde{Z})\Big) = \mathsf{negl}(\lambda)$, and since A runs in polynomial time (and thus can make at most polynomially many queries to its oracle), it holds that

$$\begin{aligned}
&\left|\Pr\left[\mathsf{A}^{O_{X'_\lambda,Y'_\lambda,Z'_\lambda}}(1^\lambda, X_\lambda, Y_\lambda) = 1\right] - \Pr\left[\mathsf{A}^{O_{X'_\lambda,Y'_\lambda,Z'_\lambda}}(1^\lambda, X_\lambda, Z_\lambda) = 1\right]\right| \\
&\geq \left|\Pr\left[\mathsf{A}^{O_{\tilde{X}'_\lambda,\tilde{Y}'_\lambda,\tilde{Z}'_\lambda}}(1^\lambda, \tilde{X}_\lambda, \tilde{Y}_\lambda) = 1\right] - \Pr\left[\mathsf{A}^{O_{\tilde{X}'_\lambda,\tilde{Y}'_\lambda,\tilde{Z}'_\lambda}}(1^\lambda, \tilde{X}_\lambda, \tilde{Z}_\lambda) = 1\right]\right| - \mathsf{poly}(\lambda) \cdot \mathsf{negl}(\lambda) \\
&\geq \frac{1}{\lambda^{\tilde{c}}} - \mathsf{negl}(\lambda) \\
&\geq \frac{1}{\lambda^c}.
\end{aligned}$$

Setting c to be large enough so that A runs in λ^c time, we conclude that Y does not have $(\lambda^c, 1/\lambda^c)$ conditional pseudoentropy at least $m - 1/\lambda^c$ given X, a contradiction to the assumption. □

3 The Assumption and Main Theorem

In this section, we specify our assumption on the existence of laconic zero-knowledge proof-systems (which we will later show to imply public-key encryption). To do so, we first introduce some necessary definitions and notations.

Throughout this section (and beyond), we use $\mathcal{L}$ to denote an NP language with witness relation $\mathcal{R}_{\mathcal{L}}$. We use $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$ to denote probabilistic polynomial-time algorithms that are to be seen as sampling algorithms for YES and NO instances of $\mathcal{L}$. More specifically, the sampler $\mathsf{Y}_{\mathcal{L}}(1^\lambda)$ outputs samples of the form (x, w) such that with all but negligible probability (in λ), it holds that $(x, w) \in \mathcal{R}_{\mathcal{L}}$. We call $\mathsf{Y}_{\mathcal{L}}$ a solved instance generator. On the other hand, $\mathsf{N}_{\mathcal{L}}(1^\lambda)$ outputs samples x such that with all but negligible probability, $x \notin \mathcal{L}$. We shall not rely on the fact that the NO sampler $\mathsf{N}_{\mathcal{L}}$ is an *efficient* algorithm. Still we find it easier to present it as such for symmetry with $\mathsf{Y}_{\mathcal{L}}$ (which must be efficient).

We shall be concerned with properties of the tuple $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ – the language $\mathcal{L}$ equipped with (efficiently sampleable) distributions over its YES and NO instances (where YES instances come with corresponding witnesses). Since the choice of YES and NO distributions is always clear from the context, we often simply refer to the above tuple as the language (although we actually mean the language $\mathcal{L}$ with these specific distributions over its instances). We start by defining what we mean when we say that such a language is *cryptographically hard*.

Definition 3.1 (Cryptographic Hardness)**.** *Let $t = t(\lambda) \in \mathbb{N}$ and $\varepsilon = \varepsilon(\lambda) \in [0, 1]$. The language $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is (t, ε)-*cryptographically hard *if $\mathsf{Y}_{\mathcal{L}}$ is a solved instance generator, and for every probabilistic algorithm A that on input $(1^\lambda, x)$ runs in time $t(\lambda)$ and for all sufficiently large $\lambda \in \mathbb{N}$ it holds that:*

$$\left| \Pr_{(x,\cdot)\leftarrow \mathsf{Y}_{\mathcal{L}}(1^\lambda)} \left[\mathsf{A}(1^\lambda, x) = 1\right] - \Pr_{x\leftarrow \mathsf{N}_{\mathcal{L}}(1^\lambda)} \left[\mathsf{A}(1^\lambda, x) = 1\right] \right| \leq \varepsilon(\lambda).$$

We say that $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is cryptographically hard *if it is $(\lambda^c, 1/\lambda^c)$-hard for every constant $c > 0$.*

Being *cryptographically hard* is a stronger requirement than the usual notion of *average-case hardness* (the latter means that it is hard to distinguish a random YES instance from a random NO instance). Specifically, cryptographic hardness requires both (1) average-case hardness *and* (2) the existence of a solved instance generator (wrt the average-case hard distribution). In particular, the existence of a cryptographically hard language is equivalent to the existence of one-way functions.[22] As noted above, when we say that the language $\mathcal{L}$ is cryptographically hard we are actually implicitly referring to the sampling algorithms $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$.

Next we define honest-verifier statistical zero-knowledge (SZK) arguments, which are similar to statistical honest-verifier zero-knowledge proofs but the soundness condition is only required to hold against malicious provers that run in polynomial-time. We remark that since we will be using the existence of SZK arguments to construct other objects, both the relaxations that we

[22] That YES instances are indistinguishable from NO instances implies that it is hard to compute a witness for a YES instance. Given this, a function that takes coins for $\mathsf{Y}_{\mathcal{L}}$ and outputs the instance (but not the witness) generated by $\mathsf{Y}_{\mathcal{L}}$ is one-way (c.f., [Gol08, Proposition 7.2]). For the other direction, assuming that one-way functions exist implies the existence of a linear-stretch pseudorandom generators (PRG) G [HILL99]. The language that is cryptographically hard contains those strings that are in the range of G. The solved instance generator samples a random string r and outputs $G(r)$ as the input and r as the witness. The corresponding NO distribution is that of a random string in the range of the PRG.

employ (namely requiring only *computational* soundness and *honest verifier* zero knowledge) only strengthen our results.

Below, we use $(\mathbf{P}, \mathbf{V})(1^\lambda, x)$ to refer to the transcript of an execution of an interactive protocol with prover $\mathbf{P}$ and verifier $\mathbf{V}$ on input $(1^\lambda, x)$. We also use $(\mathbf{P}(w), \mathbf{V})(1^\lambda, x)$ to denote a similar execution where the prover is additionally given a witness w as an auxiliary input. In both cases, we sometimes also use the same notation to refer to the result (i.e., verifier's output) of such an execution – the appropriate interpretation will be clear from context.

Definition 3.2 (SZK Arguments)**.** *Let $c = c(\lambda) \in [0,1]$ and $s = s(\lambda) \in [0,1]$. An interactive protocol $(\mathbf{P}, \mathbf{V})$ is an* Honest Verifier SZK Argument with completeness error c and soundness error s *for a language $\mathcal{L} \in \mathsf{NP}$, with witness relation $\mathcal{R}_\mathcal{L}$, if the following properties hold:*

- **Efficiency:** *Both $\mathbf{P}$ and $\mathbf{V}$ are probabilistic polynomial-time algorithms.*
- **Completeness:** *For any $(x, w) \in \mathcal{R}_\mathcal{L}$, and all large enough λ:*

$$\Pr\Big[(\mathbf{P}(w), \mathbf{V})(1^\lambda, x) \text{ accepts}\Big] \geq 1 - c(\lambda),$$

  where the parameter c is called the completeness error*.*

- **Soundness:** *For any* probabilistic polynomial-time *cheating prover $\mathbf{P}^\star$, any $x \notin \mathcal{L}$, and large enough λ:*

$$\Pr\Big[(\mathbf{P}^*, \mathbf{V})(1^\lambda, x) \text{ accepts}\Big] \leq s(\lambda),$$

  where the parameter s is called the soundness error*.*

- **Honest Verifier Statistical Zero Knowledge:** *There is a probabilistic polynomial-time algorithm $\mathbf{S}$ (called the simulator) that when given any $x \in \mathcal{L}$ simulates the transcript of the interactive proof on input x. That is, for any $(x, w) \in \mathcal{R}_\mathcal{L}$ and for all sufficiently large λ:*

$$\mathrm{SD}\Big((\mathbf{P}(w), \mathbf{V})(1^\lambda, x), \mathbf{S}(1^\lambda, x)\Big) \leq \mathsf{negl}(\lambda).$$

Note that our definition only deals with NP languages and requires that the prover is efficient. Typically, when defining an SZK *proof* (rather than argument) this is not done, and the honest prover is allowed to be computationally unbounded. However, this is the natural choice since we focus on *argument* systems (where the soundness requirement is only against malicious provers that are also efficient).

Remark 3.3 (Restricted-view Simulation)**.** *For our main result, it suffices that the simulator only simulates the transcript of the interactive proof and not the random-coins of the verifier. The standard definition of simulation is stronger – it also requires that the simulator output random-coins for the verifier that are consistent with the transcript. Ostrovsky [Ost91] called the weaker notion* restricted-view *simulation, and showed that average-case hard languages with honest-verifier* SZK *proofs with restricted-view simulation (without efficient provers) imply the existence of one-way functions.*

We will be dealing with SZK arguments that have additional properties captured by the next definition. Recall that a *round* in an interactive proof is a pair of messages, the first one (possibly empty) from $\mathbf{V}$ to $\mathbf{P}$, and the next the other way.

Definition 3.4 (Laconism)**.** *Let $q = q(\lambda) \in \mathbb{N}$ and $r = r(\lambda) \in \mathbb{N}$. An interactive protocol $(\mathbf{P}, \mathbf{V})$ is said to be r-round and q-laconic if it has at most $r(\lambda)$ rounds, and each message from $\mathbf{P}$ to $\mathbf{V}$ is at most $q(\lambda)$ bits long when run on any input $(1^\lambda, x)$, for large enough λ.*

We can now state our main assumption as follows.

Assumption 3.5. *There exists a cryptographically hard language $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ for which there is an r-round and q-laconic honest-verifier SZK argument with completeness error c and soundness error s such that:*

- *There is a constant $\beta > 0$ such that $1 - c(\lambda) > s(\lambda) + \beta$, for large enough $\lambda \in \mathbb{N}$.*
- *q and r are such that $r^2 \cdot q^3 = O(\log(\lambda))$.*

Our main result is given in the next theorem.

Theorem 3.6 (PKE from Laconic SZK)**.** *If Assumption 3.5 holds, then there exists a public-key encryption scheme.*

In Sections 4 and 5 we show how to use Assumption 3.5 to construct a public-key encryption scheme. The formal proof of Theorem 3.6 is given in the beginning of Section 5. In Section 6, we consider two relaxations of Assumption 3.5 – namely, Assumptions 6.2 and 6.6 – each of which still suffices for our construction of PKE; we then compare other concrete assumptions that have been used in the past to construct public-key encryption to these weaker assumptions.

4 From Laconic SZK to Trapdoor Pseudoentropy Generator

In this section we show that if Assumption 3.5 is true — that is, there exists a cryptographically hard NP language with a laconic (honest-verifier) statistical zero-knowledge argument-system — then there exists a "trapdoor pseudoentropy generator", which we define next. This notion turns out to be a useful technical abstraction and will be used later, in Section 5, to construct a public-key encryption scheme.

Classic pseudoentropy generators, first introduced in the work of [HILL99], are algorithms whose output distribution has pseudoentropy. That is, the output distribution of the algorithm (given a uniformly random seed) is statistically close to a distribution that had high entropy. Pseudoentropy generators play a central role in the construction of pseudorandom generators from (general) one-way functions [HILL99, HRV13, VZ12].

Trapdoor pseudoentropy generators extend the classic notion of pseudoentropy generators to the "public-key settings". Such generators consist of three algorithms. The first algorithm generates a pair of keys, one public and one secret. The second algorithm, which we think of as an encoder, gets the public key as input and generates a distribution which has high pseudoentropy, even to an observer who has the public key but not to one that has the secret-key. Indeed, to a party holding the secret key the output distribution of the encoder "looks" less random. This is captured by the third algorithm, the decoder, that samples elements from a distribution close to that of the encoder - given the secret key. We proceed to the actual definition, that realize the above discussion but for conditional distributions.

Definition 4.1 (Trapdoor Pseudoentropy Generator)**.** *Let λ be a security parameter, $\varepsilon = \varepsilon(\lambda) \in [0,1]$ and $n = n(\lambda) > 0$. An n-*entropic, trapdoor pseudoentropy generator scheme with correctness error *ε is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *where* $\mathsf{KeyGen}(1^\lambda)$ *outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*, the encoding algorithm* $\mathsf{Enc}(1^\lambda, \mathsf{pk})$ *outputs a pair of messages* (u, v) *and the decoding algorithm* $\mathsf{Dec}(1^\lambda, \mathsf{sk}, u)$ *outputs a message v'. The scheme satisfies the following properties with respect to the jointly distributed sequences of random variable $PK = \{PK_\lambda\}_{\lambda\in\mathbb{N}}$, $SK = \{SK_\lambda\}_{\lambda\in\mathbb{N}}$, $U = \{U_\lambda\}_{\lambda\in\mathbb{N}}$, $V = \{V_\lambda\}_{\lambda\in\mathbb{N}}$ and $V' = \{V'_\lambda\}_{\lambda\in\mathbb{N}}$, where $(PK_\lambda, SK_\lambda) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(U_\lambda, V_\lambda) \leftarrow \mathsf{Enc}(1^\lambda, PK_\lambda)$ and $V'_\lambda \leftarrow \mathsf{Dec}(1^\lambda, PK_\lambda, SK_\lambda, U_\lambda)$:*

- **Correctness (of Decoding):** *For all sufficiently large $\lambda \in \mathbb{N}$:*

$$\mathrm{SD}\big((PK_\lambda, SK_\lambda, U_\lambda, V_\lambda), \big(PK_\lambda, SK_\lambda, U_\lambda, V'_\lambda\big)\big) \leq \varepsilon(\lambda).$$

- **Pseudoentropy:** *V has conditional pseudoentropy*[23] *at least $\mathrm{H}(V|PK, U) + n$ given PK, U.*

*An n-entropic trapdoor pseudoentropy generator scheme is q-*laconic*, for $q = q(\lambda) \in \mathbb{N}$ with $q \geq n$, if the output of the decoding algorithm (i.e., v') and the private message of the encoding algorithm are at most $q(\lambda)$-bit long strings.*[24]

We refer to the first message of the encoding algorithm (i.e., u) as its *public* message and to the second message (i.e., v) as its *private* message.

Remark 4.2 (Accessible Entropy)**.** *Another possible way to define the correctness property of a trapdoor pseudoentropy generator is in terms of* accessible entropy *[HRVW09]. Roughly, such a definition would require that for the decoder (who has the secret-key), the entropy of V is much less than for an efficient external eavesdropper (with only the public key). That is, that V has low accessible entropy to the decoder with the secret-key and high pseudoentropy to an efficient external eavesdropper. Such a definition, however, will (slightly) complicate our proof, so we chose to define the correctness in terms of statistical distance.*

The main result of this section is to show that Assumption 3.5 implies the existence of a trapdoor pseudoentropy generator.

Lemma 4.3. *Assume that Assumption 3.5 holds for a language $\mathcal{L}$ with an r-round q-laconic* SZK *argument-system with completeness error c and soundness error s, such that $c, s > 0$ are constants. Then for every $p = \mathsf{poly}(\lambda)$ that is computable in $\mathsf{poly}(\lambda)$ time, there exists a q-laconic, $(\mathrm{KL}(1-c||s)/r)$-entropic trapdoor pseudoentropy generator scheme with correctness error $(1/p)$.*

In Section 5 we show a generic transformation from trapdoor pseudoentropy generators to public-key encryption. Together with Lemma 4.3, this establishes the proof of our main result that laconic SZK argument-system for cryptographically hard NP language implies the existence of public-key encryption (Theorem 3.6).

Section Outline. The rest of this section is devoted to the proof of Lemma 4.3. In Section 4.1, we give the construction of our trapdoor pseudoentropy generator scheme. We also state two lemmas showing, respectively, the correctness and pseudoentropy of the construction. We prove the correctness lemma in Section 4.2 and the pseudoentropy lemma in Section 4.3.

[23] Roughly speaking, a random variable B has pseudoentropy at least m if for every efficient algorithm A there exists a random variable Z with $\mathrm{H}(Z) \geq m$ and A cannot distinguish between B and Z. See Definition 2.15 and the discussion that follows.

[24] The restriction to $q \geq n$ is simply because the entropy is bounded by the length of the string.

4.1 Construction of Trapdoor Pseudoentropy Generator

In this section, we describe our trapdoor pseudoentropy generator. Since the security parameter λ will always be clear from the context, in the following we usually omit it from the notation (e.g., we will refer to the random variable X even though we actually mean X_λ).

Recall that Assumption 3.5 stipulates that there exists an cryptographically hard language $\mathcal{L} \in \mathsf{NP}$ with an r-round q-laconic SZK argument system $(\mathbf{P}, \mathbf{V}, \mathbf{S})$, with completeness error c and soundness error s. We use $(\mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ to denote the sampling algorithms for which the language is cryptographically hard, where $\mathsf{Y}_\mathcal{L}$ is a solved instance generator and $\mathsf{N}_\mathcal{L}$ samples NO instances (with all but negligible probability).

Construction. In addition to the above parameters, our construction also depends on a polynomial $p = \mathsf{poly}(\lambda)$ such that $p(\lambda)$ is computable in $\mathsf{poly}(\lambda)$ time. This parameter will control the correctness error of the scheme.

We begin with an overview of the construction (a formal description follows).

Key Generation: The public key is a yes-instance x. The secret key is w, the corresponding witness for x.

Encoding (x)**:** First, sample a full transcript of interaction using the simulator $\mathbf{S}(x)$. Second, choose a random round $i \leftarrow [r]$ and set b'_i to be the i-th message the prover sent in the transcript and c'^{-} to be the transcript up to b'_i (including the verifier's message in the i-th round and precluding b'_i). Output (i, c'^{-}) as the public message and b'_i as the private message.[25]

Decoding $(w, (i, c'^{-}))$**:** To decode, we use the prover to generate its next message in an interaction consistent with the partial transcript c'^{-}. The decoder generates fresh coin tosses ρ for $\mathbf{P}$ that are consistent with c'^{-}. This is done via rejection sampling. Namely, Dec repeatedly chooses a random string ρ and checks whether given ρ and the verifier's messages in c'^{-}, the prover $\mathbf{P}$ would have sent the prover's messages in c'^{-}. Once it finds such a random string ρ, it computes the next message function of $\mathbf{P}$ given random coins ρ and the transcript c'^{-}. To avoid running for too long, the decoder only runs $2^{r \cdot q} \cdot p$ iterations of the rejection sampling. If it fails to find consistent coins, the decoder outputs an arbitrary q-bit string (e.g., the all-zero string).

The formal description of the trapdoor pseudoentropy generator scheme is given in Fig. 5. Throughout this section we fix all the above parameters and denote

$$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}) = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})_{p, \mathsf{Y}_\mathcal{L}, \mathbf{P}, \mathbf{V}, \mathbf{S}, r, q}.$$

We next state two lemmas which establish the correctness and pseudoentropy properties of the scheme, respectively. To do so, we first introduce random variables corresponding to a random execution of the argument-system for $\mathcal{L}$. As usual, the following sequences of random variables are indexed by $\lambda \in \mathbb{N}$, but for brevity we omit λ from the notations, and also from the arguments that Enc and Dec take. Let (X, W) be an input-witness pair chosen according to the solved instance generator $\mathsf{Y}_\mathcal{L}$. Let A_i (resp., B_i) be the i-th message sent by $\mathbf{V}$ (resp., $\mathbf{P}$) in a random execution

[25] Here and below we use the prime symbol (e.g., b') to hint that a value was generated by the simulator rather than an actual execution.

$\underline{(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})_{p,\mathsf{Y}_\mathcal{L},\mathbf{P},\mathbf{V},\mathbf{S},r,q}}$

Parameters: $p = \mathsf{poly}(\lambda)$.
Algorithms:

- $\mathsf{Y}_\mathcal{L}$: Solved instance generator for $\mathcal{L} \in \mathsf{NP}$.
- $(\mathbf{P}, \mathbf{V}, \mathbf{S})$: $r(\lambda)$-round $q(\lambda)$-laconic SZK argument system for $\mathcal{L}$

$\underline{\mathsf{KeyGen}(1^\lambda)}$

1. Sample $(x, w) \leftarrow \mathsf{Y}_\mathcal{L}(1^\lambda)$
2. Set $\mathsf{pk} = x$ and $\mathsf{sk} = w$
3. Output $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{Enc}(1^\lambda, \mathsf{pk})}$

1. Interpret $\mathsf{pk} = x$
2. Sample $(a'_1, b'_1, \ldots, a'_r, b'_r) \leftarrow \mathbf{S}(1^\lambda, x)$
3. Sample $i \leftarrow [r]$
4. Set $c'^- = (a'_1, b'_1, \ldots, b'_{i-1}, a'_i)$
5. Set $u = (i, c'^-)$ and $v = b'_i$
6. Output (u, v)

$\underline{\mathsf{Dec}(1^\lambda, \mathsf{pk}, \mathsf{sk}, u)}$

1. Interpret $\mathsf{pk} = x$, $\mathsf{sk} = w$ and $u = (i, c'^-)$, where $c'^- = (a'_1, b'_1, \ldots, a'_{i-1}, b'_{i-1}, a'_i)$
2. Repeat for $2 \cdot 2^{r(\lambda) \cdot q(\lambda)} \cdot p(\lambda)$ times
   (a) Sample ρ, coins for $\mathbf{P}$, at random
   (b) For every $j \in [i]$, set $b_j = \mathbf{P}\big(1^\lambda, x, w, (a'_1, b_1, \ldots, a'_{j-1}, b_{j-1}, a'_j)\ ;\ \rho\big)$
   (c) If $(b_1, \ldots, b_{i-1}) = (b'_1, \ldots, b'_{i-1})$, then set $v' = b_i$ and abort the loop
3. If v' was not set until now, set it to $\perp$
4. Output v'

Figure 5: Trapdoor Pseudoentropy Generator from Laconic Zero-Knowledge

of the protocol on input X (where $\mathbf{P}$ also gets access to W). We denote the transcript of the protocol $(\mathbf{P}, \mathbf{V})$ by $(\mathbf{P}(W), \mathbf{V})(X) = (A_1, B_1, A_2, B_2 \ldots A_r, B_r)$, and the transcript generated by the simulator as $\mathbf{S}(X) = (A'_1, B'_1, \ldots, A'_r, B'_r)$.

For a round $i \in [r]$, we denote the partial transcript up to (and including) round i by $C_i = (A_1, B_1, \ldots, A_i, B_i)$. We use C_i^- to denote a random variable that is identical to C_i, except it is missing the i-th message of the prover (but includes the i's message of the verifier), namely $C_i^- = (A_1, B_1, \ldots, B_{i-1}, A_i)$. The partial transcripts C'_i and C'^-_i are similarly defined with respect to the simulated transcript $(A'_1, B'_1, \ldots, A'_r, B'_r)$. We also let $I \leftarrow [r]$ be a uniformly distributed integer in $[r]$, which corresponds to a random round. Finally, we denote by $\tilde{B}'_I$ the output of $\mathsf{Dec}(X, W, I, C'^-)$.

Lemma 4.4 (correctness)**.** *For large enough $\lambda \in \mathbb{N}$, it holds that*

$$\mathrm{SD}\Big((X, W, I, C'^{-}_I, B'_I), (X, W, I, C'^{-}_I, \tilde{B}'_I)\Big) \leq \frac{1}{p(\lambda)}.$$

Lemma 4.5 (pseudoentropy)**.** *The random variable B'_I has conditional pseudoentropy at least $\mathrm{H}(B'_I|X, I, C'^{-}_I) + \frac{1}{r} \cdot \mathrm{KL}(1-c||s)$ given (X, I, C'^{-}_I).*

Lemmas 4.4 and 4.5 are proven in Sections 4.2 and 4.3, respectively. Together, they immediately yield Lemma 4.3.

***Proof of Lemma 4.3** (given Lemmas 4.4 and 4.5).* By Lemmas 4.4 and 4.5, the correctness error of the trapdoor pseudoentropy generator scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is $(1/p)$ and the scheme is $\big(\frac{1}{r} \cdot \mathrm{KL}(1-c||s)\big)$-entropic. Moreover, it is easy to verify that the running times of KeyGen, Enc and Dec are all polynomials in λ (since the prover is efficient given the witness) and that the scheme is q-laconic. □

4.2 Correctness — Proving Lemma 4.4

The proof of Lemma 4.4 follows from the statistical zero-knowledge property of the argument-system and from the following analysis which shows that the distribution generated by Dec is close to that generated by the prover's next message.

Let $\tilde{B}_I$ be the output of $\mathsf{Dec}(X, W, I, C^{-}_I)$; that is, $\tilde{B}_I$ is similar to $\tilde{B}'_I$, except that Dec gets as input a partial transcript sampled from the original protocol rather than a simulated one

Claim 4.5.1. *It holds that:*

$$\mathrm{SD}\Big((X, W, I, C^{-}_I, B_I), (X, W, I, C^{-}_I, \tilde{B}_I)\Big) \leq \frac{1}{2p(\lambda)}.$$

Before proving Claim 4.5.1, let us use it to prove Lemma 4.4.

***Proof of Lemma 4.4** (given Claim 4.5.1).* The following holds for large enough $\lambda \in \mathbb{N}$. By the zero-knowledge property of the argument system it holds that,[26]

$$\begin{aligned}
\mathrm{SD}\big((X, W, I, C'^{-}_I, B'_I), (X, W, I, C^{-}_I, B_I)\big) &= \mathrm{SD}\big((X, W, I, C'_I), (X, W, I, C_I)\big) \\
&\leq \mathrm{SD}\big((X, W, C'_r), (X, W, C_r)\big) \\
&= \mathrm{SD}((X, W, (\mathbf{P}(W), \mathbf{V})(X)), (X, W, \mathbf{S}(X))) \\
&= \mathop{\mathrm{E}}_{(x,w)\leftarrow \mathsf{Y}_{\mathcal{L}}} [\mathrm{SD}((\mathbf{P}(w), \mathbf{V})(x), \mathbf{S}(x))] \\
&= \mathsf{negl}(\lambda),
\end{aligned} \tag{8}$$

where the inequality follows from the data-processing inequality for statistical distance on the randomized procedure that takes (x, w, c_r) (i.e., an instance, witness and a complete transcript), chooses $i \leftarrow [r]$ at random and outputs (x, w, i, c_i) (i.e., the partial transcript of c_r up to round i). Moreover, note that $\tilde{B}_I$ and $\tilde{B}'_I$ can be viewed as being generated by applying the *same* randomized

[26] Note that we only need the zero-knowledge to hold on average for a random instance-witness pair; see Section 6.1 for more details.

function on (X, W, I, C_I^-) and $(X, W, I, C_I'^-)$, respectively. Hence, the data-processing inequality for statistical distance yields that

$$\mathrm{SD}\Big((X, W, I, C_I^-, \tilde{B}_I), (X, W, I, C_I'^-, \tilde{B}'_I)\Big) \leq \mathrm{SD}\big((X, W, I, C_I^-), (X, W, I, C_I'^-)\big).$$

Using once again the zero-knowledge property of the argument-system and similar calculations to Eq. (8), it holds that

$$\mathrm{SD}\Big((X, W, I, C_I^-, \tilde{B}_I), (X, W, I, C_I'^-, \tilde{B}'_I)\Big) \leq \mathsf{negl}(\lambda). \tag{9}$$

Putting Eqs. (8) and (9) together with Claim 4.5.1 and using the triangle inequality for statistical distance, we have that

$$\mathrm{SD}\Big((X, W, I, C_I'^-, B'_I), (X, W, I, C_I'^-, \tilde{B}'_I)\Big) \leq \frac{1}{2p(\lambda)} + \mathsf{negl}(\lambda) \leq \frac{1}{p(\lambda)},$$

as required. □

Remark 4.6 (On the need for statistical zero-knowledge)**.** *The proof of Lemma 4.4 is in fact the only place we use that the argument-system is* statistical *zero-knowledge (rather than merely* computational *zero-knowledge). Specifically, we need that the simulator's output is indistinguishable from the protocol's transcript even to an eavesdropper who knows a witness for the input. This is not necessarily the case for computational zero-knowledge proof systems, but holds for statistical ones.*

It is left to prove Claim 4.5.1.

Proof of Claim 4.5.1. We say that Dec *fails* if it sets v' to be $\perp$ in Step 3. Assume, for sake of the analysis, that Dec never fails but merely keep running the loop again and again. That is, Dec keeps entering the loop in Step 2 until it finds random coins ρ that are consistent with the partial transcript it was given. In such case, by the principle of deferred decision, it holds that $(X, W, I, C_I^-, \tilde{B}_I)$ is identically distributed as (X, W, I, C_I^-, B_I). Hence, when considering the actual algorithm Dec, it holds that

$$\mathrm{SD}\Big((X, W, I, C_I^-, B_I), (X, W, I, C_I^-, \tilde{B}_I)\Big) \leq \Pr\big[\mathsf{Dec}(X, W, I, C_I^-) \text{ fails}\big].$$

In the rest of the proof we bound the probability that Dec fails.

Fix an instance-witness pair (x, w) and a round $i \in [r]$. We bound the failure probability with respect to these fixed values, and the claim follows. Let c be the partial transcript of the interaction thus far, including the i-th message sent by the verifier but not that sent by the prover (in this claim we only care about such transcripts, so for ease of notation, we omit the "$-$" symbol from the superscript of c^-). Let $c_{\mathbf{V}}$ be the part of the transcript corresponding to messages sent by the verifier and let $c_{\mathbf{P}}$ be the part of the transcript corresponding to messages sent by the prover. Since the proof-system is q-laconic, it holds that $c_{\mathbf{P}} \in \{0,1\}^{(i-1)\cdot q}$.

Let $S(c_{\mathbf{P}}, c_{\mathbf{V}})$ be a random variable counting the number of iterations of the loop until finding these random coins. Let $C = (C_{\mathbf{P}}, C_{\mathbf{V}})$ be a partial transcript of a random execution of $(\mathbf{P}(w), \mathbf{V})(x)$. Our goal is to show that

$$\mathrm{E}[S(C_{\mathbf{P}}, C_{\mathbf{V}})] \leq 2^{r \cdot q}. \tag{10}$$

Having shown the above equation, the claim follows by Markov's inequality and since Dec only fails if the number of iterations is larger than $2p \cdot 2^{r \cdot q}$ times.

Fix a transcript $c_{\mathbf{V}}$ of messages from the verifier to the prover up to round i and let $Q_{c_{\mathbf{P}}} = \Pr[C_{\mathbf{P}} = c_{\mathbf{P}} \mid C_{\mathbf{V}} = c_{\mathbf{V}}]$, for every $c_{\mathbf{P}}$. That is, $(Q_{c_{\mathbf{P}}})_{c_{\mathbf{P}} \in \{0,1\}^{(i-1) \cdot q}}$ is the distribution of $C_{\mathbf{P}} | (C_{\mathbf{V}} = c_{\mathbf{V}})$, a random variable corresponding to the transcript of messages sent by the prover in a random execution of the protocol in which the verifier's transcript is fixed to $c_{\mathbf{V}}$.

Claim 4.6.1. *In each iteration of the* Dec *algorithm (i.e., Step 2 of* Dec*), given a partial transcript $c = (c_{\mathbf{P}}, c_{\mathbf{V}})$ as input, the algorithm succeeds in finding consistent random coins with probability $Q_{c_{\mathbf{P}}}$.*

Proof. The proof follows from the fact that conditioned on an execution of the protocol generating a given transcript, the two parties' coins are a product distribution. Details follow.

Let $c = (c_{\mathbf{P}}, c_{\mathbf{V}})$ be a partial transcript up to round i and consider the following set

$$\mathcal{R}_c = \{(\rho_{\mathbf{P}}, \rho_{\mathbf{V}}) \colon (\mathbf{P}(w; \rho_{\mathbf{P}}), \mathbf{V}(\rho_{\mathbf{V}}))(x)_{1,\ldots,i} = c\}.$$

Namely, $\mathcal{R}_c$ is the set of coins for the parties that are consistent with the partial transcript c. Note that if $(\rho_1, \rho_2) \in \mathcal{R}_c$ and $(\rho_1', \rho_2') \in \mathcal{R}_c$, then it must be that $(\rho_1, \rho_2') \in \mathcal{R}_c$. Thus, it holds that $\mathcal{R}_c = \mathcal{R}_{c,\mathbf{P}} \times \mathcal{R}_{c,\mathbf{V}}$, where $\mathcal{R}_{c,\mathbf{P}}$ is a subset of $\mathcal{R}_{\mathbf{P}}$ — all possible coins for $\mathbf{P}$, and $\mathcal{R}_{c,\mathbf{V}}$ is a subset of $\mathcal{R}_{\mathbf{V}}$ — all possible coins for $\mathbf{V}$. It follows that

$$Q_{c_{\mathbf{P}}} = \Pr[C_{\mathbf{P}} = c_{\mathbf{P}} \mid C_{\mathbf{V}} = c_{\mathbf{V}}] = \Pr_{\rho \leftarrow \mathcal{R}_{\mathbf{P}}}[\rho \in \mathcal{R}_{c,\mathbf{P}}].$$

Finally, by construction, Dec samples $\rho \leftarrow \mathcal{R}_{\mathbf{P}}$ and checks if $\mathbf{P}(x, w; \rho)$ generates the messages $c_{\mathbf{P}}$, when given $c_{\mathbf{V}}$ as $\mathbf{V}$'s messages. Such randomness ρ satisfies the latter condition if and only if $\rho \in \mathcal{R}_{c,\mathbf{P}}$. Hence, the probability of success in each iteration is exactly $\Pr_{\rho \leftarrow \mathcal{R}_{\mathbf{P}}}[\rho \in \mathcal{R}_{c,\mathbf{P}}]$. □

Claim 4.6.1 implies that $S(c_{\mathbf{P}}, c_{\mathbf{V}})$ is a geometric random variable with parameter $Q_{c_{\mathbf{P}}}$. The expected value of such a random variable is $1/Q_{c_{\mathbf{P}}}$. It follows that

$$\begin{aligned}
\mathrm{E}[S(C_{\mathbf{P}}, C_{\mathbf{V}}) \mid C_{\mathbf{V}} = c_{\mathbf{V}}] &= \sum_{c_{\mathbf{P}} \in \mathrm{Supp}(C_{\mathbf{P}} | C_{\mathbf{V}} = c_{\mathbf{V}})} Q_{c_{\mathbf{P}}} \cdot \mathrm{E}[S(c_{\mathbf{P}}, c_{\mathbf{V}})] \\
&= \sum_{c_{\mathbf{P}} \in \mathrm{Supp}(C_{\mathbf{P}} | C_{\mathbf{V}} = c_{\mathbf{V}})} Q_{c_{\mathbf{P}}} \cdot \frac{1}{Q_{c_{\mathbf{P}}}} \\
&= |\mathrm{Supp}(C_{\mathbf{P}} \mid C_{\mathbf{V}} = c_{\mathbf{V}})| \\
&\leq 2^{r \cdot q}.
\end{aligned}$$

Finally, it holds that

$$\mathrm{E}[S(C_{\mathbf{P}}, C_{\mathbf{V}})] = \mathrm{E}_{c_{\mathbf{V}} \leftarrow C_{\mathbf{V}}}[\mathrm{E}[S(C_{\mathbf{P}}, C_{\mathbf{V}}) \mid C_{\mathbf{V}} = c_{\mathbf{V}}]] \leq \mathrm{E}_{c_{\mathbf{V}} \leftarrow C_{\mathbf{V}}}[2^{r \cdot q}] = 2^{r \cdot q}.$$

The claim follows. □

4.3 Pseudoentropy — Proving Lemma 4.5

We will establish the pseudoentropy of the scheme in two steps. First, we will show that the prover's next message function in the argument-system is unpredictable. This follows from the fact that if were predictable in the case that $x \in \mathcal{L}$, then either it is similarly predictable in the case that $x \notin \mathcal{L}$ (in which case we break soundness) or we get a distinguisher between the case that $x \in \mathcal{L}$ or $x \notin \mathcal{L}$ (thereby breaking the cryptographic hardness of $\mathcal{L}$).

Our second step is to show that unpredictability of the prover's messages implies pseudoentropy. We show this using the framework of Vadhan and Zheng [VZ12], which we review next. After this review, we go back, in Section 4.3.2, to proving Lemma 4.5.

4.3.1 Unpredictablility and Pseudoentropy

In order to show the pseudoentropy of our construction, we will leverage certain known connections between unpredictability and pseudorandomness. To that end, we use the framework of Vadhan and Zheng [VZ12] who consider a joint distribution (Z, B) and define the unpredictability of B given Z as the inability of any efficient algorithm to generate a sample from a distribution that is close to $B|Z$ in the KL-divergence measure.

Definition 4.7 (KL-hard for sampling [VZ12, Definition 3.5])**.** *Let $t = t(\lambda) \in \mathbb{N}$ and $\varepsilon = \varepsilon(\lambda) \in [0,1]$. Let $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$ and $B = \{B_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that Z_λ and B_λ are jointly distributed for every $\lambda \in \mathbb{N}$. We say that B is (t,ε)-*KL-hard for sampling given *Z if for every oracle-aided probabilistic sampling algorithm D that on input $(1^\lambda, \cdot)$ runs in time $t(\lambda)$, for large enough $\lambda \in \mathbb{N}$, it holds that*

$$\mathrm{KL}\Big((Z_\lambda, B_\lambda)\Big|\Big|\Big(Z_\lambda, \mathsf{D}^{O_{Z'_\lambda, B'_\lambda}}(1^\lambda, Z_\lambda)\Big)\Big) > \varepsilon(\lambda),$$

where $O_{Z'_\lambda, B'_\lambda}$ denotes an oracle that gives a random sample from (Z'_λ, B'_λ) when queried, where (Z'_λ, B'_λ) are identically distributed as (Z_λ, B_λ).

For the necessity of the sampling oracle (and its notation) see the discussion following Definition 2.15. The work of Vadhan and Zheng [VZ12] shows a tight characterization of pseudoentropy in terms of the foregoing notion of unpredictability (i.e., KL-hard for sampling). We only need one of the directions of their theorem, which we state below.

Theorem 4.8 (KL-hardness implies conditional pseudoentropy [VZ12, Lemma 3.7 and Theorem 3.11])**.** *Let $t = t(\lambda) \in \mathbb{N}$, $\varepsilon = \varepsilon(\lambda) \in (0,1]$, $\gamma = \gamma(\lambda) \in (0,1]$, $Q = Q(\lambda) \in \mathbb{N}$ and $p = p(\lambda) \in \mathbb{N}$ such that $p = \mathsf{poly}(\lambda)$, and such that all the above parameters are computable in time $\mathsf{poly}(\lambda)$. Let (Z_λ, B_λ) be jointly distributed $\{0,1\}^{p(\lambda)} \times [Q(\lambda)]$-valued random variables and let $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$ and $B = \{B_\lambda\}_{\lambda\in\mathbb{N}}$. If B is (t,γ)-KL hard for sampling given Z, then B has (t',ε) conditional pseudoentropy at least $\mathrm{H}(B|Z) + \gamma - \varepsilon$, for $t' = t^{\Omega(1)}/\mathsf{poly}(\lambda, Q, 1/\varepsilon)$.*

Note that this theorem is applicable when Q, the support size of B is rather small. In particular, we will only use this theorem for Q which is polynomially bounded. Vadhan and Zheng [VZ12] used Theorem 4.8 to show that if $f\colon \{0,1\}^n \to \{0,1\}^n$ is one-way, then U_n has $O(\log n)$ bits of pseudoentropy given $f(U_n)$. We will use Theorem 4.8 to show that the prover's messages in a laconic SZK argument, with soundness s, has roughly $O(\log(1/s))$ bits of pseudoentropy.[27]

[27] The above holds when the argument-system has perfect completeness. In case completeness is imperfect the amount of pseudoentropy will be slightly different.

4.3.2 Back to the Proof of Lemma 4.5

We need to show that the simulated prover's next message has high pseudoentropy given a random partial simulated transcript. The proof takes the following steps:

1. In Lemma 4.9 we show that for a random round, the prover's message in the argument-system is "KL-unpredictable", given the instance and the transcript thus far. Namely, that B_I is KL-hard for sampling given X, I, C_I^- (according to Definition 4.7);

2. In Lemma 4.10 we use Vadhan and Zheng's [VZ12] framework (via Theorem 4.8) to show that B_I has high *pseudo entropy* given X, I, C_I^-;

3. Finally, using the zero-knowledge property of the argument-system, we argue that B'_I (i.e., the next message produced by the *simulator*) has high *pseudo entropy* given $X, I, C_I'^-$.

The proofs of the first two steps are inspired by [VZ12, Section 4], where — as we mentioned above — it is shown that even given a random output of a one-way function, a random input (that the function maps to the given output) still has (relatively) high pseudoentropy. Interestingly, these steps can be interpretated as an application of [VZ12]'s result to Ostrovsky's [Ost91] one-way function.[28]

We proceed to accomplish the first step in the above outline. (In order to be more precise, in the next lemma we assume specific parameters for the cryptographic hardness of the language $\mathcal{L}$.)

Lemma 4.9. *Let $c = c(\lambda) \in [0,1]$, $s = s(\lambda) \in (0,1]$ and $\gamma = \gamma(\lambda) \in [0,1]$. Let $r = r(\lambda) \in \mathbb{N}$, and let $t = t(\lambda) \in \mathbb{N}$ be polynomially bounded. Assume that*

1. *$\mathcal{L}$ is a (t,γ)-cryptographically hard language;*

2. *$(\mathbf{P}, \mathbf{V})$ is an r-round interactive argument system for $\mathcal{L}$ with completeness error c, soundness error s; and*

3. *$1 - c \geq s + \gamma$, for all sufficiently large values of the security parameter $\lambda \in \mathbb{N}$.*

Then, there is a polynomial p such that for the function $t'(\lambda) = t(\lambda)/p(\lambda)$, the distribution B_I is $(t', \frac{1}{r} \cdot \mathrm{KL}(1-c||s+\gamma))$-KL-hard for sampling given (X, I, C_I^-).

Proof. Assume toward a contradiction that for any polynomial p and $t'(\lambda) = t(\lambda)/p(\lambda)$, the distribution B_I is not $(t', \mathrm{KL}(1-c||s+\gamma)/r)$-KL-hard for sampling given (I, X, C_I^-).[29] Recall that we use O_{I',X',C'_I} to denote an oracle that generates a random round $I' \in [r]$, a random yes instance $X' \leftarrow \mathsf{Y}_{\mathcal{L}}(1^\lambda)$ and a random transcript $C'_{I'}$ of the argument system up to round I' with respect to the instance X' (namely, $(I', X', C'_{I'})$ are identically distributed as (I, X, C_I)).

Fix some polynomial p, which will be specified below. Our assumption implies that there exists an infinite set of indices $\Lambda \subseteq \mathbb{N}$ and an oracle-aided algorithm D that on input $(1^\lambda, i, x, c)$ runs in time $t'(\lambda) = t(\lambda)/p(\lambda)$ such that for every $\lambda \in \Lambda$, it holds that:

$$\mathrm{KL}\Big((I, X, C_I^-), B_I \Big|\Big| (I, X, C_I^-), \mathsf{D}^{O_{I',X',C'_{I'}}}(1^\lambda, I, X, C_I^-)\Big) \leq \frac{1}{r} \cdot \mathrm{KL}(1 - c(\lambda)||s(\lambda) + \gamma(\lambda)). \quad (11)$$

[28] Recall that Ostrovsky [Ost91]'s one-way function maps an instance x, randomness ρ for the simulator and a round i to the transcript up to round i generated by $\mathbf{S}(x;\rho)$.

[29] Observe that we have switched the order of X and I. This minor change in notation is clearly equivalent but slightly more convenient for the current proof.

We use D to construct a distinguisher D' that breaks the cryptographic hardness of $\mathcal{L}$. Roughly speaking, D' executes the interactive argument between the honest-verifier and a prover based on D. The distinguisher D' accepts if and only if the verifier accepts. On YES-instances, Eq. (11) yields that D's outputs are close to the prover's actual responses, and so the verifier accepts with high probability. On the other hand, on NO-instances, D can be viewed as a cheating prover strategy and so, by the soundness of the argument system, the verifier will reject with high probability. Thus, D' distinguishes between YES-instances and NO-instances efficiently. The actual proof below follows this intuition, while taking into account that Eq. (11) only guarantees that D's generates messages that are "close" to the prover's messages for a *random* round I.

Fix a sufficiently large $\lambda \in \Lambda$. To avoid cluttering the notation, in the sequel we omit λ. Consider the following distinguisher for breaking the cryptographic hardness of $\mathcal{L}$.

D' on input x:

1. Sample uniformly random coins for $\mathbf{V}$, denoted by ρ
2. Set c_0 to be the empty string (corresponding to an empty transcript)
3. Repeat for $i = 1$ to r:
   (a) Set $a_i \leftarrow \mathbf{V}(x, c_{i-1}; \rho)$. That is, generate the i-th message a_i of $\mathbf{V}$ given the partial transcript c_{i-1} for rounds $1, \ldots, i-1$ and random coins ρ
   (b) Set $z_i \leftarrow \mathsf{D}^{O_{I',X',C'_{I'}}}(i, x, c_{i-1}, a_i)$
   (c) Set $c_i = (c_{i-1}, a_i, z_i)$
4. Output 1 if $\mathbf{V}(x, c_r; \rho)$ accepts, and 0 otherwise

First, observe that the oracle $O_{I',X',C'_{I'}}$ can be implemented in $\mathsf{poly}(\lambda)$ time — sample $(X', W') \leftarrow \mathsf{Y}_{\mathcal{L}}$ and $I' \leftarrow [r]$, and return X' and the first I' rounds of $(\mathbf{P}(W'), \mathbf{V})(X')$. It follows that there exists a polynomial u, independent of D', such that D' can be implemented in $t'(\lambda) \cdot u(\lambda)$ time. Set $p = u$. Hence, D' runs in time $t(\lambda)$. Next, we show that D' can distinguish between instances that belong to $\mathcal{L}$ from ones that do not with gap at least γ. We first consider NO-instances.

Claim 4.9.1 (D' on NO-instances)**.** *It holds that*

$$\Pr\left[\mathsf{D}'\left(\widetilde{X}\right) = 1\right] \leq s, \tag{12}$$

where $\widetilde{X} \leftarrow \mathsf{N}_{\mathcal{L}}$ *and the probability is also over the randomness of* D'.

Proof. Observe that $\mathsf{D}'\left(\widetilde{X}\right)$ emulates an interaction between the honest-verifier of the argument system and a "cheating prover" (as specified by D) on a random NO-instance. Recall that the running time of D' is $t(\lambda)$ which was assumed to be polynomial. The claim follows from the computational soundness of $\mathbf{V}$.[30] □

Next we consider YES-instances.

Claim 4.9.2 (D' on YES-instances)**.** *It holds that*

$$\Pr\left[\mathsf{D}'(X) = 1\right] \geq s + \gamma,$$

[30] In fact, the very same proof also shows that this claim holds even if the soundness guarantee was only *average-case*. See further discussion in Section 6.1.

where $X \leftarrow \mathsf{Y}_{\mathcal{L}}$ and the probability is also over the randomness of D'.

Proof. Denote by Z_i the message chosen by D in the i'th execution of Step 3b of $\mathsf{D}'(X)$. Analogously to the definitions of C_i and C_i^-, we define $\widetilde{C}_i = (A_1, Z_1, \ldots, A_i, Z_i)$ and $\widetilde{C}_i^- = (A_1, Z_1, \ldots, Z_{i-1}, A_i)$.

We begin by showing that real transcripts (i.e., C_r) look "close" to those generated by D' (i.e., $\widetilde{C}_r$). By the chain rule for divergence (Fact 2.11) it holds that

$$\begin{aligned}
\mathrm{KL}\Big(X, C_r \Big|\Big| X, \widetilde{C}_r\Big) &= \mathrm{KL}(X, A_1, B_1, \ldots, A_r, B_r || X, A_1, Z_1, \ldots, A_r, Z_r) \\
&= \sum_{i=1}^{r} \mathop{\mathrm{E}}_{(x,c_i)\leftarrow(X,C_i^-)} \Big[\mathrm{KL}\Big(B_i|_{X=x,C_i^-=c_i} \Big|\Big| Z_i|_{X=x,\widetilde{C}_i^-=c_i}\Big)\Big] \\
&= \sum_{i=1}^{r} \mathrm{KL}\big(X, C_i^-, B_i \big|\big| X, C_i^-, Z_i\big),
\end{aligned}$$

where we use the chain rule in both of the last two equalities. Another application of the chain rule yields that

$$\begin{aligned}
\sum_{i=1}^{r} \mathrm{KL}\big(X, C_i^-, B_i \big|\big| X, C_i^-, Z_i\big) &= r \cdot \mathrm{KL}\big(I, X, C_I^-, B_I \big|\big| I, X, C_I^-, Z_I\big) \\
&= r \cdot \mathrm{KL}\Big(I, X, C_I^-, B_I \Big|\Big| I, X, C_I^-, \mathsf{D}^{O_{I',X',C'_{I'}}}(I, X, C_I^-)\Big) \\
&\leq \mathrm{KL}(1-c||s+\gamma),
\end{aligned}$$

where the inequality follows from Eq. (11). It follows that

$$\mathrm{KL}\Big(X, C_r \Big|\Big| X, \widetilde{C}_r\Big) \leq \mathrm{KL}(1-c||s+\gamma). \tag{13}$$

Eq. (13) shows that a transcript simulated by $\mathsf{D}'(X)$ is close (in the KL divergence sense) to a transcript of an honest execution of the protocol on X. In the rest of the proof we use the completeness guarantee of the argument system to show that $\mathsf{D}'(X)$ outputs 1 with high probability.

Consider the following (inefficient) random process F:

F on input $(x, c = (a_1, b_1, \ldots, a_r, b_r))$:

1. Sample random coins for $\mathbf{V}$, denoted by ρ, conditioned on $a_i = \mathbf{V}(x, c_{i-1}; \rho)$ for every $i \in [r]$, where $c_i = (a_1, b_1, \ldots, a_i, b_i)$ (abort if no such ρ exist)
2. Output 1 if $\mathbf{V}(x, c; \rho)$ accepts, and 0 otherwise

We emphasize that F is an inefficient process which will only be used for the analysis. By the data-processing inequality for divergence (Fact 2.12) it holds that:

$$\mathrm{KL}\Big(X, C_r \Big|\Big| X, \widetilde{C}_r\Big) \geq \mathrm{KL}\Big(\mathsf{F}(X, C_r) \Big|\Big| \mathsf{F}\Big(X, \widetilde{C}_r\Big)\Big). \tag{14}$$

Define $\delta \coloneqq \Pr[\mathsf{F}(X, C_r) = 1]$ and $\delta' \coloneqq \Pr\Big[\mathsf{F}\Big(X, \widetilde{C}_r\Big) = 1\Big]$. Combining Eqs. (13) and (14) we obtain that

$$\mathrm{KL}\big(\delta \big|\big| \delta'\big) \leq \mathrm{KL}(1-c||s+\gamma).$$

By assumption, $1 - c \geq s + \gamma$, and it is easy to verify that

$$\begin{aligned}\delta &= \Pr[\mathsf{F}(X, C_r) = 1] \\ &= \Pr[\text{out}((\mathbf{P}(W), \mathbf{V})(X)) = \textit{accept}] \\ &= \mathop{\mathrm{E}}_{(x,w)\leftarrow \mathsf{Y}_{\mathcal{L}}}\Big[\Pr[\text{out}((\mathbf{P}(w), \mathbf{V})(x)) = \textit{accept}]\Big] \\ &\geq 1 - c,\end{aligned}$$

where the inequality follows from the completeness condition of $(\mathbf{P}, \mathbf{V})$.[31] Thus, by an application of Fact 2.13 we obtain that $\delta' \geq s + \gamma$.

Finally, by definition of F and the principle of deferred decisions,

$$\Pr\big[\mathsf{D}'(X) = 1\big] = \Pr\Big[\mathsf{F}(X, \widetilde{C}_r) = 1\Big] = \delta' \geq s + \gamma, \tag{15}$$

which concludes the proof of the claim. □

Claims 4.9.1 and 4.9.2 show that D' distinguishes between NO-instances and YES-instances with gap γ. Since D' runs in time $t(\lambda)$, we obtain a contradiction to the (t, γ)-cryptographic hardness of $\mathcal{L}$. This completes the proof of Lemma 4.9. □

Lemma 4.9 shows that B_I is KL-hard to sample given X, I, C_I^-. As our next step, we derive the implication that B_I has additional $\mathrm{KL}(1 - c||s)/r$ bits of pseudoentropy given X, I, C_I^-.

Lemma 4.10. *B_I has conditional pseudoentropy at least $\mathrm{H}(B_I|X, I, C_I^-) + \mathrm{KL}(1 - c||s)/r$ given (X, I, C_I^-).*

Proof. To prove the lemma, we need to show that B_I has $(\lambda^d, 1/\lambda^d)$ conditional pseudoentropy at least

$$\mathrm{H}(B_I|X, I, C_I^-) + \mathrm{KL}(1 - c||s)/r - 1/\lambda^d$$

given X, I, C_I^-, for every constant $d > 0$.

Fix a constant $d > 0$, let $\gamma = 1/\lambda^{c_1}$ for some sufficiently large constant $c_1 > 0$ such that the following two equations hold: $1 - c \geq s + \gamma$, and $1/\lambda^d \geq 2(1 - c) \cdot \gamma/(s \cdot r)$. Since by assumption $c, s > 0$ are constants and $r = \mathsf{poly}(\lambda)$, such c_1 exists.

Since $\mathcal{L}$ is cryptographically hard, in particular, it is also $(\lambda^{c_1}, 1/\lambda^{c_1})$-cryptographically hard. Thus, Lemma 4.9 yields that B_I is $(\lambda^{c_1 - c_2}, \mathrm{KL}(1 - c||s + \gamma)/r)$-KL-hard for sampling given X, I, C_I, for some constant $c_2 > 0$. Furthermore, since the argument system for $\mathcal{L}$ is $O(\log \lambda)$-laconic, it follows that B_I is $[\lambda^{c_3}]$-valued random variable (i.e., it has a support of size λ^{c_3}) for some constant $c_3 > 0$. Theorem 4.8 now yields that B_I has $(t', 1/\lambda^d - 2(1 - c) \cdot \gamma/s)$ conditional pseudoentropy at least

$$\mathrm{H}(B_I|X, I, C_I) + \frac{\mathrm{KL}(1 - c||s + \gamma)}{r} - \Big(1/\lambda^d - 2(1 - c) \cdot \gamma/(s \cdot r)\Big)$$

given X, I, C_I, for $t'(\lambda) = \lambda^{(c_1 - c_2)/c_4 - c_5}$ and some constants $c_4, c_5 > 0$.

[31] Similarly to the case in Claim 4.9.1, this holds even if the completeness guarantee was only *average-case*. See Section 6.1.

Note that c_2, c_4 and c_5 are independent of c_1. Thus, we can set c_1 such that $c_1 \geq (d+c_5)\cdot c_4 + c_2$. It follows that $t'(\lambda) \geq \lambda^d$, for every $\lambda \in \mathbb{N}$. Finally, Fact 2.14 yields that $\mathrm{KL}(1-c||s+\gamma) + 2(1-c)\cdot\gamma/s \geq \mathrm{KL}(1-c||s)$. We conclude that B_I has $(\lambda^d, 1/\lambda^d)$ conditional pseudoentropy at least $\mathrm{H}(B_I|X,I,C_I) + \mathrm{KL}(1-c||s)/r - 1/\lambda^d$ given X, I, C_I, as required. □

We are now ready to formally prove Lemma 4.5.

Proof of Lemma 4.5. Immediately follows from Lemma 4.10, Proposition 2.16 and from the zero-knowledge property of the argument-system. □

5 From Trapdoor Pseudoentropy Generator to Public-Key Encryption

In this section we show a general transformation from laconic trapdoor pseudoentropy generator to public-key encryption. Combined with the results of Section 4 we obtain our main result.

Lemma 5.1. *Let $\lambda \in \mathbb{N}$ be a security parameter and let $\gamma = \gamma(\lambda) \in [0,1]$, $q = q(\lambda) \in \mathbb{N}$ and $n = n(\lambda) > 0$ with $n(\lambda) \in (0, q(\lambda)]$. Assume that there exists a q-laconic n-entropic trapdoor pseudoentropy generator scheme with correctness error γ such that, $q^3/n^2 = O(\log(\lambda))$ and $\gamma = o(n^2/q^2)$. Then, there exists a public-key encryption scheme.*

Lemma 5.1 together with the main result of Section 4 (Lemma 4.3) immediately yield our main theorem.

***Proof of Theorem 3.6** (given Lemma 5.1).* Assume Assumption 3.5 holds for a language $\mathcal{L}$ with an r-round q-laconic SZK argument-system with completeness error c, soundness error s such that $r^2 \cdot q^3 = O(\log(\lambda))$. Further assume, without loss of generality, that $c, s > 0$ are constants (otherwise, c or s are vanishing, and we can set them to be any small constant we like).

Set $p(\lambda) = \lambda$. By Lemma 4.3, there exists a q-laconic n-entropic trapdoor pseudoentropy generator scheme with correctness error $1/\lambda$, for $n = (\mathrm{KL}(1-c||s)/r)$. Since we assumed that $r^2 \cdot q^3 = O(\log(\lambda))$ and that c and s are constants, it holds that

$$\frac{q^3}{n^2} = \frac{q^3 \cdot r^2}{\mathrm{KL}(1-c||s)^2} = O(\log(\lambda)) \qquad \text{and} \qquad \frac{n^2}{q^2} = \frac{\mathrm{KL}(1-c||s)^2}{q^2 \cdot r^2} = \omega(1/\lambda).$$

Thus, Lemma 5.1 yields the existence of public-key encryption. □

The rest of the section is dedicated to proving Lemma 5.1. We first only establish the existence of a *weak* public-key encryption scheme — an encryption scheme whose correctness and security errors are constants rather than negligible. To obtain a full-fledged (i.e., semantically-secure) scheme we rely on a general amplification result due to Holenstein and Renner [HR05] (given in Theorem 2.2).

Section Outline. In Section 5.1 we present the technical tools used in our construction. In particular, we define the notion of *typical set* and that of *smooth min-entropy*. In Section 5.2, we give the construction of our weak public-key encryption scheme. We also state two lemmas showing, respectively, the security and correctness of the construction. We prove the correctness lemma in Section 5.3 and the security lemma in Section 5.4. Actually, the construction in Section 5.2 assumes

we have access to (an approximation of) some statistical property of the underlying hard language. In Section 5.5 we get rid of this assumption by showing an algorithm to estimate the required statistical property. Finally, in Section 5.6, we formally prove Lemma 5.1.

5.1 Technical Tools

In this section we present the technical tools used to prove Lemma 5.1.

5.1.1 Typical Sets and Flattening Distributions

At the heart of our construction lies the notion of a *typical set* for repeated samples of a random variable. Loosely speaking, for a k-fold product repetition of a random variable B, the typical set contains the values whose probability mass is close to $2^{-k\cdot \mathrm{H}(B)}$.

Definition 5.2 (Typical Set)**.** *Let B be a random variable over $\mathcal{B}$, let $k \in \mathbb{N}$ and $\delta \in [0, \log(|\mathcal{B}|)]$. The δ-*typical set of B^k *is defined as*

$$\mathcal{T}^{\delta}_{B^k} = \left\{ \mathbf{b} \in \mathcal{B}^k \colon \Pr[B^k = \mathbf{b}] \in \left[2^{-k\cdot(\mathrm{H}(B)+\delta)}, 2^{-k\cdot(\mathrm{H}(B)-\delta)} \right] \right\}.$$

We will care about the typical set of the k-fold product repetition of a random variable B, jointly distributed with an additional random variable Z.

Definition 5.3 (Conditional Typical Set)**.** *Let (Z, B) be a joint distribution over $\mathcal{Z} \times \mathcal{B}$ and let (Z^k, B^k) be its k-fold product distribution (for some $k \in \mathbb{N}$). For a fixed $\mathbf{z} \in \mathrm{Supp}(Z^k)$ and $\delta > 0$, the δ-*typical set of $B^k|\mathbf{z}$ *is defined as*

$$\mathcal{T}^{\delta}_{B^k|\mathbf{z}} = \left\{ \mathbf{b} \in \mathrm{Supp}(B^k) \ : \ \Pr[B^k = \mathbf{b} | Z^k = \mathbf{z}] \in \left[2^{-k\cdot(\mathrm{H}(B|Z)+\delta)}, 2^{-k\cdot(\mathrm{H}(B|Z)-\delta)} \right] \right\}$$

We emphasize that the range of values that $\Pr[B^k = \mathbf{b}|Z^k = \mathbf{z}]$ is allowed to have (to be in the typical set) depends on $\mathrm{H}(B|Z)$ (rather than $\mathrm{H}(B|Z = \mathbf{z})$).

By definition, the typical set is close to "uniform" — the probability mass of each element is close to $2^{-k\cdot \mathrm{H}(B|Z)}$. In addition, since $\Pr[B^k = \mathbf{b}|Z^k = \mathbf{z}] \geq 2^{-k\cdot(\mathrm{H}(B|Z)+\delta)}$ for every $\mathbf{b} \in \mathcal{T}^{\delta}_{B^k|\mathbf{z}}$, it holds that the typical set is relatively small: $\left|\mathcal{T}^{\delta}_{B^k|\mathbf{z}}\right| \leq 2^{k\cdot(\mathrm{H}(B|Z)+\delta)}$.

An extremely useful property of the typical set of a k-fold product repetition is that most of the probability mass actually lies inside the typical set, as shown by the following lemma:

Lemma 5.4 ([HR11, Theorem 2])**.** *Let (Z, B) be a joint distribution over $\mathcal{Z} \times \mathcal{B}$ and let (Z^k, B^k) be its k-fold product distribution (for some $k \in \mathbb{N}$). For any $\delta \in [0, \log(|\mathcal{B}|)]$ it holds that*

$$\Pr_{(\mathbf{z},\mathbf{b}) \leftarrow (Z^k, B^k)} \left[\mathbf{b} \notin \mathcal{T}^{\delta}_{B^k|\mathbf{z}} \right] < 2 \cdot 2^{-\frac{k\cdot\delta^2}{2\log^2(|\mathcal{B}|+3)}}.$$

Loosely speaking, this means that for sufficiently large k, the k-fold product repetition of a random variable B looks like a uniform distribution over a set of size $2^{k\cdot \mathrm{H}(B)}$. This result is commonly referred to in the literature on SZK as the "flattening" of a distribution (e.g., [GV99]), and in the literature of information theory as the "Asymptotic Equipartition Property".

5.1.2 Smooth Min-Entropy

We next introduce the notion of *smooth conditional min entropy*. Loosely speaking this notion means that the conditional distribution is statistically close to a distribution with high min-entropy. We formalize this notion below.

Definition 5.5 (ε-Smooth Conditional Min-Entropy [RW05])**.** *Let X and Y be (jointly) distributed random variables over $\mathcal{X} \times \mathcal{Y}$ and let $\varepsilon > 0$. The ε-*smooth min-entropy *of X given Y is defined as*

$$\mathrm{H}^{\varepsilon}_{\infty}(X|Y) = \max_{E} \min_{y \in \mathrm{Supp}(Y)} \min_{x \in \mathcal{X}} \log\left(\frac{1}{\Pr[X = x \wedge E | Y = y]}\right),$$

where the maximum ranges over all events E with $\Pr[E] \geq 1 - \varepsilon$.

The distribution close to X with a high min-entropy is the distribution X conditioned on the event E. Next, we state a result due to Holenstein and Renner [HR11] which shows that the k-fold repetition of a distribution (X, Y) has smooth conditional min-entropy close to the conditional *Shannon entropy* (and not conditional min-entropy).

Theorem 5.6 ([HR11, Theorem 1])**.** *Let (X^k, Y^k) be the k-fold product repetition of (X, Y) over $\mathcal{X}^k \times \mathcal{Y}^k$. Then, for any $\delta \geq 0$*

$$\mathrm{H}^{\varepsilon}_{\infty}(X^k|Y^k) \geq k \cdot \big(\mathrm{H}(X|Y) - \delta\big),$$

where $\varepsilon = 2^{-\frac{k\delta^2}{2\log^2(|\mathcal{X}|+3)}}$.

(Theorem 5.6 is proven via the notion of typical sets and is in fact a corollary of Lemma 5.4.)

In our security proof, we shall use the leftover hash lemma [HILL99]. Below, we state a variant that only needs high ε-smooth min-entropy (rather than standard min-entropy).

Lemma 5.7 (Leftover Hash Lemma for ε-Smooth Min-Entropy (c.f., [RW05, Theorem 1]))**.** *Let $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of universal hash functions. Then, for any jointly distributed random variables X and Y, such that X is distributed over $\{0,1\}^n$, it holds that*

$$\mathrm{SD}\Big(\big(H(X), H, Y\big), \big(U_m, H, Y\big)\Big) \leq \varepsilon + \frac{1}{2} \cdot \sqrt{2^{-\mathrm{H}^{\varepsilon}_{\infty}(X|Y)} \cdot 2^m}\,,$$

where $H \leftarrow \mathcal{H}$ and U_m is distributed uniformly over $\{0,1\}^m$.

For sake of completeness, and since the actual statement of [RW05, Theorem 1] does not directly refer to universal hash functions, we give a full proof of Lemma 5.7 in Appendix B.1.

We also need the computational variant of ε-smooth min-entropy, which is a natural extension of conditional pseudoentropy (Definition 2.15).

Definition 5.8 (Pseudo ε-smooth min-entropy)**.** *Let $t = t(\lambda) \in \mathbb{N}$, $\gamma = \gamma(\lambda) \in [0,1]$, $\varepsilon = \varepsilon(\lambda) \in [0,1]$ and $m = m(\lambda) \in \mathbb{R}_{\geq 0}$. Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$, for every $\lambda \in \mathbb{N}$. We say that X has (t, γ)* conditional pseudo ε-smooth min-entropy at least m given Y *if for every probabilistic algorithm A that on input $(1^\lambda, x, y)$ runs in time $t(\lambda)$, there are sequences of random variables $\{Z_\lambda\}_{\lambda \in \mathbb{N}}$ over $\mathcal{X}_\lambda$, jointly distributed with X_λ, Y_λ, such that the following hold for large enough $\lambda \in \mathbb{N}$:*

1. $\mathrm{H}^{\varepsilon}_{\infty}(Z_\lambda|Y_\lambda) \geq m(\lambda)$;

2. *It holds that*

$$\left|\Pr\left[\mathsf{A}\left(1^\lambda, X_\lambda, Y_\lambda\right) = 1\right] - \Pr\left[\mathsf{A}\left(1^\lambda, Z_\lambda, Y_\lambda\right) = 1\right]\right| \leq \gamma(\lambda),$$

where the above probabilities are over $X_\lambda, Y_\lambda, Z_\lambda$ and the random coins of A.

We say that X has conditional pseudo ε-smooth min-entropy at least m given Y *if for every constant $c > 0$, the random variable X has $(\lambda^c, 1/\lambda^c)$ conditional pseudo ε-smooth min-entropy at least $m - 1/\lambda^c$ given Y.*

The next lemma shows how conditional pseudo-entropy can be transformed into conditional pseudo ε-smooth *min-entropy* by repetition. This is a fairly standard technique in the literature of constructing pseudorandom generators from any one-way functions (e.g, [HILL99, HRV13, VZ12]), and enables the use of the Leftover Hash Lemma in our application.

Lemma 5.9. *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$, for every $\lambda \in \mathbb{N}$. Assume X has conditional pseudo entropy at least $\mathrm{H}(X|Y) + n$ given Y, for $n = n(\lambda) \geq 0$.*

Then, for any $\delta = \delta(\lambda) \in [0, \log(|\mathcal{X}_\lambda|)]$ and every polynomial $k = k(\lambda)$, the sequence X^k has conditional pseudo ε-smooth min-entropy at least $k \cdot (\mathrm{H}(X|Y) + n - \delta)$ given Y^k, for $\varepsilon = 2^{-\frac{k\cdot\delta^2}{2\log^2(|\mathcal{X}|+3)}}$, where (X^k, Y^k) are the k-fold product repetition of (X, Y).

Since previous statements transforming pseudo-entropy into a computational notion of min-entropy did not explicitly refer to the notion of ε-smooth min-entropy, we give a full proof of Lemma 5.9 in Appendix B.2.

5.2 Construction of Weak PKE

In this section, we describe our weak public-key encryption scheme. As usual, since the security parameter λ will always be clear from the context, in the following we omit it from the notation.

Assume that there exists a q-laconic n-entropic trapdoor pseudoentropy generator scheme $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ with correctness error γ.

We first introduce (jointly distributed) random variables corresponding to a random execution of the trapdoor pseudoentropy generator scheme. Let (PK, SK) be a public-key secret-key pair chosen by a random execution of $\mathsf{KeyGen}'(1^\lambda)$. Let (U, V) be the messages returned by a random execution of $\mathsf{Enc}'(PK)$. Finally, let V' be the message returned by a random execution of $\mathsf{Dec}'(PK, SK, U)$.

As we previously mentioned, our weak public-key encryption scheme requires some access to some statistical property of the pseudoentropy generator scheme. We will later show how to efficiently estimate this property (in Section 5.5). Specifically, we require access to (an approximation of) the entropy of the decoder's message given the public key, secret key and the encoder's public message.

Definition 5.10. *Let $\xi = \xi(\lambda) \geq 0$ and let $\mathsf{Ent}(1^\lambda)$ be an algorithm whose output is always in $[0, q(\lambda)]$. We say that* Ent ξ-approximates the entropy of Dec''s message *if*

$$\left|\mathsf{Ent}(1^\lambda) - \mathrm{H}\left(V'\middle|PK, SK, U\right)\right| \leq \xi(\lambda),$$

for large enough $\lambda \in \mathbb{N}$.

For now we will simply assume such an estimator Ent exists. Later, in Section 5.5, we will show how to efficiently implement Ent (by strongly relying on the laconism of the scheme).

Construction of Weak PKE. Our construction of a weak PKE depends on two parameters, $k = k(\lambda) \in \mathbb{N}$ and $\delta = \delta(\lambda) \in [0, q]$. The parameter δ, which corresponds to the slackness in the definition of the typical set, will depend on the amount of pseudoentropy the encoder's private message has (i.e., n bits). The parameter k, which corresponds to the number of repetitions that we need of the underlying trapdoor pseudoentropy generator scheme, will depend on δ and q. Let Ent be an estimation algorithm defined above (see Definition 5.10), where the approximation factor is $\xi = \delta/4$. We begin with an overview of the construction (a formal description follows).

Keys: The public key is a vector of public keys of the underlying pseudoentropy generator scheme $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ and a number $\ell = \mathsf{Ent}(1^\lambda)$. The secret key is a vector $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_k)$ of the corresponding secret-keys of the underlying pseudoentropy generator scheme.

Encryption of bit σ: For every $j \in [k]$ sample $(u_j, v_j) \leftarrow \mathsf{Enc}'(\mathsf{pk}_j)$. Set $\mathbf{u} = (u_1, \ldots, u_k)$ and $\mathbf{v} = (v_1, \ldots, v_k)$. Next, sample a universal hash function $h \colon \{0,1\}^{k \cdot q} \to \{0,1\}^{k \cdot (\ell + \delta)}$. The ciphertext consists of $\mathbf{u}$, h, $h(\mathbf{v})$, a random bit-vector $\mathbf{s}$ and a mask of the bit σ with $\langle \mathbf{s}, \mathbf{v} \rangle$ (the latter is the inner product of $\mathbf{v}$ with $\mathbf{s}$), which we denote by $\sigma' = \sigma \oplus \langle \mathbf{s}, \mathbf{v} \rangle$.

Decryption of $(\mathbf{u}, h, h(\mathbf{v}), \mathbf{s}, \sigma')$: Repeatedly sample $\mathbf{v}' = (v'_1, \ldots, v'_k)$, where $v'_i \leftarrow \mathsf{Dec}'(\mathsf{pk}_i, sk_i, u_i)$, until $h(\mathbf{v}') = h(\mathbf{v})$ (namely, do rejection sampling to find the first vector $\mathbf{v}'$ that matches the hash value given). If no such $\mathbf{v}'$ is found, abort. Otherwise, output $\sigma' \oplus \langle \mathbf{s}, \mathbf{v}' \rangle$.

The formal description of the weak public-key encryption scheme is given in Fig. 6. Throughout this section we fix all the parameters as discussed above and denote

$$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}) = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})_{\delta, k, q, \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Ent}}.$$

Below we state the correctness and security lemmas, establishing these properties for our encryption scheme. The proofs of these lemmas are given in Sections 5.3 and 5.4, respectively. It is not yet clear that the current construction is efficient since it requires access to the entropy approximator algorithm (which we have not yet implemented). In Section 5.5 we show how to efficiently implement this approximator and in Section 5.6 we use this implementation with the following lemmas to argue that our construction is a public-key encryption scheme.

Lemma 5.11 (weak correctness)**.** *If $k \cdot \delta^2 \geq 21000 \cdot q^2$ and $k \cdot \gamma \leq 2^{-6}$, then for sufficiently large $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\Big[\mathsf{Dec}\Big(1^\lambda, \mathsf{sk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\Big) = \sigma\Big] > 1 - 2^{-5},$$

where the probability is over $\sigma \leftarrow \{0,1\}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and the randomness of Dec and Enc.

At a (very) high level, to prove the above lemma we show that the decryption algorithm manages to find $\mathbf{v}$, the value the encryption algorithm used to mask the encrypted bit, with (constant) high probability. To do so we rely on the correctness property of the underlying trapdoor pseudoentropy generator.

Lemma 5.12 (weak security)**.** *If $k \cdot (n - 3\delta) \geq 14$, $k \cdot \delta^2 \geq 140q^2$ and $16(q+2)^2 \cdot \gamma \leq 9\delta^2$, then for every polynomial time adversary A and sufficiently large $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\Big[\mathsf{A}\Big(1^\lambda, \mathsf{pk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\Big) = \sigma\Big] \leq \frac{1}{2} + 2^{-5},$$

where the probability is over $\sigma \leftarrow \{0,1\}$, $(\mathsf{pk}, \cdot) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and the randomness of Enc and A.

At a high level, to prove the above lemma we use the pseudoentropy of the underlying trapdoor pseudoentropy generator scheme to argue (together with the Leftover Hash Lemma) that the encrypted bit looks random to an adversary that does not have the secret key.

By these two lemmas, a decryptor who has the secret-key can decrypt with probability that is close to 1, whereas an adversary, that only has the public key, can only decrypt with probability that is close to 1/2.

5.3 Correctness — Proving Lemma 5.11

We need to show that with high probability over the randomness of KeyGen, Enc and Dec, the decryption algorithm Dec returns the same value that was encrypted.

Fix large enough $\lambda \in \mathbb{N}$ and $\sigma \in \{0,1\}$ and let $\delta = \delta(\lambda)$, $k = k(\lambda)$, $q = q(\lambda)$ and $\gamma = \gamma(\lambda)$. Let $L \leftarrow \mathsf{Ent}(1^\lambda)$, $M = \lfloor k \cdot (L + \delta) \rfloor$, $H \leftarrow \mathcal{H}_{k \cdot q, M}$ and $\mathbf{S} \leftarrow \{0,1\}^{k \cdot q}$. By construction, our goal is to show that

$$\Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V^k \rangle\big) = \sigma\right] \geq 1 - 2^{-5}, \tag{16}$$

where recall that (PK^k, SK^k, U^k, V^k) are the k-fold product repetition of $(PK, SK) \leftarrow \mathsf{KeyGen}'(1^\lambda)$ and $(U, V) \leftarrow \mathsf{Enc}'(PK)$. Also recall that $V' \leftarrow \mathsf{Dec}'(PK, SK, U)$ is a jointly distributed random variable. Finally, recall that the public-key for our encryption scheme is the pair (PK^k, L) (and thus L appear next to PK^k in the above expression).

Our first step is to replace V^k with V'^k in the above equation. Since the correctness error of the underlying trapdoor pseudoentropy generator is at most γ and by the triangle inequality for statistical distance, it holds that

$$\begin{aligned}
\mathrm{SD}\Big(\big(PK^k, SK^k, U^k, V^k\big), \big(PK^k, SK^k, U^k, V'^k\big)\Big) &\leq k \cdot \mathrm{SD}\big((PK, SK, U, V), \big(PK, SK, U, V'\big)\big) \\
&\leq k \cdot \gamma \\
&\leq 2^{-6},
\end{aligned}$$

where the last inequality follows from the assumption of the lemma. Combined with Eq. (16) we obtain that

$$\begin{aligned}
&\Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V^k \rangle\big) = \sigma\right] \\
&\geq \Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V'^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V'^k \rangle\big) = \sigma\right] - 2^{-6}.
\end{aligned} \tag{17}$$

Eq. (17) allows us to think, for sake of the analysis, that the encryption algorithm Enc, rather than sampling using Enc' alone, samples the public message using Enc'— namely $(u, \cdot) \leftarrow \mathsf{Enc}'(\mathsf{pk})$, and then the private message using Dec' and the secret-key (which it does not have) — namely $v' \leftarrow \mathsf{Dec}'(\mathsf{pk}, \mathsf{sk}, u)$.

$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})_{\delta,k,q,\mathsf{KeyGen}',\mathsf{Enc}',\mathsf{Dec}',\mathsf{Ent},\mathsf{Hist}}$

Parameters: $\delta = \delta(\lambda) \in [0, q(\lambda)]$, $k = k(\lambda) \in \mathbb{N}$ with $k = \mathsf{poly}(\lambda)$, with $k(\lambda) \cdot \delta(\lambda) \geq 32$.
Algorithms:

- $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$: $q(\lambda)$-laconic $n(\lambda)$-entropic trapdoor pseudoentropy generator scheme with correctness error $\gamma(\lambda)$
- Ent: $(\delta/4)$-approximator[a] for the entropy of V' given PK, SK, U

$\mathsf{KeyGen}(1^\lambda)$

1. For $j \in [k]$, sample $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}'(1^\lambda)$
2. Set $\ell = \mathsf{Ent}(1^\lambda)$
3. Set $\mathsf{pk} = ((\mathsf{pk}_1, \ldots, \mathsf{pk}_k), \ell)$ and $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_k)$
4. Output $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)$

1. Compute $\delta = \delta(\lambda)$, $k = k(\lambda)$ and $q = q(\lambda)$
2. Interpret $\mathsf{pk} = ((\mathsf{pk}_1, \ldots, \mathsf{pk}_k), \ell)$
3. For $j \in [k]$, sample $(u_j, v_j) \leftarrow \mathsf{Enc}'(1^\lambda, \mathsf{pk}_j)$
4. Set $\mathbf{u} = (u_1, \ldots, u_k)$ and $\mathbf{v} = (v_1, \ldots, v_k)$
5. Sample $h \leftarrow \mathcal{H}_{k \cdot q, m}$ and $\mathbf{s} \leftarrow \{0,1\}^{k \cdot q}$, for $m = \lfloor k \cdot (\ell + \delta) \rfloor$
6. Output $\mathsf{ct} = \big(\mathbf{u}, h, h(\mathbf{v}), \mathbf{s}, \sigma \oplus \langle \mathbf{s}, \mathbf{v} \rangle\big)$

$\mathsf{Dec}(1^\lambda, \mathsf{pk}, \mathsf{sk}, \mathsf{ct})$

1. Compute $\delta = \delta(\lambda)$, $k = k(\lambda)$ and $q = q(\lambda)$
2. Interpret $\mathsf{pk} = ((\mathsf{pk}_1, \ldots, \mathsf{pk}_k), \ell)$, $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_k)$ and $\mathsf{ct} = \big((u_1, \ldots, u_k), h, z, \mathbf{s}, \sigma'\big)$
3. Repeat $2^{10 \cdot k \cdot (\ell + \delta/2)}$ many times:
   (a) Compute $\mathbf{v} = (v_1, \ldots, v_k) \in \{0,1\}^{k \cdot q}$ where $v_i \leftarrow \mathsf{Dec}'(\mathsf{pk}_i, \mathsf{sk}_i, u_i)$ for all i.
   (b) If $h(\mathbf{v}) = z$, then set $\mathbf{v}' = \mathbf{v}$ and exit the loop.
4. If $\mathbf{v}'$ was not set, output $\perp$. Otherwise, Output $\sigma' \oplus \langle \mathbf{s}, \mathbf{v}' \rangle$.

[a] See Definition 5.10. Recall that (PK, SK) are a pair of random keys chosen by KeyGen', U is a random public message chosen by $\mathsf{Enc}'(PK)$ and V' is a random output of $\mathsf{Dec}'(PK, SK, U)$.

Figure 6: Public-Key Encryption from Trapdoor Pseudoentropy Generator

The rest of the proof proceeds as follows: First, we show that the message $\mathbf{v}'$ sampled by the encryption algorithm (according to our new understanding of the encryption algorithm) is "typical", namely its probability mass is not too small. Put differently, $\mathbf{v}'$ belongs to the typical set. Second, we show that h is injective over the typical set. This holds by our choice for the output length of h and since the typical set cannot be too large. Third, we show that the rejection sampling done in Step 3 of Dec – if successful (that is, exits the loop via Step 3b) – returns $\mathbf{v}'$ with high probability. Indeed, since h is injective over the typical set, every $\mathbf{v}''$ with $h(\mathbf{v}'') = h(\mathbf{v}')$ cannot belong to the typical set, namely, must be sampled with only low probability. Finally, we argue that since there exists a typical element that hashes to $h(\mathbf{v}')$ – namely $\mathbf{v}'$ itself – the rejection sampling succeeds with high probability.

Our first step is to show that with high probability $\mathbf{v}'$ belongs to the typical set. This is established in the following claim:

Claim 5.12.1 (w.h.p. $\mathbf{v}'$ belongs to the typical set)**.**

$$\Pr_{(\mathsf{pk},\mathsf{sk},\mathbf{u},\mathbf{v}')\leftarrow(PK^k,SK^k,U^k,V'^k)}\left[\mathbf{v}' \notin \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\right] \leq 2^{-\frac{k\cdot\delta^2}{200q^2}}$$

Proof. Applying Lemma 5.4 with $Z = (PK, SK, U)$, $B = V'$, k and $\delta/4$ yields that

$$\Pr_{(\mathsf{pk},\mathsf{sk},\mathbf{u},\mathbf{v}')\leftarrow(PK^k,SK^k,U^k,V'^k)}\left[\mathbf{v}' \notin \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\right] \leq 2\cdot 2^{-\frac{k(\delta/4)^2}{2\log^2(2^q+3)}}.$$

The claim follows since $\log^2(2^q+3) \leq 10\cdot q^2$ for $q \geq 1$ and since we assumed (in the statement of Lemma 5.11) that $1 \leq k\cdot\delta^2/(3000q^2)$. □

Next we show that with high probability there is a unique value inside the typical set that hashes to a given value.

Claim 5.12.2 (only $\mathbf{v}'$ hashes to $h(\mathbf{v}')$)**.** *Assume* $\mathbf{v}' \in \mathrm{Supp}(V'^k)$, $\ell \in \mathrm{Supp}(L)$ *and let* $m = \lfloor k\cdot(\ell+\delta)\rfloor$. *Then, it holds that*

$$\Pr_{h\leftarrow\mathcal{H}_{k\cdot q,m}}\left[\exists\ \mathbf{v} \in \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}} \text{ such that } \mathbf{v}\neq\mathbf{v}' \text{ and } h(\mathbf{v}) = h(\mathbf{v}')\right] < 2^{-k\cdot\delta/4}.$$

Proof. Note that the size of the typical set is small. Specifically, it holds that

$$\begin{aligned}\left|\mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\right| &\leq 2^{k\cdot(\mathrm{H}(V'|PK,SK,U)+\delta/4)}\\ &\leq 2^{k\cdot(\ell+\delta/2)},\end{aligned} \tag{18}$$

where the first inequality follows since every element in the typical set has probability mass at least $2^{-k\cdot(\mathrm{H}(V'|PK,SK,U)+\delta/4)}$ and the second follows from the assumption that the Ent algorithm $(\delta/4)$-approximates the entropy $\mathrm{H}(V'|PK,SK,U)$.

Next, we use the properties of universal hashing to derive the claim. Intuitively, since our hash output is $k\cdot(\ell+\delta)$ bits long, the size of the output domain is larger than that of the typical set by a factor of at least $2^{k\cdot(\delta/2)}$. We can now apply a union bound to prove an upper bound on the collision probability. Formally, let $\mathbf{v}\neq\mathbf{v}'$ with $\mathbf{v} \in \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}$. Since h is chosen from a family of

universal hash functions, it holds that $\Pr_{h\leftarrow\mathcal{H}_{k\cdot q,m}}[h(\mathbf{v})=h(\mathbf{v}')]=\frac{1}{2^{\lfloor k\cdot(\ell+\delta)\rfloor}}$. Applying the union bound over the typical set yields that:

$$\begin{aligned}\Pr_{h\leftarrow\mathcal{H}_{k\cdot q,m}}\left[\exists\ \mathbf{v}\in\mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\text{ such that }\mathbf{v}\neq\mathbf{v}'\text{ and }h(\mathbf{v})=h(\mathbf{v}')\right]&\leq\frac{\left|\mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\right|}{2^{\lfloor k\cdot(\ell+\delta)\rfloor}}\\&\leq\frac{2^{k\cdot(\ell+\delta/2)}}{2^{k\cdot(\ell+\delta)-1}}\\&\leq 2^{-k\cdot\delta/2+1}.\end{aligned}$$

The claim now follows since by assumption $1\leq k\cdot\delta/4$. □

The next claim shows that if the rejection sampling succeeds, it returns an element in the typical set with high probability. It does so by showing that the output distribution of the rejection sampling is identical to that of Dec. Let $\mathbf{W}$ be the random variable, jointly distributed with $PK^k, SK^k, U^k, V'^k, L, H$ and $\mathbf{S}$, distributed as the value of $\mathbf{v}'$ at the end of the loop in Step 3 in a random execution of $\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V'^k), \mathbf{S}, \sigma\oplus\langle\mathbf{S}, V'^k\rangle\big)$. Let $\mathsf{RejSampFail}$ be the event that the rejection sampling in Step 3 fails in the same random execution of Dec, i.e., no $\mathbf{v}$ that hashes to the given hash value is found in all the iterations.

Claim 5.12.3 ($\mathbf{v}'$ and $\mathbf{v}$ are identically distributed when rejection sampling succeeds)**.** *Let* $(\mathsf{pk},\mathsf{sk},\mathbf{u})\in\mathrm{Supp}(PK^k, SK^k, U^k)$*. Then,*

$$\big(V'^k|_{PK^k=\mathsf{pk},SK^k=\mathsf{sk},U^k=\mathbf{u}}\big)\equiv\big(\mathbf{W}|_{PK^k=\mathsf{pk},SK^k=\mathsf{sk},U^k=\mathbf{u},\neg\mathsf{RejSampFail}}\big).$$

Proof. The proof of this claim follows from the fact that for any random variable A and any function f taking values in $\mathrm{Supp}(A)$, sampling from A can be done by first sampling $b\leftarrow f(A)$ and then $A|_{f(A)=b}$.

Fix $h\in\mathrm{Supp}(H)$ and let $z\leftarrow h(V'^k)$. Since we conditioned on $\neg\mathsf{RejSampFail}$, the principle of deferred decision and the definition of Dec imply that

$$\big(\mathbf{W}|_{PK^k=\mathsf{pk},SK^k=\mathsf{sk},U^k=\mathbf{u},H=h,\neg\mathsf{RejSampFail}}\big)\equiv\big(V'^k|_{PK^k=\mathsf{pk},SK^k=\mathsf{sk},U^k=\mathbf{u},H=h,h(V'^k)=z}\big).$$

Since the above holds for any fixing of h, the claim follows. □

The final claim before proving Lemma 5.11 show that if $\mathbf{v}'$ belongs to the typical set, then the rejection sampling succeeds.

Claim 5.12.4. *It holds that,*

$$\Pr\left[\mathsf{RejSampFail}\middle|V'^k\in\mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}\right]\leq 2^{-10}.$$

(A note about notation: in the term "$V'^k\in\mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}$" the first V'^k that appears to the left of the "$\in$" symbol and PK^k, SK^k, U^k are all random variables part of the joint distribution over which the probability is taken. However, the second V'^k that appear in the subscript of the typical set is *not* a random variable, but rather a part of our notation of the typical set of the random variable V'^k. Until now we managed to avoid this notation overload by restricting the probability space to only be over the random variables (PK^k, SK^k, U^k, V'^k), for example as we did in Claim 5.12.1. Here and below, it will be more convenient not to restrict the probability space.)

Proof. Fix any $(\mathsf{pk}, \mathsf{sk}, \mathbf{u}, \mathbf{v}') \in \mathrm{Supp}(PK^k, SK^k, U^k, V'^k)$ such that $\mathbf{v}' \in \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}$ and any $(\ell, h, \mathbf{s}) \in \mathrm{Supp}(L, H, \mathbf{S})$. Consider a random execution of Dec with these fixed values. By construction, the probability of a single iteration of the loop in Step 3 succeeding is given by,

$$\Pr_{\mathbf{v}\leftarrow V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\big[h(\mathbf{v}) = h(\mathbf{v}')\big].$$

It holds that,

$$\begin{aligned}\Pr_{\mathbf{v}\leftarrow V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\big[h(\mathbf{v}) = h(\mathbf{v}')\big] &\geq \Pr_{\mathbf{v}\leftarrow V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\big[\mathbf{v} = \mathbf{v}'\big] \\ &\geq 2^{-k(\mathrm{H}(V'|PK,SK,U)+\delta/4)} \\ &\geq 2^{-k(\ell+\delta/2)}.\end{aligned}$$

where the second inequality follows since $\mathbf{v}' \in \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}$ and from the definition of the typical set (Definition 5.3), and the third inequality follows from the assumption that the Ent algorithm $(\delta/4)$-approximates the entropy $\mathrm{H}(V'|PK, SK, U)$.

So, the failure probability of each loop iteration is at most $1 - 2^{-k(\ell+\delta/2)}$. Hence, the probability that all $2^{10\cdot k\cdot(\ell+\delta/2)}$ *independent* iterations fail is at most

$$\Big(1 - 2^{-k(\ell+\delta/2)}\Big)^{(2^{10\cdot k\cdot(\ell+\delta/2)})} \leq 2^{-10},$$

where we used that $(1 - \frac{1}{x})^x < e^{-1} < 2^{-1}$.

The claim now follows since the above failure probability holds for any fixing of $(\mathsf{pk}, \mathsf{sk}, \mathbf{u}, \mathbf{v}')$ such that $\mathbf{v}' \in \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}$ and $(\ell, h, \mathbf{s})$. □

Equipped with Claims 5.12.1 to 5.12.4 we can now complete the proof of Lemma 5.11. We say that the event Coll occurs when $H(V'^k)$ has at least two pre-images inside the typical set; that is, $\Big|\mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \cap H^{-1}(H(V'^k))\Big| \geq 2$. (Here and below we again overload the notation of "V'^k"; see the text following Claim 5.12.4 for a discussion.) By construction it holds that,

$$\Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V'^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V'^k\rangle\big) = \sigma \;\middle|\; \begin{array}{c} V'^k \in \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \wedge \neg\mathsf{RejSampFail} \\ \wedge \mathbf{W} \in \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \wedge \neg\mathsf{Coll} \end{array}\right] = 1.$$

Indeed, $\neg\mathsf{Coll}$ implies that there is only one element in the typical set that hashes to the given value, while $\neg\mathsf{RejSampFail}$ implies that both V'^k and $\mathbf{W}$ hash to that value. Since we also condition on both V'^k and $\mathbf{W}$ being in the typical set, they must be equal. Finally, by construction, that V'^k and $\mathbf{W}$ are equal implies that the decryption algorithm will succeed in decrypting the message correctly.

Hence,

$$
\begin{aligned}
&\Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V'^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V'^k\rangle\big) = \sigma\right] \qquad (19)\\
&\geq \Pr\left[V'^k \in \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \wedge \mathbf{W} \in \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \wedge \neg\mathsf{Coll} \wedge \neg\mathsf{RejSampFail}\right]\\
&\geq 1 - \Pr\left[\mathbf{W} \notin \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}\right] - \Pr[\mathsf{Coll}] - \Pr\left[V'^k \notin \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k} \vee \mathsf{RejSampFail}\right]\\
&\geq 1 - \Pr\left[\mathbf{W} \notin \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}\right] - \Pr[\mathsf{Coll}] - \Pr\left[V'^k \notin \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}\right]\\
&\quad - \Pr\left[\mathsf{RejSampFail}\middle| V'^k \in \mathcal{T}^{\delta/4}_{V'^k|PK^k,SK^k,U^k}\right]\\
&\overset{(*)}{\geq} 1 - 2 \cdot \Pr_{(\mathsf{pk},\mathsf{sk},\mathbf{u},\mathbf{v}')\leftarrow(PK^k,SK^k,U^k,V'^k)}\left[\mathbf{v}' \notin \mathcal{T}^{\delta/4}_{V'^k|\mathsf{pk},\mathsf{sk},\mathbf{u}}\right] - 2^{-k\cdot\delta/4} - 2^{-10}\\
&\overset{(**)}{\geq} 1 - 2^{-\frac{k\cdot\delta^2}{200q^2}+1} - 2^{-8} - 2^{-10}\\
&\geq 1 - 2^{-15+1} - 2^{-8} - 2^{-10}\\
&\geq 1 - 2^{-6},
\end{aligned}
$$

where $(*)$ follows from Claims 5.12.2 to 5.12.4 and $(**)$ follows by the assumptions that $7 \leq k \cdot \delta^2/3000q^2$ and $k \cdot \delta \geq 32$.

Combining Eq. (19) with Eq. (17) implies that Eq. (16) holds. That is,

$$\Pr\left[\mathsf{Dec}\big(PK^k, L, SK^k, U^k, H, H(V^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V^k\rangle\big) = \sigma\right] \geq 1 - 2^{-5}.$$

This concludes the proof of Lemma 5.11.

5.4 Security — Proving Lemma 5.12

The proof relies on the fact that by assumption, V, a random private message of the encoder of the underlying trapdoor pseudoentropy generator, has pseudoentropy given PK and U, a random public key and public message of the encoder of the underlying trapdoor pseudoentropy generator, respectively. By repetition, we can convert this pseudoentropy into pseudo *min-entropy*.

Claim 5.12.5. *The conditional pseudo ε-smooth min-entropy of V^k, given PK^k, U^k is at least $k \cdot \big(\mathrm{H}(V|PK,U) + n - \delta\big)$ where $\varepsilon = 2^{-\frac{k\delta^2}{20q^2}}$.*

Proof. By assumption, V has conditional pseudoentropy at least $\mathrm{H}(V|PK,U) + n$ given PK, U. Lemma 5.9 yields that V^k has conditional pseudo ε'-smooth min-entropy at least $k\cdot(\mathrm{H}(V|PK,U) + n - \delta)$ given PK^k, U^k, for $\varepsilon' = 2^{-\frac{k\delta^2}{2\log^2(2q+3)}}$. Since $2\log^2(2^q+3) \leq 20q^2$ for $q \geq 1$, it holds that $\varepsilon \geq \varepsilon'$. The claim follows since the ε-smooth min-entropy increases with ε (i.e., $\mathrm{H}^{\varepsilon}_{\infty}(Z) \geq \mathrm{H}^{\varepsilon'}_{\infty}(Z)$). □

Using the above claim we turn to prove Lemma 5.12 which shows that our encryption scheme is weakly secure. The proof follows from the leftover hash lemma and the pseudo min-entropy of $V^k|PK^k, U^k$.

Proof of Lemma 5.12. The proof is by a reduction. Given an algorithm A that can decrypt with good probability, we construct a distinguisher $\widehat{\mathsf{A}}$ that distinguishes between V^k and any random variable with high conditional ε-smooth min-entropy. This contradicts Claim 5.12.5.

Fix large enough $\lambda \in \mathbb{N}$ and[32] $\sigma \in \{0,1\}$ and let $\delta = \delta(\lambda)$, $k = k(\lambda)$, $q = q(\lambda)$, $\gamma = \gamma(\lambda)$ and $n = n(\lambda)$. Let $L \leftarrow \mathsf{Ent}(1^\lambda)$, $M = \lfloor k \cdot (L + \delta) \rfloor$, $H \leftarrow \mathcal{H}_{k \cdot q, M}$ and $V \leftarrow \{0,1\}^{k \cdot q}$. Assume toward a contradiction that the statement does not hold. Namely that there exists a polynomial time algorithm A, an infinite index set $\Lambda \subseteq \mathbb{N}$ such that for every $\lambda \in \Lambda$, it holds that

$$\begin{aligned} \Pr[\mathsf{A}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \sigma)) = \sigma] &= \Pr\Big[\mathsf{A}\big(PK^k, L, U^k, H, H(V^k), \mathbf{S}, \sigma \oplus \langle \mathbf{S}, V^k \rangle\big) = \sigma\Big] \\ &\geq \frac{1}{2} + 2^{-5}. \end{aligned} \tag{20}$$

Moreover, by the assumptions of the lemma it holds that $k \cdot (n/2 - 1.5\delta) \geq 7$ and $\frac{k \cdot \delta^2}{20q^2} \geq 7$. Thus

$$2^{-5} \geq 2^{-7} + 2^{-7} + \frac{1}{\lambda^c} \geq 2^{-k \cdot (n/2 - 1.5\delta)} + 2^{-\frac{k \cdot \delta^2}{20q^2}} + \frac{1}{\lambda^c},$$

for large enough $\lambda \in \mathbb{N}$ and some $c > 0$. Thus, for for large enough $\lambda \in \Lambda$, it also holds that

$$\Pr\Big[\mathsf{A}\big(PK^k, L, U^k, H, H(V^k), \mathbf{S}, \sigma \oplus \big\langle \mathbf{S}, V^k \big\rangle\big) = \sigma\Big] \geq \frac{1}{2} + 2^{-k \cdot (n/2 - 1.5\delta)} + 2^{-\frac{k \cdot \delta^2}{20q^2}} + \frac{1}{\lambda^c}. \tag{21}$$

Let $\varepsilon = 2^{-\frac{k \cdot \delta^2}{20q^2}}$. We use A to break the conditional pseudo ε-smooth min-entropy of V^k. Or, in other words, to distinguish between V^k and any random variable with ε-smooth min-entropy at least $k \cdot (\mathrm{H}(V|PK, U) + n - \delta)$. Intuitively, Eq. (21) yields that A can recover the encrypted bit σ with "high" probability when it is masked by V^k (namely, recovering σ from $\sigma \oplus \langle \mathbf{S}, V^k \rangle$ and $\mathbf{S}$). In contrast, we next show that when σ is encrypted using a random variable whose ε-smooth min-entropy at least $k \cdot (\mathrm{H}(V|PK, U) + n - \delta)$, the algorithm A can only recover σ with "small" probability. Thus A breaks the conditional pseudo ε-smooth min-entropy of V^k.

Formally, let $\mathbf{Z} = \{\mathbf{Z}_\lambda\}_{\lambda \in \mathbb{N}}$ be an sequence of random variables over $\{0,1\}^{k \cdot q}$ such that

$$\begin{aligned} \mathrm{H}^\varepsilon_\infty\Big(\mathbf{Z}\Big|PK^k, U^k\Big) &\geq k \cdot (\mathrm{H}(U|PK, U) + n - \delta) - \frac{1}{\lambda^{c'}} \\ &\geq k \cdot (\mathrm{H}(U|PK, U) + n - \delta) - 1, \end{aligned} \tag{22}$$

for large enough $\lambda \in \mathbb{N}$ and for some $c' > 0$ to be determined by the analysis. Fix $\ell \in \mathrm{Supp}(L)$ and let $m = \lfloor k \cdot (\ell + \delta) \rfloor$. By the (generalized) leftover hash lemma (Lemma 5.7), Eq. (22) and since concatenating the inner product to a universal hash function is also a universal hash function[33] it holds that

$$\begin{aligned} &\Pr\Big[\mathsf{A}\Big(PK^k, \ell, U^k, H, \mathbf{S}, H(\mathbf{Z}), \sigma \oplus \langle \mathbf{S}, \mathbf{Z} \rangle\Big) = \sigma\Big] \\ &\leq \Pr\Big[\mathsf{A}\Big(PK^k, \ell, U^k, H, \mathbf{S}, R_m, \sigma \oplus R_1\Big) = \sigma\Big] + \varepsilon + \frac{1}{2} \cdot \sqrt{2^{-k \cdot (\mathrm{H}(V|PK,U) + n - \delta) + 1} \cdot 2^{m+1}}, \end{aligned} \tag{23}$$

where $R_m \leftarrow \{0,1\}^m$ and $R_1 \leftarrow \{0,1\}$.

[32] As in the correctness proof, we prove (the stronger statement) that the correctness of the decryption holds for any enceypted bit $\sigma \in \{0,1\}$, rather than a randomly chosen one.

[33] Namely, the function $h'(x) = (h(x), \langle r, x \rangle)$, where $h \in \mathcal{H}_{qk,m}$ and $r \in \{0,1\}^{qk}$, is a universal hash function.

Recall that by the assumption on Ent, it holds that $\ell \leq \mathrm{H}(V'|PK, SK, U) + \delta/4$. Moreover, since by assumption $\mathrm{SD}((PK, SK, U, V), (PK, SK, U, V')) \leq \gamma$ and the pseudoentropy generator scheme is q-laconic, Fact 2.9 yields that

$$\begin{aligned}\mathrm{H}(V'|PK, SK, U) &\leq \mathrm{H}(V|PK, SK, U) + q \cdot \gamma + h(\gamma) \\ &\leq \mathrm{H}(V|PK, SK, U) + q \cdot \gamma + 2\sqrt{\gamma} \\ &\leq \mathrm{H}(V|PK, SK, U) + 3\delta/4,\end{aligned}$$

where the second inequality follows since $h(\gamma) \leq 2\sqrt{\gamma}$ for every $\gamma \in [0, 1]$ and the third inequality follows from the assumption of the lemma that imply $q \cdot \gamma + 2\sqrt{\gamma} \leq (q + 2)\sqrt{\gamma} \leq 3\delta/4$. It follows that for large enough λ,

$$m \leq k \cdot (\ell + \delta) \leq k \cdot (\mathrm{H}(V'|PK, SK, U) + \delta/4 + \delta) \leq k \cdot (\mathrm{H}(U|PK, SK, U) + 2\delta). \tag{24}$$

Furthermore, observe that

$$\Pr\Big[\mathsf{A}\Big(PK^k, \ell, U^k, H, \mathbf{S}, R_m, \sigma \oplus R_1\Big) = \sigma\Big] = \Pr\Big[\mathsf{A}\Big(PK^k, \ell, U^k, H, \mathbf{S}, R_m, R_1\Big) = \sigma\Big] = \frac{1}{2}. \tag{25}$$

Plugging Eqs. (24) and (25) into Eq. (23) yields that

$$\Pr\Big[\mathsf{A}\Big(PK^k, \ell, U^k, H, \mathbf{S}, H(\mathbf{Z}), \sigma \oplus \langle \mathbf{S}, \mathbf{Z}\rangle\Big) = \sigma\Big] \leq \frac{1}{2} + 2^{-k\cdot(n/2-1.5\delta)} + \varepsilon. \tag{26}$$

Consider the distinguisher $\widehat{\mathsf{A}}$ that on input $(1^\lambda, \mathsf{pk}, \mathbf{u}, \mathbf{v})$ sets $\ell = \mathsf{Ent}(1^\lambda)$ and $m = \lfloor k \cdot (\ell + \delta)\rfloor$, samples $h \leftarrow \mathcal{H}_{k\cdot q,m}$, $\mathbf{s} \leftarrow \{0, 1\}^{k\cdot q}$ and $\sigma \leftarrow \{0, 1\}$, and outputs 1 iff $\mathsf{A}(\mathsf{pk}, \ell, \mathbf{u}, h, \mathbf{s}, h(\mathbf{v}), \sigma \oplus \langle \mathbf{s}, \mathbf{v}\rangle) = \sigma$. By construction, Eqs. (21) and (26) it holds that

$$\Big|\Pr\Big[\widehat{\mathsf{A}}\Big(PK^k, U^k, V^k\Big) = 1\Big] - \Pr\Big[\widehat{\mathsf{A}}\Big(PK^k, U^k, \mathbf{Z}\Big) = 1\Big]\Big| \geq \frac{1}{\lambda^c}, \tag{27}$$

for large enough $\lambda \in \Lambda$.

Finally, since the running times of A and Ent are polynomials, there exists $d > 0$ such that $\widehat{\mathsf{A}}$'s running time is at most λ^d. Set $c' = \max\{c, d\}$. We conclude that V^k does not have $(\lambda^{c'}, 1/\lambda^{c'})$ conditional pseudo ε-smooth min-entropy at least $k \cdot (\mathrm{H}(V|PK, U) + n - \delta)$ given PK^k, U^k, a contradiction to Claim 5.12.5.

Thus, there cannot exist an adversary A as above for the encryption scheme. This concludes the proof of Lemma 5.12. □

5.5 Implementing the Approximation Algorithm Ent

In Sections 5.2 to 5.4 we constructed and proved the correctness and security of a weak PKE scheme, assuming the existence of an algorithm approximating a statistical property of the trapdoor pseudoentropy generator. In this section, we show how to efficiently implement the needed approximation algorithm.

Recall that the required algorithm (see Definition 5.10) is meant to approximate the entropy of the decoder's message in the underlying pseudoentropy generator scheme, given the public key, the secret key and the encoder's public message. We show how to implement such an entropy approximator for general random variables, assuming they can be sampled in a way that satisfy

some efficiency requirements and have sufficiently small support. Naturally, the keys and the messages of the underlying pseudoentropy generator scheme satisfy these requirements.

To describe the algorithm for estimating entropy, first we describe an algorithm that approximates the probability mass function — which we call the *histogram*, of an efficiently samplable distribution. The algorithm will approximate the histogram by repeatedly sampling from the distribution and returning the empirical distribution constructed from these samples. Indeed, the key fact that we will use is that the support size of the distribution is small (namely, polynomial in the security parameter). Note that such an algorithm can be used also to approximate the entropy of the distribution as well. In fact, this is how we implement the algorithm for approximating the entropy.

Lemma 5.13. *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$. Assume that for every $y \in \mathrm{Supp}(Y_\lambda)$, it is possible to sample from the distribution $(X_\lambda|(Y_\lambda = y))$ in $\mathsf{poly}(\lambda)$ time. Finally, let $\varepsilon, \delta\colon \mathbb{N} \to [0,1]$ be such that $\varepsilon = \varepsilon(\lambda)$ and $\delta(\lambda) = \delta$ are computable in $\mathsf{poly}(\lambda)$ time.*

Then, there exists a randomized algorithm $\mathsf{Hist} = \mathsf{Hist}^{(X,Y,\varepsilon,\delta)}$ that when given input of the form $(1^\lambda, y)$, for $y \in \mathrm{Supp}(Y_\lambda)$, runs in time $\mathsf{poly}(\lambda, |\mathcal{X}_\lambda|, 1/\varepsilon(\lambda), \log(1/\delta(\lambda)))$ and returns a vector representing a distribution over $\mathcal{X}_\lambda$ such that for all sufficiently large λ it holds that

$$\Pr_{Q\leftarrow\mathsf{Hist}(1^\lambda,y)}[\mathrm{SD}(Q, (X_\lambda|(Y_\lambda = y))) > \varepsilon(\lambda)] \leq \delta(\lambda).$$

Proof. The algorithm Hist simply returns the empirical distribution of $X_\lambda|(Y_\lambda = y)$.

$\mathsf{Hist}^{(X,Y,\varepsilon,\delta)}$ on input $(1^\lambda, y)$:

1. Set $\varepsilon = \varepsilon(\lambda)$, $\delta = \delta(\lambda)$ and let $\mathcal{X} = \mathcal{X}_\lambda$, $X = X_\lambda$ and $Y = Y_\lambda$
2. For every $x \in \mathcal{X}$, set $P_x = 0$
3. Repeat $N = \left\lceil \frac{|\mathcal{X}|+\log(1/\delta)}{2\varepsilon^2} \right\rceil$ times:
   (a) Sample $x \leftarrow (X|Y = y)$.
   (b) Set $P_x = P_x + 1$.
4. Return the vector $(P_x/N)_{x\in\mathcal{X}}$.

It is easy to verify that the running time of $\mathsf{Hist}^{(X,Y,\varepsilon)}$ is $\mathsf{poly}(\lambda, |\mathcal{X}_\lambda|, 1/\varepsilon(\lambda), \log(1/\delta(\lambda)))$. Indeed, computing $\varepsilon = \varepsilon(\lambda)$ and $\delta(\lambda)$, and sampling from $(X|Y = y)$ can be done in $\mathsf{poly}(\lambda)$ time and the loop has $O\left(\frac{|\mathcal{X}|+\log(1/\delta)}{\varepsilon^2}\right)$ iterations.

The correctness of the algorithm follows immediately from the following fact, showing that computing the empirical distribution from large enough number of samples approximates the original distribution well.

Fact 5.14 (Folklore (see, e.g., [Gol17, Exercise 11.4]))**.** *Let P be a distribution over n elements and let $\hat{P}$ be the empirical distribution obtained from taking N samples $P_1, \ldots, P_N$ from P, namely, $\hat{P}(i) = |\{j\colon P_j = i\}|/N$. Then, if $N \geq \left\lceil \frac{n+\log(1/\delta)}{2\varepsilon^2} \right\rceil$, it holds that*

$$\Pr\left[\mathrm{SD}\left(P, \hat{P}\right) \geq \varepsilon\right] \leq \delta.$$

The following proof was communicating to us by John Wright [Wri17].

Proof. Assume without loss of generality that the distribution P is over the elements $[n] = \{1, 2, \ldots, n\}$. By definition, $\mathrm{SD}\Big(P, \hat{P}\Big) \geq \varepsilon$ if and only if $\exists \mathcal{S} \subseteq [n]$ such that $\hat{P}(\mathcal{S}) - P(\mathcal{S}) \geq \varepsilon$.

Fix $\mathcal{S} \subseteq [n]$. Since $\hat{P}$ is the empirical distribution obtained from taking N samples from P, it easy to see that $\hat{P}(\mathcal{S})$ is identically distributed as $(Z_1 + \cdots + Z_N)/N$, where each Z_i is independent Bernoulli random variable which is 1 with probability $P(\mathcal{S})$ and 0 otherwise. Hence,

$$\Pr\Big[\hat{P}(\mathcal{S}) - P(\mathcal{S}) \geq \varepsilon\Big] = \Pr\left[\frac{1}{N}\sum_{i=1}^{N} Z_i \geq P(\mathcal{S}) + \varepsilon\right] \leq e^{-2N\varepsilon^2} \leq 2^{-2N\varepsilon^2},$$

where the first inequality follows from the Chernoff bound.

Applying the union bound we have that

$$\begin{aligned}
\Pr\Big[\mathrm{SD}\Big(P, \hat{P}\Big) \geq \varepsilon\Big] &= \Pr\Big[\exists \mathcal{S} \subseteq [n] \colon \hat{P}(\mathcal{S}) - P(\mathcal{S}) \geq \varepsilon\Big] \\
&\leq \sum_{\mathcal{S} \subseteq [n]} \Pr\Big[\hat{P}(\mathcal{S}) - P(\mathcal{S}) \geq \varepsilon\Big] \\
&\leq 2^n \cdot 2^{-2N\varepsilon^2} \\
&\leq \delta,
\end{aligned}$$

where the last inequality follows since $N \geq \big\lceil (n + \log(1/\delta))/(2\varepsilon^2) \big\rceil$. □

Applying Fact 5.14 with $n = |\mathcal{X}|$, ε and δ completes the proof of Lemma 5.13. □

We now turn to the main task of estimating the entropy of a random variable X, given another random variable Y; that is, approximating $\mathrm{H}(X|Y)$. Our algorithm will repeatedly call the previous algorithm (Hist) to get approximations for the distribution $X|(Y = y)$, for randomly sampled y's, and compute the entropy with respect to each such y based on these approximations. Since the entropy with respect to each y is bounded by the logarithm of the support size, Hoeffding's inequality tells us that the average of these entropies is concentrated around $\mathrm{H}(X|Y)$. Hence, the average of the above entropies approximates the original entropy well.

Lemma 5.15. *Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$. Assume that it is possible to sample from Y_λ in $\mathsf{poly}(\lambda)$ time, and that for every $y \in \mathrm{Supp}(Y_\lambda)$, it is also possible to sample from the distribution $(X_\lambda|(Y_\lambda = y))$ in $\mathsf{poly}(\lambda)$ time. Finally, let $\varepsilon = \varepsilon(\lambda) > 0$ and $\delta = \delta(\lambda) \in (0, 1]$ be computable in $\mathsf{poly}(\lambda)$ time.*

Then, there exists a randomized algorithm $\mathsf{Ent} = \mathsf{Ent}^{(X,Y,\varepsilon,\delta)}$ that on input 1^λ runs in time $\mathsf{poly}(\lambda, |\mathcal{X}_\lambda|, 1/\varepsilon(\lambda), 1/\delta(\lambda))$ and outputs a number in $[0, \log(|\mathcal{X}_\lambda|)]$ such that the following holds for large enough $\lambda \in \mathbb{N}$.

$$\Pr\Big[\Big|\mathsf{Ent}(1^\lambda) - \mathrm{H}(X_\lambda|Y_\lambda)\Big| > \varepsilon\Big] \leq \delta.$$

Proof. Consider the following algorithm.

$\mathsf{Ent}^{(X,Y,\varepsilon,\delta)}$ on input (1^λ):

1. Set $\varepsilon = \varepsilon(\lambda)$, $\delta = \delta(\lambda)$ and let $\mathcal{X} = \mathcal{X}_\lambda$, $X = X_\lambda$ and $Y = Y_\lambda$

2. Set $\varepsilon' = \left\lfloor \frac{\varepsilon^2}{4(\log(|\mathcal{X}|)+2)^2} \right\rfloor$, $T = \left\lceil \frac{\log(4/\delta)\cdot\log^2(|\mathcal{X}|)}{(\varepsilon/2)^2} \right\rceil$ and $\delta' = \min\{\frac{\delta}{2T}, \varepsilon'\}$
3. For $i = 1$ to T:
   (a) Sample $y_i \leftarrow Y$
   (b) Sample $\hat{P}_i \leftarrow \mathsf{Hist}^{(X,Y,\delta',\delta')}(1^\lambda, y_i)$
   (c) Set $Z_i = \mathrm{H}(\hat{P}) = \sum_{x\in\mathcal{X}} \hat{P}_i(x) \cdot \log(1/\hat{P}_i(x))$
4. Return $\frac{1}{T} \cdot \sum_{i=1}^{T} Z_i$

We first argue correctness and then analyze the running time. Fix a sufficiently large $\lambda \in \mathbb{N}$. In the sequel we omit λ from the notation. We think of Z_i as a random variable over the probability space of choosing $y_i \leftarrow Y$ and the randomness of Hist used to generate $\hat{P}_i$. Let W_i be a random variable coupled with Z_i as follows: W_i takes the value of $\mathrm{H}(X|Y = y_i)$ for y_i being the value sampled at step 3a in the i'th iteration of the loop. It holds that $\mathrm{E}[W_i] = \mathrm{H}(X|Y)$ for every $i \in [T]$. Hence, our goal is to bound

$$\Pr\left[\left|\mathsf{Ent}^{(X,Y,\varepsilon,\delta)}(1^\lambda) - \mathrm{H}(X|Y)\right| > \varepsilon\right] = \Pr\left[\left|\frac{1}{T}\sum_{i=1}^{T} Z_i - \frac{1}{T}\sum_{i=1}^{T}\mathrm{E}[W_i]\right| > \varepsilon\right] = \Pr\left[\left|\bar{Z} - \mathrm{E}\left[\bar{W}\right]\right| > \varepsilon\right],$$

where $\bar{Z} = \frac{1}{T}\sum_{i=1}^{T} Z_i$, and $\bar{W} = \frac{1}{T}\sum_{i=1}^{T} W_i$.

Let P_i the distribution of the random variable $(X|Y = y_i)$, where y_i is again the value sampled at step 3a in the i'th iteration of the loop. It holds that

$$\begin{aligned}\Pr_{\substack{(y_1,\ldots,y_T)\leftarrow Y^T\\ \hat{P}_1,\ldots,\hat{P}_T}}\left[\left|\bar{Z} - \mathrm{E}\left[\bar{W}\right]\right| > \varepsilon\right] \le &\Pr_{\substack{(y_1,\ldots,y_T)\leftarrow Y^T\\ \hat{P}_1,\ldots,\hat{P}_T}}\left[\left|\bar{Z} - \mathrm{E}\left[\bar{W}\right]\right| > \varepsilon \;\middle|\; \forall i\colon \mathrm{SD}\left(P_i, \hat{P}_i\right) \le \delta'\right] \\ &+ \Pr_{\hat{P}_1,\ldots,\hat{P}_T}\left[\exists i\colon \mathrm{SD}\left(P_i, \hat{P}_i\right) > \delta'\right].\end{aligned} \tag{28}$$

By Lemma 5.13 and the union bound it holds that

$$\Pr_{\hat{P}_1,\ldots,\hat{P}_T}\left[\exists i\colon \mathrm{SD}\left(P_i, \hat{P}_i\right) > \delta'\right] \le T \cdot \delta'$$

In order to bound the first term in the right-hand side of Eq. (28), we need to show that if $\hat{P}_i$ and P_i are close, then so are Z_i and W_i. We abuse notation and let P_i and $\hat{P}_i$ denote also random variables chosen according to the distributions P_i and $\hat{P}_i$, respectively. It holds that $Z_i = \mathrm{H}(\hat{P}_i)$ and $W_i = \mathrm{H}(P_i)$. Assume for now that $\forall i\colon \mathrm{SD}\left(P_i, \hat{P}_i\right) \le \delta'$. Using the fact that small statistical distance implies small entropy difference (Fact 2.8), it holds that

$$\begin{aligned}\left|\bar{Z} - \mathrm{E}\left[\bar{W}\right]\right| &= \left|\bar{Z} - \mathrm{E}\left[\bar{W}\right] + \bar{W} - \bar{W}\right| \\ &\le \left|\bar{W} - \mathrm{E}\left[\bar{W}\right]\right| + \frac{1}{T}\sum_{i=1}^{T}|Z_i - W_i| \\ &\le \left|\bar{W} - \mathrm{E}\left[\bar{W}\right]\right| + \frac{1}{T}\sum_{i=1}^{T}\log(|\mathcal{X}|)\cdot\delta' + h(\delta') \\ &\le \left|\bar{W} - \mathrm{E}\left[\bar{W}\right]\right| + \log(|\mathcal{X}|)\cdot\delta' + h(\delta'),\end{aligned}$$

where the second inequality follows from Fact 2.8 (recall that $h(p) = p\log(1/p)+(1-p)\log(1/(1-p))$ is the binary entropy function). Since $h(p) \leq 2\sqrt{p}$ for any $p \in [0,1]$, we have

$$\log(|\mathcal{X}|)\cdot\delta' + h(\delta') \leq \log(|\mathcal{X}|)\cdot\delta' + 2\sqrt{\delta'} \leq \log(|\mathcal{X}|)\cdot\sqrt{\delta'} + 2\sqrt{\delta'} = \sqrt{\delta'(\log(|\mathcal{X}|)+2)^2} \leq \varepsilon/2,$$

where the last inequality follows since $\delta' \leq \varepsilon'$ and from the choice of ε'. Thus, it holds that

$$\begin{aligned}\Pr\Big[\big|\bar{Z} - \mathrm{E}\big[\bar{W}\big]\big| > \varepsilon \;\Big|\; \forall i\colon \mathrm{SD}\Big(P_i, \hat{P}_i\Big) \leq \delta'\Big] &\leq \Pr\big[\big|\bar{W} - \mathrm{E}\big[\bar{W}\big]\big| > \varepsilon - \log(|\mathcal{X}|)\cdot\delta' - h(\delta')\big] \\ &\leq \Pr\big[\big|\bar{W} - \mathrm{E}\big[\bar{W}\big]\big| > \varepsilon/2\big],\end{aligned}$$

where the first inequality follows from the coupling of Z_i and W_i and from Fact 2.8. To bound the last term we use Hoeffding's inequality:

Fact 5.16 (Hoeffding's inequality)**.** *Let $W_1,\ldots,W_n$ be independent random variables bounded in the interval $[a_i,b_i]$. Let $\bar{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$. Then, for any $\varepsilon > 0$ it holds that*

$$\Pr\big[\big|\bar{W} - \mathrm{E}\big[\bar{W}\big] > \varepsilon\big|\big] \leq 2\cdot 2^{-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i-a_i)^2}}.$$

All in all, we get that

$$\begin{aligned}\Pr\Big[\Big|\mathsf{Ent}(1^\lambda) - \mathrm{H}(X|Y)\Big| > \varepsilon\Big] &\leq 2\cdot 2^{-\frac{T(\varepsilon/2)^2}{\log^2(|\mathcal{X}|)}} + T\cdot\delta' \\ &\leq \delta/2 + \delta/2 \\ &= \delta,\end{aligned}$$

where the second inequality follows from the choice of T and δ'.

As for the running time, computing $\delta = \delta(\lambda)$ and sampling from Y are done in $\mathsf{poly}(\lambda)$. Computing Z_i can be done in $\mathsf{poly}(|\mathcal{X}|)$ time. By construction $T = \mathsf{polylog}(|\mathcal{X}|, 1/\delta)\cdot\mathsf{poly}(1/\varepsilon)$ and $\varepsilon' = 1/\mathsf{poly}(\log(|\mathcal{X}|), 1/\varepsilon)$, and thus $\delta' = 1/\mathsf{poly}(|\mathcal{X}|, 1/\varepsilon, 1/\delta)$. Finally, every call to $\mathsf{Hist}^{(X,Y,\delta',\delta')}$ takes $\mathsf{poly}(\lambda, |\mathcal{X}|, 1/\delta') = \mathsf{poly}(\lambda, |\mathcal{X}|, 1/\varepsilon, 1/\delta)$ time. All in all, the running time is thus $\mathsf{poly}(\lambda, |\mathcal{X}|, 1/\varepsilon, 1/\delta)$, as required. □

5.6 Proving Lemma 5.1

Equipped with the results of Sections 5.3 to 5.5, we are now ready to prove Lemma 5.1. Namely, constructing a public-key encryption scheme based on the existence of a suitable trapdoor pseudoentropy generator.

We begin with a brief overview of the proof. Our first step is to show that the scheme outlined in Fig. 6 is a weak PKE scheme. Recall that our construction is parameterized by k and δ, where the former determines the number of repetitions and the latter determines the output length of the hash function. The construction also require access to an approximation algorithm. The proof sets k and δ and instantiates a weak version of the approximation algorithm such that: (1) the conditions in the weak correctness lemma (Lemma 5.11) and the weak security lemma (Lemma 5.12) are satisfied; (2) the loss from only implementing the weak version of the approximating algorithm is small; and (3) the running time of all the algorithms is polynomial in the security parameter λ.

We set $\delta = O(n)$ and $k = \Omega(q^2/\delta^2)$. This setting indeed satisfies the conditions of the weak correctness and security lemmas. As for the approximation algorithms, we require $O(\delta)$-approximation for the entropy of the decoder's message in the pseudoentropy generator scheme. Since $k \cdot q = O(q^3/n^2) = O(\log(\lambda))$ and $\delta = O(n) = 1/\log(\lambda)$, we can implement such approximation algorithms whose running times is $\mathsf{poly}(\lambda)$ (Section 5.5). As for the running times of the algorithms of the PKE scheme, it is easy to see that the key-generation algorithm KeyGen and the encryption algorithm Enc run in polynomial time. As for the decryption algorithm Dec, its running time is exponential in $k \cdot q$. Since $k \cdot q = O(\log(\lambda))$, Dec's running time is also $\mathsf{poly}(\lambda)$.

After establishing the existence of a weak public-key encryption scheme we use the amplification result of Holenstein and Renner [HR05] to obtain a full-fledged semantically-secure public-key encryption scheme.

We now proceed to the formal proof.

Proof of Lemma 5.1. Let $\gamma = \gamma(\lambda) \in [0,1]$, $q = q(\lambda) \in \mathbb{N}$ and $n = n(\lambda) > 0$ such that $n \leq q$, $q^3/n^2 = O(\log(\lambda))$ and $\gamma = o(n^2/q^2)$. Let $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ be a q-laconic n-entropic trapdoor pseudoentropy generator scheme with correctness error γ. Let $PK = \{PK_\lambda\}_{\lambda\in\mathbb{N}}$, $SK = \{SK_\lambda\}_{\lambda\in\mathbb{N}}$, $U = \{U_\lambda\}_{\lambda\in\mathbb{N}}$ and $V' = \{V'_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of jointly distributed random varialbes defined as $(PK_\lambda, SK_\lambda) \leftarrow \mathsf{KeyGen}'(1^\lambda)$, $(U_\lambda, \cdot) \leftarrow \mathsf{Enc}(1^\lambda, PK_\lambda)$ and $V'_\lambda \leftarrow \mathsf{Dec}(1^\lambda, PK_\lambda, SK_\lambda)$.

Set $\delta = n/6$ and $k = \left\lceil \frac{21000 \cdot q^2}{\delta^2} \right\rceil$. Note that $\delta \in [0, q]$, $k \cdot q = O(q^3/n^2) = O(\log(\lambda))$ and $k \cdot \delta \geq 32$. Let $\widetilde{\mathsf{Ent}} = \mathsf{Ent}^{(V',(PK,SK,U),\delta/4,1/\lambda)}$, be the algorithm from Lemma 5.15. Finally, let

$$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}) = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})_{\delta,k,q,\mathsf{KeyGen}',\mathsf{Enc}',\mathsf{Dec}',\widetilde{\mathsf{Ent}}}.$$

We show that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a $(7/8)$-correct $(1/15)$-secure public-key encryption scheme; that is, the decryption succeeds with probability $0.5 + 0.5 \cdot (7/8)$, any adversary can decrypt without the secret-key with probability at most $0.5 + 0.5 \cdot (1/15)$ and the algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ all run in $\mathsf{poly}(\lambda)$ time. (See Definition 2.1.)

Correctness. Clearly, the setting of k and δ satisfies that $k \cdot \delta^2 \geq 21000 \cdot q^2$. Furthermore, $k \cdot \gamma = O\left(\frac{q^2}{n^2} \cdot \gamma\right) = o(1)$. At this point we would like to use Lemma 5.11. However, recall that in proving Lemma 5.11 we assumed that the approximation algorithm *always* satisfied Definition 5.10. However, Lemma 5.15 guarantees this to hold for $\widetilde{\mathsf{Ent}}$ only with high probability. To overcome this issue we use the fact that the scheme makes only a single oracle call to $\widetilde{\mathsf{Ent}}$.

Formally, let E_1 be the event that $\widetilde{\mathsf{Ent}}$ returns a value that satisfies Definition 5.10. By the setting of parameters above, it holds that $\Pr[\neg E] \leq 1/\lambda$. Then, for large enough λ, Lemma 5.11 yields that

$$\begin{aligned}
\Pr\left[\mathsf{Dec}\left(1^\lambda, \mathsf{sk}, \mathsf{Enc}(1^\lambda, \mathsf{sk}, \mathsf{pk}, \sigma)\right) = \sigma\right] &\geq \Pr\left[\mathsf{Dec}\left(1^\lambda, \mathsf{sk}, \mathsf{Enc}(1^\lambda, \mathsf{sk}, \mathsf{pk}, \sigma)\right) = \sigma \middle| E_1\right] \\
&\quad - \Pr[\neg E_1] \\
&\geq 1 - 2^{-5} - \left(\frac{1}{\lambda}\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \frac{15}{16} - \left(\frac{1}{\lambda}\right) \\
&> \frac{1}{2} + \frac{1}{2} \cdot \frac{7}{8},
\end{aligned}$$

as required.

Security. We start by showing that the setting of k and δ satisfies the conditions of Lemma 5.12. Clearly, the setting of k and δ satisfy that $k \cdot \delta^2 \geq 120 \cdot q^2$ and $k \cdot (n - 3\delta) \geq 14$. To see that $16(q+2)^2 \cdot \gamma \leq 9\delta^2$ for large enough λ, note that $16(q+2)^2 \cdot \gamma = o(n^2)$ while $\delta^2 = \Omega(n^2)$.

Similar to the correctness case, let E_1 be the event that $\widetilde{\mathsf{Ent}}$ returns a value that satisfy Definition 5.10. Lemma 5.12 now yields that for every polynomial-time algorithm A and sufficiently large λ, it holds that

$$\begin{aligned}\Pr\left[\mathsf{A}\left(1^\lambda, \mathsf{pk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\right) = \sigma\right] &\leq \Pr\left[\mathsf{A}\left(1^\lambda, \mathsf{pk}, \mathsf{Enc}(1^\lambda, \mathsf{pk}, \sigma)\right) = \sigma \middle| E_1\right] + \Pr[\neg E_1] \\ &\leq \frac{1}{2} + 2^{-5} + \frac{1}{\lambda} \\ &= \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{16} + \frac{1}{\lambda} \\ &< \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{15},\end{aligned}$$

as required.

Running Times. We begin by showing that the running time of $\widetilde{\mathsf{Ent}}$ is polynomial. Indeed, since the trapdoor pseudoentropy generator is q-laconic, and since $q = O(\log(\lambda))$, it holds that $\mathrm{Supp}(V') = \mathsf{poly}(\lambda)$. Moreover, since by assumption KeyGen' and Enc' run in $\mathsf{poly}(\lambda)$ time, sampling from (PK, SK, U) can be done in $\mathsf{poly}(\lambda)$ time: simply call $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}'(1^\lambda)$ and $(u, \cdot) \leftarrow \mathsf{Enc}'(\mathsf{pk})$, and output $(\mathsf{pk}, \mathsf{sk}, u)$. Finally, using that by assumption Dec also run in $\mathsf{poly}(\lambda)$ time, we can also sample from $V'|(PK = \mathsf{pk}, SK = \mathsf{sk}, U = u)$ for every $(\mathsf{pk}, \mathsf{sk}, u) \in \mathrm{Supp}(PK, SK, U)$: simply output $\mathsf{Dec}'(\mathsf{pk}, \mathsf{sk}, u)$. Hence, $\widetilde{\mathsf{Ent}}$ runs in $\mathsf{poly}(\lambda)$ time.

We can now show that the algorithms of the encryption scheme are also polynomial-time.

- KeyGen makes k calls to KeyGen' and a single oracle call to $\widetilde{\mathsf{Ent}}$ and thus run in $\mathsf{poly}(\lambda)$ time.
- Enc makes k calls to Enc' and samples and evaluates a universal hash function from $\mathcal{H}_{k \cdot q, m}$, where $m = \lceil k \cdot (\ell + \delta) \rceil$, for $\ell \leq q$. By Fact 2.4 sampling and evaluating such hash function can be done in $\mathsf{polylog}(\lambda)$, and thus the running time of Enc is $\mathsf{poly}(\lambda)$.
- Dec makes $k \cdot 2^{10 \cdot k(\ell+\delta)}$ calls to Dec' and evaluates the hash function $2^{10 \cdot k(\ell+\delta)}$ times. As both ℓ and δ are less than q and $k \cdot q = O(\log \lambda)$, the running time of Dec is $\mathsf{poly}(\lambda)$.

We have shown that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is $(7/8)$-correct $(1/15)$-secure public-key encryption scheme. Since $(7/8)^2 > 1/15$, the amplification result of Holenstein and Renner [HR05] (given in Theorem 2.2) yields that there exists public-key encryption scheme. □

6 Extensions

In this section, we exhibit two relaxations of Assumption 3.5 that are suffice for our construction of PKE. The first relaxation, discussed in Section 6.1, is implied by several concrete assumptions that have been used in the past to construct PKE, and the other, discussed in Section 6.2, turns out to be equivalent to the existence of PKE itself, leading to a complexity-theoretic characterization

of the same. Lastly, in Section 6.3, also show how Assumption 3.5 can be strengthened to yield a (two-message) oblivious transfer protocol.

6.1 A Weaker Assumption

Recall that $\mathcal{L}$ is a language in NP with witness relation $\mathcal{R}_{\mathcal{L}}$, and $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$ are sampling algorithms that output, with all but negligible probability, samples of the form $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and $x \notin \mathcal{L}$ respectively.

Towards weakening Assumption 3.5, which we used earlier to construct PKE, we define a more general variant of SZK arguments by relaxing the requirements in its definition to only hold on *average*. For instance, we earlier required that for any $x \in \mathcal{L}$ and witness w for it, the prover $\mathbf{P}$ that knows w can make the verifier $\mathbf{V}$ accept with high probability. Instead, we will now require this to hold only with high probability over the (x, w) pairs produced by $\mathsf{Y}_{\mathcal{L}}$. (In particular, there may exist some rare $(x, w) \in \mathcal{R}_{\mathcal{L}}$ for which $\mathbf{P}$ is unable to make $\mathbf{V}$ accept.)

Similarly, the soundness condition is relaxed to only require that security against cheating provers hold with high probability over the x's produced by $\mathsf{N}_{\mathcal{L}}$ – there may be some $x \notin \mathcal{L}$ where a malicious prover $\mathbf{P}^\star$ is able to make $\mathbf{V}$ accept, but these are rare. Zero-knowledge is also relaxed and is required to hold only with high probability over the x's produced by $\mathsf{Y}_{\mathcal{L}}$. We refer to this notion as *average-case* SZK (ASZK).

Definition 6.1 (ASZK Arguments)**.** *Let $c, s \colon \mathbb{N} \to [0, 1]$. An interactive proof system $(\mathbf{P}, \mathbf{V})$ is an* Average-case Statistical Zero Knowledge (ASZK) argument *(against honest verifiers) for $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ with completeness error c and soundness error s if the following properties hold:*

- **Efficiency:** *Both $\mathbf{P}$ and $\mathbf{V}$ are probabilistic polynomial-time algorithms.*
- **Completeness:** *For all large enough λ:*

$$\Pr_{(x,w)\leftarrow\mathsf{Y}_{\mathcal{L}}(1^\lambda)}\Big[(\mathbf{P}(w), \mathbf{V})(1^\lambda, x) \text{ accepts}\Big] \geq 1 - c(\lambda).$$

- **Soundness:** *For any, possibly malicious,* polynomial-time *$\mathbf{P}^\star$ and all large enough λ:*

$$\Pr_{x\leftarrow\mathsf{N}_{\mathcal{L}}(1^\lambda)}\Big[(\mathbf{P}^*, \mathbf{V})(1^\lambda, x) \text{ accepts}\Big] \leq s(\lambda).$$

- **Honest Verifier Statistical Zero Knowledge:** *There is a polynomial-time algorithm $\mathbf{S}$, called the simulator, such that for random $(x, w) \leftarrow \mathsf{Y}_{\mathcal{L}}(1^\lambda)$, the simulator $\mathbf{S}$ simulates the transcript of the interactive proof given x. That is, the following holds for all large enough λ:*

$$\mathop{\mathrm{E}}_{(x,w)\leftarrow\mathsf{Y}_{\mathcal{L}}}\Big[\mathrm{SD}\Big((\mathbf{P}(w), \mathbf{V})(1^\lambda, x), \mathbf{S}(1^\lambda, x)\Big)\Big] \leq \mathsf{negl}(\lambda).$$

Note that since $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$ are concentrated on instances in and not in $\mathcal{L}$ respectively, any protocol that is an SZK argument for $\mathcal{L}$ is also an ASZK argument for $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ with the correctness, soundness and simulation errors degraded by a negligible additive factor.

We will still require our arguments to be laconic – that the number of rounds and the size of the prover's messages be small. We state our next assumption as follows.

Assumption 6.2. *There exists a language $\mathcal{L} \in \mathsf{NP}$ with associated distributions over instances $(\mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$, and a constant $\alpha < 1/2$ such that:*

1. *$(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is (poly, α)-cryptographically hard.*
2. *There is an r-round q-laconic honest-verifier ASZK argument for $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ with completeness error c and soundness error s such that:*
   - *There is a constant $\beta > 0$ such that for large enough λ: $1 - c(\lambda) > s(\lambda) + \alpha + \beta$.*
   - *q and r are such that $r^2 \cdot q^3 = O(\log(\lambda))$.*

Notice that we have weakened Assumption 3.5 in two ways. First, where we earlier required SZK arguments, now we only require ASZK arguments. As noted earlier, SZK arguments are themselves also ASZK arguments with almost the same correctness, soundness and simulation errors.

Second, we relax requirements of the cryptographic hardness of $\mathcal{L}$. In Assumption 3.5 we required that $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ be cryptographically hard – that no polynomial-time algorithm be able to distinguish between $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$ with non-negligible advantage. In Assumption 6.2, however, we only require that this advantage be less than a fixed constant α that satisfies certain properties in relation to the completeness and soundness errors.

Despite being weaker, Assumption 6.2 suffices, with very few changes to the proofs in Sections 4 and 5, for our construction of a PKE.

Theorem 6.3. *If Assumption 6.2 holds, then there exists a PKE scheme.*

Proof Sketch. We show how to modify the statement and proof of Lemma 4.3 and the proof of Theorem 3.6 so they can be applied with Assumption 6.2 (instead of Assumption 3.5). The proof of Lemma 5.1 remains unchanged.

In the statements and proof of Lemma 4.3, $\mathrm{KL}(1-c||s)$ is replaced with $\mathrm{KL}(1-c||s+\alpha)$, where α is the cryptographic hardness parameter from Assumption 6.2. The construction proving Lemma 4.3 is the same as in Fig. 5. The proof is almost the same as in Section 4, except for the following changes:

1. The proof of Lemma 4.4 remains the same. It uses the zero-knowledge of the argument-system, but only assumes it to be average-case; see Footnote 26.
2. In Lemma 4.9, the hypothesis would only require an interactive argument system whose completeness and soundness hold on average. The proof of this lemma would then work as is, except the guarantees of the algorithm F defined there now only hold on average.
3. In the statements of Lemma 4.10, $\mathrm{KL}(1-c||s)$ is replaced with $\mathrm{KL}(1-c||s+\alpha)$, where α is the cryptographic hardness parameter from Assumption 6.2. In the proof of that lemma we make the following changes (all variables are in the context of that proof): We set $\gamma = 1/\mathsf{poly}$ such that $1 - c > s + \alpha + \gamma$ and $1/\lambda^d > \frac{2(1-c)\cdot\gamma}{(s+\alpha)\cdot r}$. Using that $\mathcal{L}$ is (poly, α)-cryptographically hard, and applying Lemma 4.9 with soundness error $s+\alpha$, we get that B_I has $\left(\mathsf{poly}, 1/\lambda^d - \frac{2(1-c)\cdot\gamma}{(s+\alpha)\cdot r}\right)$ conditional pseudoentropy at least $\mathrm{H}(B_I|X, I, C_I) + \mathrm{KL}(1-c||s+\alpha+\gamma)/r - \left(1/\lambda^d - \frac{2(1-c)\cdot\gamma}{(s+\alpha)\cdot r}\right)$. The last step of "getting rid" of the dependency in γ is done as in the proof of Lemma 4.10.
4. In the proof of Lemma 4.3, $\mathrm{KL}(1-c||s)$ is replaced with $\mathrm{KL}(1-c||s+\alpha)$.

Finally, as for the proof of Theorem 3.6 (given in the beginning of Section 5), $\mathrm{KL}(1-c||s)$ is again replaced with $\mathrm{KL}(1-c||s+\alpha)$. That proof only used that $\mathrm{KL}(1-c||s)$ is a constant (or chose it to be a constant with out loss of generality), and the same holds for $\mathrm{KL}(1-c||s+\alpha)$. □

We next argue that Assumption 6.2 implies most of the assumptions that are already known to imply PKE. Specifically, Assumption 6.2 is implied by each of the following assumptions (with whatever respective parameters are known to imply public-key encryption):

- Quadratic Residuosity
- Decisional Diffie-Hellman
- Learning Parity with Noise
- Learning With Errors
- Combinations of the LIN, DUE, and DSF assumptions from [ABW10]

The definition of the above assumptions, requisite references, and explanations of how they imply our assumption are given in Appendix A. There are some important assumptions, however, that give public-key encryption but which we do not know to imply Assumption 6.2, such as the hardness of factoring and the *computational* Diffie-Hellman assumption. Another is the 3LIN assumption from [ABW10]. It would be interesting to understand what distinguishes these assumptions from the ones in the list above.

6.2 A Complexity-Theoretic Characterization of PKE

In this section we present a complexity-theoretic characterization of public key encryption; that is, a complexity-theoretic assumption that is equivalent to the existence of public-key encryption schemes. We also show that a relaxed version of this assumption (which we get by removing a laconism condition) is equivalent to the existence of one-way functions.

Up to this point we showed the existence of public-key encryption from assumptions that require some underlying hard *decision* problem (Assumptions 3.5 and 6.2). Specifically, an NP language for which it was hard to distinguish between instances that are in the language to those that are not in the language.

In many cryptographic settings, however, it seems more natural to consider hardness of *search* problems. For example, the hardness of computing a secret-key corresponding to a known public-key or the hardness of finding a the pre-image of random element for a one-way function.

To get our complexity-theoretic characterization we will focus on *search* problems. Instead of assuming the existence of two sampling algorithms Y and N that sample YES and NO instances, respectively, we assume the existence of a single solved instance generator G (we use G to denote the solved instance generator instead of Y to emphasize that we are no longer considering a language membership problem). We would like to fit this generator into the framework of statistical zero-knowledge arguments we used thus far. Specifically, we need to define what does a statistical zero-knowledge argument-system wrt search problems.

The completeness property of argument-systems carries over smoothly to our current setting. It holds with respect to a random instance-witness pair sampled from G (in the same way that this property is defined with respect to Y in Assumption 6.2).

The main challenge lies in defining soundness. Standard argument-systems require the verifier to reject inputs that are not in the language. But now we do no longer have a notion of NO inputs.[34] To define soundness, we take an approach related to Proofs of Knowledge (PoK) [GMR85, BG92]. In a PoK, not only is the verifier convinced that the instance is in the language, but also that the prover *knows* a witness to this affect. This "knowledge" is captured in a form of an efficient extractor that can retrieve the witness given black-box access to the prover's strategy.

In our definition, the soundness property is replaced by what we refer to as an *Argument of Weak Knowledge* (AoWK) property, which requires that an efficient prover without access to a witness, cannot convince the verifier to accept, even though the input was generated by the solved instance generator. This is a weakening of the notion of *Argument* of Knowledge (AoK, where, in contrast to PoK, the extractor is only required to work with respect to efficient provers). Indeed, we do not require the ability to extract a witness from the prover, but only the inability to convince the verifier without one. Observe that no guarantee is given against provers who have partial access to the witness.

Lastly, we want to define the zero-knowledge property. In the standard setting, satisfying this property roughly translates into ensuring that the verifier learns nothing beyond the fact that the input is in the language. Again we face the issue that we no longer have a language. In this case, however, the solution is simple. In our definition, satisfying the zero-knowledge property roughly translates into ensuring that the verifier learns nothing beyond the fact that the prover knows a witness. The formal definition naturally follows the simulation paradigm, and is in fact identical to way that this property is defined in Assumption 6.2.

Definition 6.4 (ASZK Arguments of Weak Knowledge)**.** *Let $c, s\colon \mathbb{N} \to [0,1]$. An interactive proof system $(\mathbf{P}, \mathbf{V})$ is an* Average-case Statistical Zero Knowledge (ASZK) Argument of Weak Knowledge (AoWK) *(against honest verifiers) for G with completeness error c and soundness error s if the following properties hold:*

- **Efficiency:** *Both $\mathbf{P}$ and $\mathbf{V}$ are probabilistic polynomial-time algorithms.*
- **Completeness:** *For all large enough λ:*

$$\Pr_{(x,w)\leftarrow\mathsf{G}(1^\lambda)}\Big[(\mathbf{P}(w),\mathbf{V})(1^\lambda,x) \text{ accepts}\Big] \geq 1 - c(\lambda)$$

- **Argument of Weak Knowledge:** *For any, possibly malicious,* polynomial-time $\mathbf{P}^\star$ *and all large enough λ:*

$$\Pr_{(x,\cdot)\leftarrow\mathsf{G}(1^\lambda)}\Big[(\mathbf{P}^*,\mathbf{V})(1^\lambda,x) \text{ accepts}\Big] \leq s(\lambda)$$

- **Honest Verifier Statistical Zero Knowledge:** *There is a polynomial-time algorithm $\mathbf{S}$, called the simulator, such that for $(x,w) \leftarrow \mathsf{G}(1^\lambda)$, the simulator $\mathbf{S}$ simulates the transcript of the interactive proof given just x. That is, the following holds for all large enough λ:*

$$\mathop{\mathrm{E}}_{(x,w)\leftarrow\mathsf{G}(1^\lambda)}\Big[\mathrm{SD}\Big((\mathbf{P}(w),\mathbf{V})(1^\lambda,x),\mathbf{S}(1^\lambda,x)\Big)\Big] \leq \mathsf{negl}(\lambda).$$

[34] Moreover, it may be that every string x is in the support of G. Consider, e.g., the task of inverting a one-way permutation over $\{0,1\}^\lambda$.

Remark 6.5 (Worst-case AoWK)**.** *The above definition considers* average-case *notions of completeness, weak knowledge and zero knowledge. This fact does not seem inherent to the notion of arguments of weak knowledge and in fact, we find the worst-case variant to be more natural. However, for our results it is important that we focus on the average-case variant. We leave the study of the worst-case variant to future research.*

The existence of an ASZK argument of weak knowledge for G immediately implies that it is hard to compute a witness for a random instance. Indeed, if this would not be the case, then a cheating prover can find such a witness and run the honest prover with this witness to convince the verifier, breaking the argument of weak knowledge property.

We are now ready to state our assumption and show its equivalence to public-key encryption.

Assumption 6.6. *There is an r-round q-laconic honest-verifier* ASZK *argument of weak knowledge for a solved instance generator* G *with completeness error c and soundness error s such that:*

- *There is a constant $\beta > 0$ such that for large enough λ: $1 - c(\lambda) > s(\lambda) + \beta$; and*
- *q and r are such that $r^2 \cdot q^3 = O(\log(\lambda))$.*

Theorem 6.7. *Assumption 6.6 is equivalent to the existence of public-key encryption schemes.*

Proof Sketch. A key observation is that in the proof of Theorem 6.3 we never really used the existence of an underlying language $\mathcal{L}$. Instead, we used the sampling algorithms Y and N. Hence, given a solved instance generator G we can define $\mathsf{Y} \equiv \mathsf{G}$, and N to output x such that $(x, w) \leftarrow \mathsf{G}$. Note that Y and N actually produce the *same* distribution over the instances (and thus cannot describe a language). It follows that (Y, N) satisfy the conditions of Assumption 6.2 (with $\alpha = 0$), and thus one direction of Theorem 6.7 follows from Theorem 6.3. It is left to show that PKE implies Assumption 6.6.

Suppose there exists a PKE scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We define the solved instance generator G to be the same as KeyGen, outputting pk as the instance and sk as the witness. An ASZK-AoWK for G is presented in Figure 7.

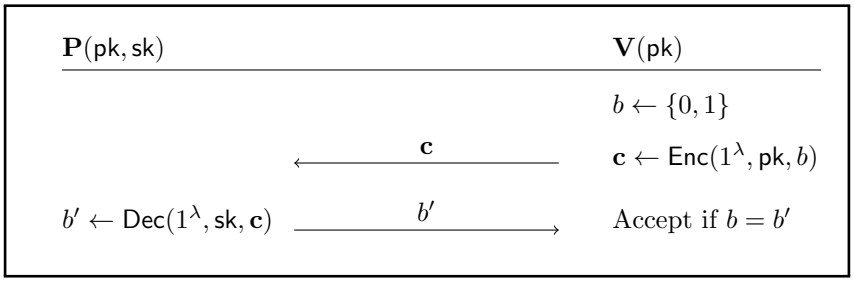

Figure 7: ASZK argument of weak knowledge for G

The completeness of this protocol follows from the correctness of the encryption scheme, and argument of weak knowledge follows from semantic security. These imply that the completeness error is negligible and the soundness error is only negligibly more than $1/2$.

The simulator, on input pk, runs the verifier $\mathbf{V}$ that generates b and $\mathbf{c}$. It then sets the prover's message to be b itself and outputs $(b, \mathbf{c}, b)$ as the transcript. The distance of this distribution from

the actual transcript is now exactly the probability that $\mathbf{P}$ does not guess b correctly. Thus the simulation error is the same as the completeness error, which is negligible.

The laconism follows immediately from the structure of the protocol. This complete the proof of the second direction and Theorem 6.7 follows. □

Interestingly, when we remove the laconism requirement from Assumption 6.6, then the resulting primitive is equivalent to the existence of one-way functions.

Proposition 6.8. *The following two statements are equivalent:*

- *There is an r-round q-laconic honest-verifier* ASZK *argument of weak knowledge for a solved instance generator* G *with completeness error c and soundness error s such that $1 - c(\lambda) > s(\lambda) + \beta$ for some constant $\beta > 0$ and large enough λ.*
- *One-way functions exist.*

Theorem 6.7 and Proposition 6.8 illustrate again (and perhaps more clearly) what we argued in Section 1.1.2 — that removing the laconism requirement from our assumption would prove that the existence of one-way functions implies that of public-key encryptions.

Proof Sketch. As we argued above, the existence of an ASZK-AoWK for G implies that it is hard to find a witness for a random instance. Specifically, consider the function $f(r) = \mathsf{G}(r)_1$, where r are coins for G and $\mathsf{G}(r)_1$ denotes the instance generated by G when it is run with r set to its random coins.

We claim that f is a distributional one-way function.[35] Indeed, suppose there exists an efficient inverter that given a random instance x can find a pre-image of f that is β-close to a *random* pre-image. Namely. it finds almost random coins for G that generate x (together with an almost random witness w). We can construct a prover strategy that, given x, runs this inverter to find a β-close-to-random witness w and then runs the honest prover strategy with respect to (x, w).

By completeness, we know that the foregoing prover strategy will convince the verifier with probability $1 - c - \beta > s$, a contradiction to the argument of weak knowledge property of the protocol. Hence, f is a distributional one-way function. A distributional one-way function can be transformed into full-fledged one-way function (see [IL89, Lemma 1]). Thus, the existence of ASZK-AoWK implies the existence of a one-way function.

As for the other direction, consider the relation $(f(x), x)$, such that f is a one-way function. Clearly this relation is an NP relation. It follows from [HNO+09] that any NP relation has a statistical zero-knowledge argument of knowledge with an efficient prover, assuming the existence of one-way functions. Hence, we can define G to output the pair $(f(x), x)$ for a random x. The argument-system for the relation $(f(x), x)$ is in fact ASZK-AoWK for G. Indeed, it is immediate that the argument-system satisfies the correctness and zero-knowledge conditions. To show that it is also an argument of weak knowledge assume the contrary. Namely, that there exists an efficient cheating prover that convince the verifier to accept with high probability without knowing the witness. Since the argument-system is an argument of knowledge, we can efficiently extract the witness from such a cheating prover. This witness is a pre-image of a random output of f, a contradiction to f being one-way. □

[35] A polynomial-time computable function $f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is distributional one-way if for some polynomial p and every efficient algorithm A it holds that $\mathrm{SD}\left((X, f(X)), (\mathsf{A}(f(X)), f(X))\right) > 1/p(n)$ for sufficiently large n, where X is uniform in $\{0,1\}^n$.

6.3 Oblivious Transfer

In this section, we show how (a mild strengthening of) Assumption 3.5 yields a 2-message semi-honest oblivious transfer (OT) protocol. Using the classical protocol of Goldreich, Micali and Wigderson [GMW87], this yields general purpose secure multiparty computation from the same assumption.

In order to construct an OT scheme, we need to strengthen the notion of cryptographic hardness, which we used in our construction of PKE. The reason is that we would like one of the parties to sample a random instance x in a way that does not reveal whether x belongs to $\mathcal{L}$ or not. Note that, even in the semi-honest case that we consider here, the party that samples x has access not only to x but also to the random coins that sampled x. In particular, if we used the naive sampler that samples $x \leftarrow \mathsf{Y}_{\mathcal{L}}$ with probability $1/2$ and $x \leftarrow \mathsf{N}_{\mathcal{L}}$ with probability $1/2$, the choice of which of the two distributions was sampled reveals whether $x \in \mathcal{L}$. Thus, we need a stronger definition of cryptographic hardness in which the adversary trying to determine whether $x \in \mathcal{L}$, sees not only the instance but also the random coins that sampled it.

A closely related issue comes in the classical construction of OT from trapdoor permutations [EGL85]. It was resolved in that context by requiring a form of "enhanced" sampling, in which elements can be sampled from the domain of the permutation so that the coins of the sampling algorithm do not reveal the inverse (see [Gol09, Appendix C.1] or [GR13] for further discussion). In direct analogy, our strengthening of cryptographic hardness also introduces an enhanced sampling condition. Namely, that there is a way to sample an instance x such that no efficient adversary, even given the random coins of the sampler, can distinguish whether $x \in \mathcal{L}$ or $x \notin \mathcal{L}$ (except with negligible advantage).

Analogous to our PKE construction, we only construct *weak* semi-honest 1-round OT protocol. Weak OT means that the correctness and security errors are small constants, rather than being negligible. To get a a full-fledged OT protocol, we use known amplification techniques (e.g., [Wul07]).

First we formalize the notion of *weak* oblivious transfer that we need. The definition is adapted from [Wul07].

Definition 6.9 (Weak OT)**.** *An $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$-*Weak Oblivious Transfer *is a two-party protocol, between a sender $\mathbf{S}$ and a receiver $\mathbf{R}$. The sender is given as input a security parameter 1^λ as well as two inputs $\sigma_0, \sigma_1 \in \{0,1\}$ and the receiver gets as input 1^λ and a bit $\beta \in \{0,1\}$. An $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$-WOT has the following properties:*

1. **Correctness:** *The receiver learns σ_β with probability at least $1 - \varepsilon_1$, where the probability is over the randomness of both the sender and the receiver.*

2. **Sender's Privacy:** *For any probabilistic polynomial time algorithm $\mathsf{R}^\star$, for any choices of $\sigma_0, \sigma_1, \beta \in \{0,1\}$ and sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{R}^\star\left(\mathrm{view}_{\mathbf{R}}(1^\lambda, \sigma_0, \sigma_1, \beta)\right) = \sigma_{1-\beta}\right] < \frac{1+\varepsilon_2}{2}$$

where $\mathrm{view}_{\mathbf{R}}$ corresponds to the honest receiver's view. Namely, the index β, his private randomness, and the transcript.

3. **Receiver's Privacy:** *For any probabilistic polynomial time algorithm* $\mathsf{S}^\star$*, for any choices of* $\sigma_0, \sigma_1, \beta \in \{0,1\}$ *and sufficiently large* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{S}^\star\left(\mathrm{view}_{\mathbf{S}}(1^\lambda, \sigma_0, \sigma_1, \beta)\right) = \beta\right] < \frac{1+\varepsilon_3}{2}$$

where $\mathrm{view}_{\mathbf{S}}$ *corresponds to the honest sender's view. Namely,* σ_0, σ_1*, his private randomness, and the transcript.*

We now turn to defining our notion of *enhanced* cryptographic hardness.

Definition 6.10 (Enhanced Cryptographic Hardness)**.** *Let* $t = t(\lambda) \in \mathbb{N}$ *and* $\varepsilon = \varepsilon(\lambda) \in [0,1]$*. The tuple* $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L}, \mathsf{O}_\mathcal{L})$ *is* (t,ε)-enhanced cryptographically hard *if the following properties hold.*

1. **Cryptographic Hardness:** $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ *is* (t,ε)*-cryptographically hard.*
2. **Correctness:** *The sampler* $\mathsf{O}_\mathcal{L}$ *is a probabilistic polynomial-time algorithm such that*

$$\mathrm{SD}\left(\mathsf{O}_\mathcal{L}(1^\lambda), x_\beta\right) < \mathsf{negl}(\lambda)$$

where x_β *is generated by first sampling* $(x_0, w_0) \leftarrow \mathsf{Y}(1^\lambda)$ *and* $x_1 \leftarrow \mathsf{N}(1^\lambda)$ *and outputting* x_β *for randomly chosen* $\beta \leftarrow \{0,1\}$*.*

3. **Enhanced Indistinguishability:** *For every probabilistic polynomial-time adversary* A *that on input* $(1^\lambda, s)$ *runs in time* $t(\lambda)$ *and all sufficiently large* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathsf{A}(1^\lambda, s) = \mathcal{L}(\mathsf{O}_\mathcal{L}(1^\lambda; s))\right] < \frac{1}{2} + \varepsilon(\lambda)$$

where s *is a uniformly random string and* $\mathsf{O}_\mathcal{L}(1^\lambda; s)$ *indicates that the oblivious sampler* $\mathsf{O}_\mathcal{L}$ *is run with randomness* s*; and* $\mathcal{L}(x) = 1$ *if and only if* $x \in \mathcal{L}$ *and the probability is also over the random coins of* A*.*

We say that the $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ *is* enhanced cryptographically hard *if it is* $(\lambda^c, 1/\lambda^c)$*-hard for every constant* $c > 0$*.*

Remark 6.11 (Doubly Enhanced Cryptographic Hardness)**.** *It is natural to also consider a "doubly enhanced" variant of cryptographic hardness, in analogy to doubly-enhanced trapdoor permutations (see [GR13]). However, since we do not require this notion for our construction, we avoid doing so in this work.*

Using the foregoing notion of enhanced cryptographic hardness, we are ready to state our result on the existence of weak OT (which we shall later amplify to full fledged OT).

Lemma 6.12 (Weak Oblivious Transfer)**.** *Assuming that there exists a language* $\mathcal{L}$ *that satisfies Assumption 3.5 and, moreover, is* enhanced *cryptographically hard (as in Definition 6.10), then there exists a 1-round* $(2^{-4}, 2^{-4}, \mathsf{negl}(\lambda))$*-weak oblivious transfer protocol.*

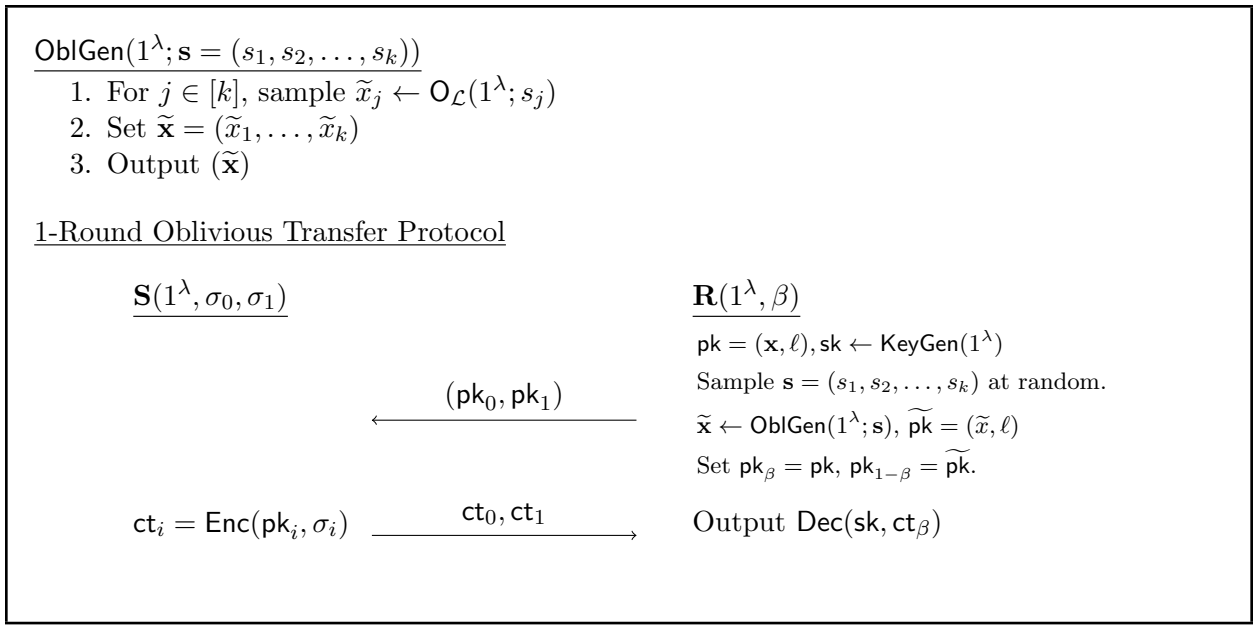

Figure 8: A Weak Oblivious Transfer Protocol

Proof sketch. This proof builds on the construction of the weak public key encryption scheme in Theorem 3.6. Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be the the weak public-key encryption scheme from Fig. 6. Our OT protocol is reminiscent of the construction of Oblivious Transfer from Lossy encryption. The receiver samples two public keys — one real public key along with its secret key and a fake public key whose ciphertexts it cannot decrypt. The real public key is generated by KeyGen. It consists of a tuple of yes-instances sampled by $\mathsf{Y}_{\mathcal{L}}$ along with ℓ, an estimate on the entropy of the distribution (see discussion in Section 5). The fake public key on the other hand consists of obliviously sampled instances using $\mathsf{O}_{\mathcal{L}}$. The receiver sends the real public key as pk_β for the receiver's input β and the fake public key as $\mathsf{pk}_{1-\beta}$. The sender now encrypts his inputs σ_0, σ_1 with the corresponding public keys and sends the ciphertexts $(\mathsf{ct}_0, \mathsf{ct}_1)$ across. The receiver can then decrypt ct_β using the secret key. In lossy encryption, the fake public key is a lossy key. In that case, the ciphertext $\mathsf{ct}_{1-\beta}$ reveals no information about senders second input $\sigma_{1-\beta}$. In our case, the fake public key is a tuple of instances $\widetilde{\mathbf{x}}$ sampled using the sampler $\mathsf{O}_{\mathcal{L}}$. We show that an efficient sender cannot decrypt messages encrypted with such fake public keys. The oblivious transfer protocol is described in Fig. 8.

We need to prove that our protocol satisfies the three properties of weak OT. These are summarized in the following three claims.

Claim 6.12.1 (Correctness)**.** *The receiver learns the value σ_β with probability at least $1 - 2^{-5}$*

Proof Sketch. This follows from the correctness of the encryption scheme (see Lemma 5.11). □

Claim 6.12.2 (Receiver's Privacy)**.** *For any probabilistic polynomial time algorithm $\mathsf{S}^\star$, for any choices of $\sigma_0, \sigma_1, \beta \in \{0,1\}$ and sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{S}^\star\Big(\mathrm{view}_{\mathbf{S}}(1^\lambda, \sigma_0, \sigma_1, \beta)\Big) = \beta\right] < \frac{1}{2} + \mathsf{negl}(\lambda)$$

Proof Sketch. To recover β, the honest-but-curious sender has to determine which instance was sampled using the enhanced sampler $\mathsf{O}_{\mathcal{L}}$ and which instance was sampled using the $\mathsf{Y}_{\mathcal{L}}$ sampler. From the correctness property of the enhanced sampling (Definition 6.10), the distribution on yes-instances generated by $\mathsf{O}_{\mathcal{L}}$ is statistically close to those generated by $\mathsf{Y}_{\mathcal{L}}$. So, to distinguish between the two, the adversary has to distinguish between the no-instances sampled by $\mathsf{O}_{\mathcal{L}}$ and $\mathsf{Y}_{\mathcal{L}}$. This is equivalent to distinguishing between outputs of $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$. No efficient adversary can distinguish between the two due to the (standard) cryptographic hardness of $\mathcal{L}$. □

Lemma 6.13 (Sender's Privacy)**.** *For any probabilistic polynomial-time algorithm $\mathsf{R}^\star$, for any choices of $\sigma_0, \sigma_1, \beta \in \{0,1\}$ and sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{R}^\star\left(\text{view}_{\mathbf{R}}(1^\lambda, \sigma_0, \sigma_1, \beta)\right) = \sigma_{1-\beta}\right] < \frac{1}{2} + 2^{-5}.$$

Proof Sketch. We need to show that the receiver, even given the random coins for sampling the fake public key $\widetilde{\mathbf{x}}$, cannot decrypt (with high probability). Namely, prove the following.

Claim 6.13.1 (Weak security for OT)**.** *For every polynomial time adversary $\mathsf{R}^\star$, it holds*

$$\Pr\left[\mathsf{R}^\star\left(1^\lambda, \widetilde{\mathsf{pk}}, \mathsf{Enc}(1^\lambda, \widetilde{\mathsf{pk}}, \sigma), \mathbf{s}\right) = \sigma\right] \leq \frac{1}{2} + 2^{-5},$$

where the above probability is over $\sigma \leftarrow \{0,1\}$, $\mathbf{s}$ and $\widetilde{\mathsf{pk}}$ sampled according to the real receiver $\mathbf{R}$'s distribution, the randomness of Enc and $\mathsf{R}^\star$.

Note that the honest-but-curious receiver $\mathsf{R}^\star$ also gets access to $\mathbf{s}$, the randomness the real receiver $\mathbf{R}$ used to sample $\widetilde{\mathsf{pk}}$.

The proof of Claim 6.13.1 is similar to the proof of the security of the encryption scheme (Lemma 5.12). Specifically, the latter proof relies on the pseudoentropy of the encoder's public message in the public-key pseudoentropy generator, which in turn relies on the KL-unpredictability of a random prefix in the argument-system for the language $\mathcal{L}$. Finally, this unpredictability was based on the cryptographic hardness of the language and the soundness of the argument-system (Lemma 4.5). To prove Claim 6.13.1 we require a stronger statement than Lemma 4.5, and to achieve this we use the enhanced cryptographic hardness of $\mathcal{L}$ (rather than the standard cryptographic hardness).

Specifically, we have the following lemma, analogous to Lemma 4.9, which is the core of the proof of Lemma 4.5 (see the text next to Lemma 4.9 for more explanations about the notations).

Claim 6.13.2. *Let $c = c(\lambda) \in [0,1]$, $s = s(\lambda) \in (0,1]$ and $\gamma = \gamma(\lambda) \in [0,1]$. Let $r = r(\lambda) \in \mathbb{N}$, and let $t = t(\lambda) \in \mathbb{N}$ be polynomially bounded. Assume that*

1. *$\mathcal{L}$ is a (t,γ)-enhanced cryptographically hard language;*
2. *$(\mathbf{P}, \mathbf{V})$ is an r-round interactive argument system for $\mathcal{L}$ with completeness error c soundness error s; and*
3. *$1 - c \geq s + \gamma$, for all sufficiently large values of the security parameter $\lambda \in \mathbb{N}$.*

Then, there is a polynomial p such that for the function $t'(\lambda) = t(\lambda)/p(\lambda)$, the distribution B_I is $(t', \frac{1}{r} \cdot \mathrm{KL}(1-c||s+\gamma))$-KL-hard for sampling given (X, S, I, C_I^-) where S represents the randomness used for sampling X.

The proof of Claim 6.13.2 is identical to the proof of Lemma 4.9 except that we invoke the enhanced cryptographic hardness to show that the prover's message is hard to predict even given the random coins of the oblivious sampler. If the adversary could predict the prover's next message, we can use this adversary as a cheating prover along with the verifier $\mathbf{V}$ to distinguish between the Yes and No instances sampled by the oblivious sampler $\mathsf{O}_{\mathcal{L}}$.

Given Claim 6.13.2, the proof of Claim 6.13.1 follows similar lines to those of the proof of Lemma 5.12, where we also include the randomness used to sample the instances where needed. □

This concludes the proof of Lemma 6.12. □

As a corollary, we get a 1-round OT protocol by generic amplification technique of [Wul07].

Corollary 6.14. *Assuming that there exists a language $\mathcal{L}$ that satisfies Assumption 3.5 (i.e. cryptographically hard language with a laconic* SZK *argument) and, moreover, $\mathcal{L}$ is* enhanced *cryptographically hard. Then, there exists a 1-round oblivious transfer protocol.*

Acknowledgments

We thank Vinod Vaikuntanathan for his encouragement and for helpful discussions. We thank the anonymous reviewers for very useful comments and in particular for suggesting the abstraction of trapdoor pseudoentropy generator.

Research supported in part by NSF Grants CNS-1413920 and CNS-1350619, and by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236. The third author was also supported by the SIMONS Investigator award agreement dated 6-5-12 and the Cybersecurity and Privacy Institute at Northeastern University.

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 171–180, 2010.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[AR16] Benny Applebaum and Pavel Raykov. On the relationship between statistical zero-knowledge and statistical randomized encodings. In *Annual Cryptology Conference*, pages 449–477. Springer, 2016.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697, 2016.

[BDV16] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs hardness through the obfuscation lens. *IACR Cryptology ePrint Archive*, 2016:574, 2016.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 374–383, 1997.

[BL13] Andrej Bogdanov and Chin Ho Lee. Limits of provable security for homomorphic encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 111–128. Springer, 2013.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 45–64, 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6(2):97–116, 1993.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[Gol17] Oded Goldreich. *Introduction to Property Testing*. forthcoming (http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html), 2017.

[GOVW12] Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 494–511. Springer, 2012.

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *Journal of cryptology*, 26(3):484–512, 2013.

[GV99] Oded Goldreich and Salil P. Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, page 54, 1999.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[HHRS15] Iftach Haitner, Jonathan J Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols—tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM Journal on Computing*, 44(1):193–242, 2015.

[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.

[HLWW16] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. *J. Cryptology*, 29(3):514–551, 2016.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009.

[HR05] Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 478–493, 2005.

[HR11] Thomas Holenstein and Renato Renner. On the randomness of independent experiments. *IEEE Transactions on Information Theory*, 57(4):1865–1871, 2011.

[HRV13] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM J. Comput.*, 42(3):1405–1430, 2013.

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 611–620, 2009.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61. ACM, 1989.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[LV16] Tianren Liu and Vinod Vaikuntanathan. On basing private information retrieval on np-hardness. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2016.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[NV06] Minh-Huyen Nguyen and Salil P. Vadhan. Zero knowledge with efficient provers. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 287–295, 2006.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 133–138, 1991.

[OV08] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 482–500, 2008.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings*, volume 7147 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2012.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[PW16] Yury Polyanskiy and Yihong Wu. Lecture notes on information theory. Available at: http://people.lids.mit.edu/yp/homepage/data/itlectures_v4.pdf, 2016.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Theory of Cryptography Conference*, pages 219–234. Springer, 2011.

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

[RW05] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 199–216, 2005.

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.

[Vad99] Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[VZ12] Salil Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 817–836. ACM, 2012.

[Wri17] John Wright. Personal communication, 2017.

[Wul07] Jürg Wullschleger. Oblivious-transfer amplification. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 555–572. Springer, 2007.

A Comparing Assumptions

In this section, we compare various concrete assumptions used in the past to construct public-key encryption to the generic complexity-theoretic assumptions we use. We will mostly be concerned with Assumption 6.2, which is stated again below for convenience, and which we show to be implied by a number of these concrete assumptions. We also briefly discuss other concrete assumptions that are known to imply public-key encryption but for which we do not know whether they imply Assumption 6.2.

Assumption 6.2. *There exists a language $\mathcal{L} \in \mathsf{NP}$ with associated distributions over instances $(\mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$, and a constant $\alpha < 1/2$ such that:*

1. *$(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ is (poly, α)-cryptographically hard.*
2. *There is an r-round q-laconic honest-verifier ASZK argument for $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ with completeness error c and soundness error s such that:*
   - *There is a constant $\beta > 0$ such that for large enough λ: $1 - c(\lambda) > s(\lambda) + \alpha + \beta$.*
   - *q and r are such that $r^2 \cdot q^3 = O(\log(\lambda))$.*

In each case, we first state the assumption, define the relevant language $\mathcal{L}$ and distributions $\mathsf{Y}_\mathcal{L}$ and $\mathsf{N}_\mathcal{L}$, and prove that the assumption implies that $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ is cryptographically hard. We then present a laconic ASZK argument for $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ and prove its completeness, soundness and zero-knowledge, again using the assumption.

In all of the SZK argument systems we construct the verifier chooses at random a bit b, encrypts it and sends the ciphertext to the prover. The prover decrypts the ciphertext to a bit b' and sends it to the verifier, which accepts if $b' = b$. This is not surprising since all the concrete assumptions here imply public-key encryption schemes, and we already saw in Section 6.2 that any such scheme implies an argument system with this structure.

Section Organization. In Appendix A.1 we show that Lossy encryption schemes imply Assumption 6.2 (in fact, they imply the even stronger version, Assumption 3.5). In Appendix A.2 we show that Learning Parity with Noise (LPN) implies Assumption 6.2. Finally, in Appendix A.3 we show that two assumptions made in [ABW10] also imply Assumption 6.2.

A.1 Lossy Encryption

Lossy Encryption [PVW08, BHY09] schemes have two modes of operation: the real and the lossy mode. In the real mode, it behaves like a semantically secure public key encryption scheme while in the lossy mode, the ciphertexts contain *no information* about the message encoded.

Definition A.1 (Lossy Encryption)**.** *A Lossy Encryption scheme is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where the $\mathsf{Gen}(1^\lambda, \mathsf{mode})$ has two modes. A real mode where it behaves like a semantically secure encryption scheme and a lossy mode that produces fake public keys. In the real mode, it outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. In the lossy mode, it outputs a lossy public key $\widetilde{\mathsf{pk}}$. The encryption algorithm $\mathsf{Enc}(\mathsf{pk}, \sigma)$ outputs ciphertexts ct given the message and the public key and the decryption algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ returns a decrypted message given the secret key and the ciphertext.*

The encryption scheme satisfies the following two additional properties:

- **Key Indistinguishability:** *Real public keys are indistinguishable from lossy public keys. That is,*

$$\left\{\mathsf{pk} \text{ where } (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda,\mathsf{real})\right\}_\lambda \approx_c \left\{\widetilde{\mathsf{pk}} \text{ where } \widetilde{\mathsf{pk}} \leftarrow \mathsf{Gen}(1^\lambda,\mathsf{lossy})\right\}_\lambda.$$

- **Lossy Encryption:** *Encryption using the lossy key completely loses information about the message encrypted. That is, output distributions of encryptions of* 0 *and* 1*, under lossy keys, are statistically indistinguishable. For every* $\widetilde{\mathsf{pk}} \leftarrow \mathsf{Gen}(1^\lambda,\mathsf{lossy})$*,*

$$\mathsf{Enc}(\widetilde{\mathsf{pk}},0) \approx_s \mathsf{Enc}(\widetilde{\mathsf{pk}},1)$$

where the randomness is over the coins of the Enc *algorithm.*

Lossy encryption schemes can be constructed from various number theoretic assumptions like Quadratic Residuosity[36] [GM82], Decisional Diffie Hellman [NP01, PVW08, BHY09] and standard lattice assumptions (e.g. LWE) [Reg05, PVW08].

Assume that $(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ is a lossy encryption scheme. We show that Assumption 6.2 holds. In fact, lossy encryption scheme implies the stronger Assumption 3.5.[37] The language $\mathcal{L}$ and its associated distributions are defined as follows:

- $\mathcal{L}$ consists of all possible public-keys that can be generated by Gen in the real mode, i.e., $\mathcal{L} = \left\{\mathsf{pk}\colon \exists \mathsf{sk}, (\mathsf{pk},\mathsf{sk}) \in \mathrm{Supp}(\mathsf{Gen}(1^\lambda,\mathsf{real}))\right\}$.
- $\mathsf{Y}_\mathcal{L}(1^\lambda)$ runs Gen in the real mode, i.e., $\mathsf{Gen}(1^\lambda,\mathsf{real})$.
- $\mathsf{N}_\mathcal{L}(1^\lambda)$ runs Gen in the lossy mode, i.e., $\mathsf{Gen}(1^\lambda,\mathsf{lossy})$.

It immediately follows that $\mathcal{L}$ is cryptographically hard. An SZK proof for $(\mathcal{L},\mathsf{Y}_\mathcal{L},\mathsf{N}_\mathcal{L})$ is presented in Fig. 9.

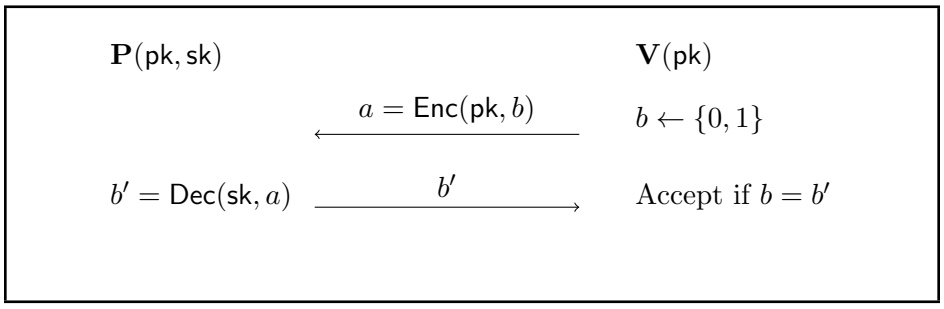

Figure 9: Laconic Proofs from Lossy Encryption

It is again easy to verify that the proof system has perfect completeness (assuming the underlying encryption scheme has no decryption errors), soundness error $1/2 + \mathsf{negl}(\lambda)$, and is honest-verifier zero-knowledge.

[36] A variant of the Goldwasser-Micali scheme can be shown to be lossy. Let the public key be (N, x_0, x_1) where x_0 is a random quadratic residue and x_1 is a random non-residue with Jacobi symbol $+1$. To encrypt σ, output $x_\sigma \cdot r^2$ where $r \leftarrow \mathbb{Z}_N$. The corresponding lossy key would be $(N, \widetilde{x_0}, \widetilde{x_1})$ where both $\widetilde{x_0}$ and $\widetilde{x_1}$ are random quadratic residues.

[37] As a matter of fact, it implies an SZK *proof*, rather than just an argument.

A.2 Learning Parities with Noise

The problem of learning parities with noise (LPN) (or, equivalently, of decoding random linear codes) has found extensive use in cryptography [Pie12]. The hardness of a variant of this problem was used by Alekhnovich [Ale03] to construct a public-key encryption scheme. This hardness assumption is paraphrased below. Let $m = 2n$ and for any $\delta \in [0,1]$, let $\chi_{m,\delta}$ represents the uniform distribution over vectors in $\{0,1\}^m$ of Hamming weight m^δ.

Recall that two distributions are $(\mathsf{poly}, \epsilon)$-hard to distinguish if no polynomial-time algorithm has advantage more than ϵ in distinguishing between them.

Assumption A.2 (LPN)**.** *There is a constant $\delta < 1/2$ such that the following two distributions are $\left(\mathsf{poly}, \frac{1}{n^{\Omega(1)}}\right)$-hard to distinguish:*

- $(\mathbf{A}, \mathbf{As} + \mathbf{e})$*, where* $\mathbf{A} \leftarrow \{0,1\}^{m \times n}$*,* $\mathbf{s} \leftarrow \{0,1\}^n$*, and* $\mathbf{e} \leftarrow \chi_{m,\delta}$*.*
- $(\mathbf{A}, \mathbf{u})$*, where* $\mathbf{A} \leftarrow \{0,1\}^{m \times n}$ *and* $\mathbf{u} \leftarrow \{0,1\}^m$*.*

Let δ be the constant that is promised by the above assumption. The language $\mathcal{L}$ and its associated distributions are defined as follows:

- $\mathcal{L}$ consists of all pairs $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ where $\mathbf{e}$ has Hamming weight n^δ. Notice that $\mathcal{L}$ is in NP because the vector $\mathbf{e}$ serves as a witness that there is an $\mathbf{s}$ such that $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ is contained in $\mathcal{L}$.
- $\mathsf{Y}_\mathcal{L}(1^n)$ picks random $\mathbf{A} \leftarrow \{0,1\}^{m \times n}$, $\mathbf{s} \leftarrow \{0,1\}^n$, and $\mathbf{e} \leftarrow \chi_{m,\delta}$, and outputs $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ as the instance and $\mathbf{e}$ as the corresponding witness.
- $\mathsf{N}_\mathcal{L}(1^n)$ picks random $\mathbf{A} \leftarrow \{0,1\}^{m \times n}$, $\mathbf{u} \leftarrow \{0,1\}^m$ and outputs the instance $(\mathbf{A}, \mathbf{u})$.

Our choice of the language and distributions above is such that the LPN assumption immediately implies that $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ is $\left(\mathsf{poly}, \frac{1}{n^{\Omega(1)}}\right)$-hard. An ASZK argument for $(\mathcal{L}, \mathsf{Y}_\mathcal{L}, \mathsf{N}_\mathcal{L})$ is presented in Figure 10.

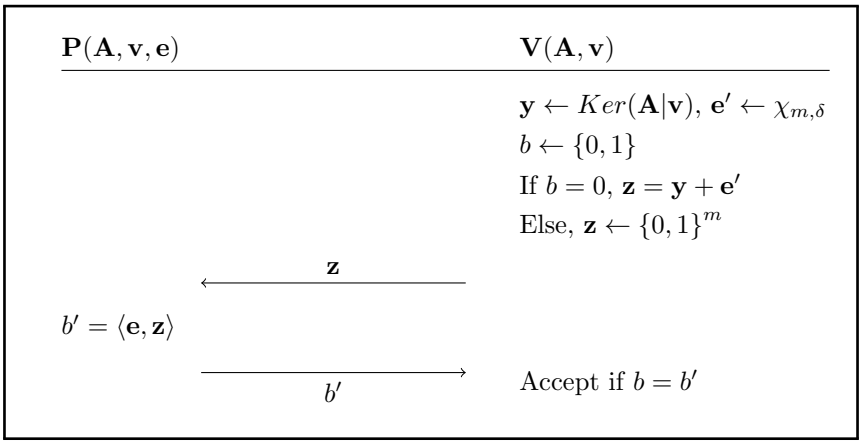

Figure 10: ASZK argument for LPN

The various properties required of this protocol are shown as follows:

– *Completeness:* If $\mathbf{v} = \mathbf{A}\mathbf{s} + \mathbf{e}$ for some $\mathbf{s}$ and $\mathbf{e}$ of Hamming weight n^δ, then $\mathbf{y} \in Ker(\mathbf{A}|\mathbf{v})$ implies that $\langle \mathbf{y}, \mathbf{e} \rangle = 0$. So, in the case $b = 0$, it holds that $\langle \mathbf{e}, \mathbf{z} \rangle = \langle \mathbf{e}, \mathbf{y} \rangle + \langle \mathbf{e}, \mathbf{e}' \rangle = \langle \mathbf{e}, \mathbf{e}' \rangle$. As both $\mathbf{e}$ and $\mathbf{e}'$ have Hamming weight $n^\delta = o(m^{1/2})$ and are chosen at random, the probability that $\langle \mathbf{e}, \mathbf{e}' \rangle = 1$ is $o(1)$. On the other hand, when $b = 1$, $\langle \mathbf{e}, \mathbf{z} \rangle$ is unbiased. So $\mathbf{P}$ can guess b with constant advantage.

– *Soundness:* If $\mathbf{v}$ is a uniformly random vector, then the matrix $(\mathbf{A}|\mathbf{v})$ is completely random. The kernel of this matrix, in turn, can be written as the span of $(m-n-1)$ uniformly random vectors in $\{0,1\}^m$. That is, in the case $b = 1$, it holds that $\mathbf{z} = \mathbf{y} + \mathbf{e}' = \mathbf{B}\mathbf{s}' + \mathbf{e}'$ for a uniformly random matrix $\mathbf{B} \in \{0,1\}^{m \times (m-n-1)}$. And in the case $b = 0$, $\mathbf{z}$ is a uniformly random vector. These are the two distributions that the LPN assumption says are indistinguishable. So any malicious prover has advantage at most $1/n^{\Omega(1)}$ in guessing b correctly.

– *Honest Verifier Statistical Zero-Knowledge:* The simulator $\mathbf{S}$, on input $(\mathbf{A}, \mathbf{v})$, first runs $\mathbf{V}$, which selects b and $\mathbf{z}$. It then sets the prover's message to also be b, and outputs $(b, \mathbf{z}, b)$ as the transcript. The distance of this distribution from the actual transcript is now exactly the probability that $\mathbf{P}$ does not guess b correctly. Thus the simulation error is the same as the completeness error, which is a constant less than $1/2$.

The simulation error being as large as a constant is insufficient for our constructions, but the error in this case can be made negligible by modifying the protocol as follows. $\mathbf{V}$ picks the bit b and generates, say $\log^2 n$ different z's for this b, with independently chosen $\mathbf{y}$'s and $\mathbf{e}'$'s if necessary. It sends them all over and asks $\mathbf{P}$ to guess b. $\mathbf{P}$ runs as it did in the above protocol for each $\mathbf{z}$, takes the majority of the results and sends that as its guess. This makes the completeness error (and hence simulation error) negligible, while still keeping the soundness error below $1/2 + 1/n^{\Omega(1)}$.

A.3 Assumptions from [ABW10]

Applebaum et. al. [ABW10] construct three public-key encryption schemes based on the average-case hardness of various combinatorial problems. They formulate three assumptions and use different combinations and instantiations of these in their constructions. We show that in two out of these three cases the set of assumptions used implies Assumption 6.2.

We shall state the assumptions from [ABW10] with some admissible simplifications for ease of illustration.[38]

The first assumption, called LLIN, concerns solving a noisy system of linear equations. Given integers m, n and d, let $M_{m,n,d}$ denote the distribution over matrices in $\{0,1\}^{m \times n}$ sampled by picking each row to be a random vector in $\{0,1\}^n$ of Hamming weight d. For $\epsilon \in [0,1]$, denote by Ber_ϵ^m the distribution over vectors in $\{0,1\}^m$ where each bit is set to 1 with probability ϵ. The assumption LLIN, parametrised by m, d, and ϵ, which we will later instantiate as functions of n (the security parameter), is as below.

Assumption A.3 ($\mathsf{LLIN}(m, d, \epsilon)$)**.** *Consider the $m \times n$ matrix $\mathbf{A} \leftarrow M_{m,n,d}$, vectors $\mathbf{s}, \mathbf{v} \leftarrow \{0,1\}^n$, $\mathbf{e} \leftarrow Ber_\epsilon^m$, and bit $b \leftarrow \{0,1\}$. There is a constant $\mu < 1/2$ such that the following distributions are (poly, μ)-hard to distinguish:*

[38] For Assumption A.3, stated below, Applebaum *et al.* actually assume the hardness of a related search problem and show, by means of a non-trivial reduction, that the assumption stated here is implied by it, and proceed to use this simpler assumption in their constructions.

- $(\mathbf{A}, \mathbf{As} + \mathbf{e}, \mathbf{v}, \langle \mathbf{v}, \mathbf{s} \rangle)$
- $(\mathbf{A}, \mathbf{As} + \mathbf{e}, \mathbf{v}, b)$

Typically, a matrix from $M_{m,n,d}$ will be such that any small set of rows is linearly independent. In order to state the next assumption, we will consider picking a matrix from $M_{m,n,d}$ and planting a small linear dependency among its rows. Let $M^q_{m,n,d}$ be the distribution over $\{0,1\}^{m \times n} \times \{0,1\}^m$ that is sampled as follows:

1. Sample matrices $\mathbf{A}$ from $M_{m,n,d}$, and $\mathbf{B}$ from $M_{q,q/3,d}$.
2. Pick a random $q \times n$ sub-matrix of $\mathbf{A}$ that contains its last row. Call this sub-matrix $\mathbf{S}$.
3. Embed the columns of $\mathbf{B}$ as a randomly chosen subset of $q/3$ columns in $\mathbf{S}$, and set all other entries in $\mathbf{S}$ to 0. Call the matrix thus obtained $\hat{\mathbf{A}}$.
4. Notice that because of the dimensions of $\mathbf{B}$, there is a vector $\mathbf{t}_{\hat{\mathbf{A}}}$ such that $\mathbf{t}^T_{\hat{\mathbf{A}}} \hat{\mathbf{A}} = \mathbf{0}$, that has Hamming weight at most $q/3$, and its last coordinate is 1. This is the $\mathbf{t}_{\hat{\mathbf{A}}}$ expresses the last row of the embedded $\mathbf{B}$ as a linear combination of its other rows.
5. Output $(\hat{\mathbf{A}}, \mathbf{t}_{\hat{\mathbf{A}}})$.

The next assumption, called DUE, states that this planting of a small linear dependence cannot be detected by polynomial-time algorithms.

Assumption A.4 ($\mathsf{DUE}(m, d, q)$)**.** *There is a small constant $\alpha \ll 1/2$ such that the following distributions are are (poly, α)-hard to distinguish:*

- $\mathbf{A} \leftarrow M_{m,n,d}$
- $\hat{\mathbf{A}}$, *where* $(\hat{\mathbf{A}}, \mathbf{t}_{\hat{\mathbf{A}}}) \leftarrow M^q_{m,n,d}$

The first combination of assumptions we will consider that was used in [ABW10] is that there exists a constant d and functions $m(n) = O(n)$, $q(n) = o(n)$, and $\epsilon(n) = o(1/q(n))$ such that both $\mathsf{LLIN}(m, d, \epsilon)$ and $\mathsf{DUE}(m, d, q)$ are true. In order to relate this to Assumption 6.2, we formulate the following language and distributions:

– $\mathcal{L}$ consists of $\mathbf{A}$ such that there is a set of at most q rows that are linearly dependent. $\mathcal{L}$ is in NP because such a set of rows acts as a witness.

– $\mathsf{Y}_{\mathcal{L}}(1^n)$ is $M^q_{m,n,d}$, with $\hat{\mathbf{A}}$ as the instance and $\mathbf{t}_{\hat{\mathbf{A}}}$ as the witness.

– $\mathsf{N}_{\mathcal{L}}(1^n)$ is $M_{m,n,d}$.

By our choice of the distributions, $\mathsf{DUE}(m, d, q)$ immediately implies that $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is (poly, α)-hard. An ASZK argument for $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is presented in Figure 11.

The various properties required of this protocol are shown as follows:

– *Completeness:* If the instance $\mathbf{A}$ happens to be drawn from $\mathsf{Y}_{\mathcal{L}}$ then, using the fact that $\mathbf{t}^T_{\mathbf{A}} \mathbf{A} = \mathbf{0}$, and that the last bit of $\mathbf{t}_{\mathbf{A}}$ is 1, we have the following relations:

$$\langle \mathbf{t}_{\mathbf{A}}, \mathbf{z} \rangle = \langle \mathbf{t}_{\mathbf{A}}, \mathbf{As} + \mathbf{e} \rangle + b = \langle \mathbf{t}_{\mathbf{A}}, \mathbf{e} \rangle + b$$

As each entry in $\mathbf{e}$ is 1 with probability ϵ, with all but negligible probability, the Hamming weight of $\mathbf{e}$ is at most $100 \cdot m \cdot \epsilon = o(m/q)$. As the Hamming weight of $\mathbf{t}_{\mathbf{A}}$ is at most q, the probability that $\langle \mathbf{t}_{\mathbf{A}}, \mathbf{e} \rangle = 1$ is $o(1)$. Thus, with all but $o(1)$ probability, $\mathbf{P}$ guesses b correctly.

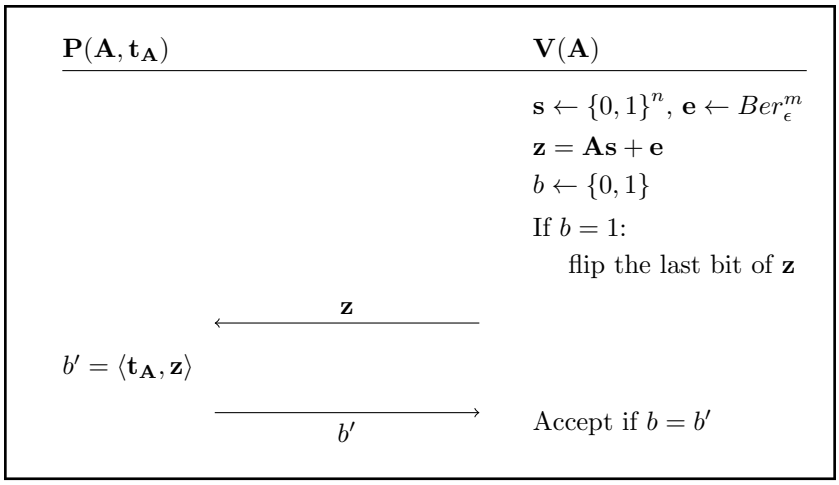

Figure 11: ASZK argument for LLIN+DUE

– *Soundness:* The assumption $\mathsf{LLIN}(m-1, d, \epsilon)$, when applied with the last row of the instance $\mathbf{A}$ as the vector $\mathbf{v}$ there, immediately implies that the last bit of $\mathbf{z}$ is (poly, μ)-hard to distinguish from random. Thus, a malicious prover cannot guess b correctly with advantage more than this μ promised by the assumption.

– *Statistical Zero-Knowledge:* The simulator $\mathbf{S}$, on input $\mathbf{A}$, first runs $\mathbf{V}$, which selects $\mathbf{s}$, $\mathbf{e}$, b, and $\mathbf{z}$. It then sets the prover's message to also be b, and outputs $(\mathbf{s}, \mathbf{e}, b, \mathbf{z}, b)$ as the transcript. The distance of this distribution from the actual transcript is now exactly the probability that $\mathbf{P}$ does not guess b correctly. Thus the simulation error is the same as the completeness error, which is $o(1)$.

Having shown that Assumption 6.2 is implied by the above combination of LLIN and DUE, we next introduce the other assumption used in [ABW10], called DSF.

Given any function $f : \{0,1\}^d \to \{0,1\}$ and a matrix $\mathbf{A} \in \{0,1\}^{m \times n}$, each of whose rows has Hamming weight d, we define the composite function $f_{\mathbf{A}} : \{0,1\}^n \to \{0,1\}^m$ where the i^{th} output bit of $f_{\mathbf{A}}(x)$ is obtained by evaluating f on the bits of x corresponding to the positions in the i^{th} row of A that are 1.

Assumption A.5 ($\mathsf{DSF}(m, d, \epsilon)$)**.** *There exists a function $f : \{0,1\}^d \to \{0,1\}$ such that the following distributions are $(\mathsf{poly}, \epsilon)$-hard to distinguish:*

- $(\mathbf{A}, \mathbf{u})$, *where* $\mathbf{A} \leftarrow M_{m,n,d}$, $\mathbf{u} \leftarrow \{0,1\}^m$.
- $(\mathbf{A}, f_{\mathbf{A}}(\mathbf{u}))$, *where* $\mathbf{A} \leftarrow M_{m,n,d}$, $\mathbf{u} \leftarrow \{0,1\}^n$.

The next combination of assumptions used in [ABW10] is that there is a constant d, and functions $m(n) = \omega(n)$, $q(n) = \Theta(\log n)$ and $\epsilon(n) \leq 1/10$ such that $\mathsf{DUE}(m, d, q)$ and $\mathsf{DSF}(m, d, \epsilon)$ are true. Let f be the function that is promised by DSF.

We use the same language $\mathcal{L}$ and distributions $\mathsf{Y}_{\mathcal{L}}$ and $\mathsf{N}_{\mathcal{L}}$ that we did earlier, only with the values of m, d and q that come up here and that in $\mathsf{Y}_{\mathcal{L}}$, instead of $\mathbf{t_A}$, we use the set S of rows where the matrix $\mathbf{B}$ was embedded when sampling $M^q_{m,n,d}$ as witness. $\mathsf{DUE}(m, d, q)$ again implies that $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is (poly, α)-hard. An ASZK argument for $(\mathcal{L}, \mathsf{Y}_{\mathcal{L}}, \mathsf{N}_{\mathcal{L}})$ is presented in Figure 12.

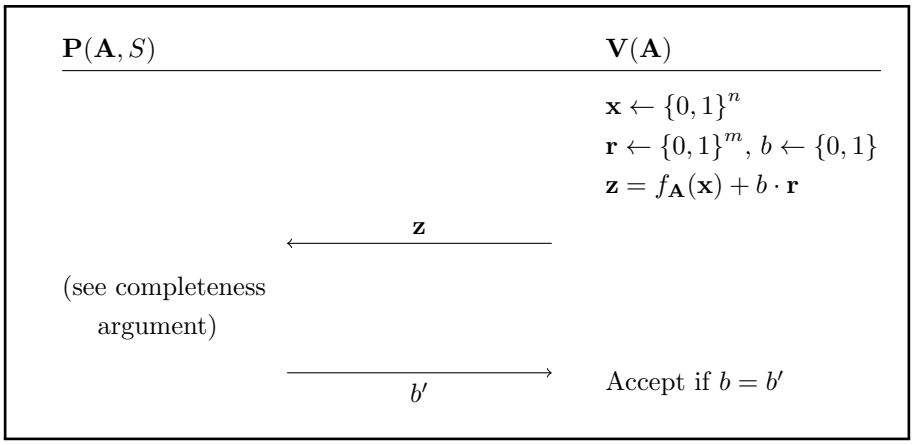

Figure 12: ASZK argument for DSF+DUE

The various properties required of this protocol are shown as follows:

– *Completeness:* The prover $\mathbf{P}$, since it knows S, knows a set of q bits of the output of $f_{\mathbf{A}}$ that depend on less than $q/3 = \Theta(\log n)$ bits of its input. It goes through all possible settings of these bits and checks whether the respective values of all these output bits of $f_{\mathbf{A}}$ agree with the corresponding bits in z. If it finds a setting where they do agree, it sets $b' = 0$, else $b' = 1$.

When $b = 0$, $\mathbf{P}$ will always find that $b' = 0$. When $b = 1$, for any value of x, there are at most $2^{q/3} \cdot 2^{m-q}$ values of r that will make $\mathbf{P}$ guess $b' = 0$. Thus, except with probability at most $2^{-2q/3} = o(1)$, $\mathbf{P}$ will guess $b' = 0$.

– *Soundness:* The assumption $\mathsf{DSF}(m, d, \epsilon)$ immediately implies that a malicious prover cannot guess b correctly except with advantage ϵ.

– *Statistical Zero-Knowledge:* The simulator $\mathbf{S}$, on input $\mathbf{A}$, first runs $\mathbf{V}$, which selects $\mathbf{x}$, $\mathbf{r}$, b, and $\mathbf{z}$. It then sets the prover's message to also be b, and outputs $(\mathbf{x}, \mathbf{r}, b, \mathbf{z}, b)$ as the transcript. The distance of this distribution from the actual transcript is now exactly the probability that $\mathbf{P}$ does not guess b correctly. Thus the simulation error is the same as the completeness error, which is $o(1)$.

B Missing Proofs

B.1 Proving Lemma 5.7

In this section we prove Lemma 5.7, restated below for convenience.

Lemma 5.7 (Leftover Hash Lemma for ε-Smooth Min-Entropy (c.f., [RW05, Theorem 1]))**.** *Let $\mathcal{H} = \{h\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of universal hash functions. Then, for any jointly distributed random variables X and Y, such that X is distributed over $\{0,1\}^n$, it holds that*

$$\mathrm{SD}\Big(\big(H(X), H, Y\big), \big(U_m, H, Y\big)\Big) \leq \varepsilon + \frac{1}{2} \cdot \sqrt{2^{-\mathrm{H}^{\varepsilon}_{\infty}(X|Y)} \cdot 2^m}\ ,$$

where $H \leftarrow \mathcal{H}$ and U_m is distributed uniformly over $\{0,1\}^m$.

The proof of Lemma 5.7 is similar to that of [DORS08, Lemma 2.4] which considers a different notion of pseudo min-entropy which they call "Average Min-Entropy".

Proof. Fix $\varepsilon > 0$ and let E be the event such that $\Pr[E] \geq 1 - \varepsilon$ and

$$\mathrm{H}^{\varepsilon}_{\infty}(X|Y) = \min_{y \in \mathrm{Supp}(Y)} \min_{x \in \mathcal{X}} \log \frac{1}{\Pr[(X = x) \wedge E \mid Y = y]}.$$

Let I_E be an indicator random variable for whether the event E occurred and let $X_{y,b} = (X|Y = y, I_E = b)$. Then,

$$\begin{aligned} \mathrm{SD}\Big(\big(H(X), H, Y\big), \big(U_m, H, Y\big)\Big) &\leq \mathrm{SD}\Big(\big(H(X), H, Y, I_E\big), \big(U_m, H, Y, I_E\big)\Big) \\ &= \mathop{\mathrm{E}}_{(y,b) \leftarrow (Y, I_E)}\Big[\mathrm{SD}\Big(\big(H(X_{y,b}), H\big), \big(U_m, H\big)\Big)\Big] \\ &\leq \Pr[E] \cdot \mathop{\mathrm{E}}_{y \leftarrow Y|E}\Big[\mathrm{SD}\Big(\big(H(X_{y,1}), H\big), \big(U_m, H\big)\Big)\Big] + \Pr[\neg E], \end{aligned} \tag{29}$$

where the first inequality follows from data-processing for statistical distance. At this point we would like to use the original leftover-hash lemma, stated next.

Lemma B.1 ([HILL99, Lemma 4.8])**.** *Let $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of universal hash functions. Then, for any random variable X distributed over $\{0,1\}^n$, and the random variable $H \leftarrow \mathcal{H}$, it holds that*

$$\mathrm{SD}\Big(\big(H(X), H\big), \big(U_m, H\big)\Big) \leq \frac{1}{2} \cdot \sqrt{2^{-\mathrm{H}_{\infty}(X)} \cdot 2^m},$$

where U_m is distributed uniformly over $\{0,1\}^m$.

Plugging Lemma B.1 into Eq. (29) yields that

$$\begin{aligned} \mathrm{SD}\Big(\big(H(X), H, Y\big), \big(U_m, H, Y\big)\Big) &\leq \Pr[E] \cdot \mathop{\mathrm{E}}_{y \leftarrow Y|E}\left[\frac{1}{2} \cdot \sqrt{2^{-\mathrm{H}_{\infty}(X_{y,1})} \cdot 2^m}\right] + \varepsilon \\ &\leq \frac{1}{2} \cdot \sqrt{2^m \cdot \Pr[E] \cdot \mathop{\mathrm{E}}_{y \leftarrow Y|E}\left[2^{-\mathrm{H}_{\infty}(X_{y,1})}\right]} + \varepsilon, \end{aligned} \tag{30}$$

where the last inequality follows from Jensen's inequality and since square-root is concave.

Let $y \in \mathrm{Supp}(Y|E)$. Then,

$$\begin{aligned} \mathrm{H}_{\infty}(X_{y,1}) &= \mathrm{H}_{\infty}(X|Y = y, E) \\ &= \min_{x \in \mathcal{X}} \log \frac{1}{\Pr[X = x|Y = y, E]} \\ &= \min_{x \in \mathcal{X}} \log \frac{\Pr[E|Y = y]}{\Pr[X = x, E|Y = y]} \\ &= \log \Pr[E|Y = y] + \min_{x \in \mathcal{X}} \log \frac{1}{\Pr[X = x, E|Y = y]} \\ &\geq \log \Pr[E|Y = y] + \min_{y \in \mathrm{Supp}(Y)} \min_{x \in \mathcal{X}} \log \frac{1}{\Pr[X = x, E|Y = y]} \\ &= \log \Pr[E|Y = y] + \mathrm{H}^{\varepsilon}_{\infty}(X|Y). \end{aligned} \tag{31}$$

Plugging Eq. (31) into Eq. (30) yields that

$$\begin{aligned}\mathrm{SD}\Big(\big(H(X),H,Y\big),\big(U_m,H,Y\big)\Big) &\leq \frac{1}{2}\cdot\sqrt{2^m\cdot\Pr[E]\cdot\mathop{\mathrm{E}}_{y\leftarrow Y|E}\left[\frac{1}{\Pr[E|Y=y]}\cdot 2^{-\mathrm{H}^{\varepsilon}_{\infty}(X|Y)}\right]}+\varepsilon \quad (32)\\ &= \frac{1}{2}\cdot\sqrt{2^{-\mathrm{H}^{\varepsilon}_{\infty}(X|Y)}\cdot 2^m\cdot\mathop{\mathrm{E}}_{y\leftarrow Y|E}\left[\frac{\Pr[E]}{\Pr[E|Y=y]}\right]}+\varepsilon.\end{aligned}$$

Finally, noting that

$$\begin{aligned}\mathop{\mathrm{E}}_{y\leftarrow Y|E}\left[\frac{\Pr[E]}{\Pr[E|Y=y]}\right] &= \sum_{y\in\mathrm{Supp}(Y|E)}\Pr[Y=y|E]\cdot\frac{\Pr[E]}{\Pr[E|Y=y]}\\ &= \sum_{y\in\mathrm{Supp}(Y|E)}\frac{\Pr[Y=y,E]}{\Pr[Y=y,E]/\Pr[Y=y]}\\ &= \sum_{y\in\mathrm{Supp}(Y|E)}\Pr[Y=y]\\ &\leq 1\end{aligned}$$

completes the proof. □

B.2 Proving Lemma 5.9

In this section we prove Lemma 5.9, restated below for convenience.

Lemma 5.9. *Let $X=\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y=\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of random variables such that X_λ and Y_λ are jointly distributed over $\mathcal{X}_\lambda\times\mathcal{Y}_\lambda$, for every $\lambda\in\mathbb{N}$. Assume X has conditional pseudo entropy at least $\mathrm{H}(X|Y)+n$ given Y, for $n=n(\lambda)\geq 0$.*

Then, for any $\delta=\delta(\lambda)\in[0,\log(|\mathcal{X}_\lambda|)]$ and every polynomial $k=k(\lambda)$, the sequence X^k has conditional pseudo ε-smooth min-entropy at least $k\cdot(\mathrm{H}(X|Y)+n-\delta)$ given Y^k, for $\varepsilon=2^{-\frac{k\cdot\delta^2}{2\log^2(|\mathcal{X}|+3)}}$, where (X^k,Y^k) are the k-fold product repetition of (X,Y).

Proof. Assume toward a contradiction that there exist δ and k such that X^k does not have conditional pseudo ε-smooth min-entropy at least $k\cdot(\mathrm{H}(Y|X)+n-\delta)$ given Y^k. Namely, there exists $c>0$ and an λ^c-time algorithm D such that for every $\{Z_\lambda\}_{\lambda\in\mathbb{N}}$ over $\left\{\mathcal{Y}_\lambda^{k(\lambda)}\right\}_{\lambda\in\mathbb{N}}$ with $\mathrm{H}^{\varepsilon}_{\infty}\left(Z_\lambda|Y_\lambda^{k(\lambda)}\right)\geq k(\lambda)\cdot\Big(\mathrm{H}(X_\lambda|Y_\lambda)+n(\lambda)-\delta(\lambda)\Big)-1/\lambda^c$ for large enough $\lambda\in\mathbb{N}$, there exists an infinite index set $\mathcal{I}\subseteq\mathbb{N}$ such that for every $\lambda\in\mathcal{I}$ it holds that

$$\left|\Pr\left[\mathsf{D}\left(1^\lambda,X_\lambda^{k(\lambda)},Y_\lambda^{k(\lambda)}\right)=1\right]-\Pr\left[\mathsf{D}\left(1^\lambda,Z_\lambda,Y_\lambda^{k(\lambda)}\right)=1\right]\right|>\frac{1}{\lambda^c}. \quad (33)$$

We use D to show that X does not have high pseudoentropy given Y, in contradiction to our assumption. Consider the following algorithm for the latter task, and recall that such algorithm also has access to an oracle that outputs three samples (one from X, one from Y and one from a distribution that has high ε-smooth min-entropy; see ahead).

$\widehat{\mathsf{D}}$ on input $\big(1^\lambda,x,y\big)$:
Oracle: O outputs 3 values
Operation:

1. Set $k = k(\lambda)$.
2. Make k calls to the oracle O and let (x_j, y_j, z_j) be the oracle's output for the j'th call.
3. Sample $i \leftarrow [k]$, and set $\mathbf{y} = (y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_k)$ and $\mathbf{h} = (x_1, \ldots, x_{i-1}, x, z_{i+1}, \ldots, z_k)$.
4. Output $\mathsf{D}(1^\lambda, \mathbf{h}, \mathbf{y})$.

We show that $\widehat{\mathsf{D}}$ distinguishes between (X, Y) and (Z, Y), for any Z that has sufficiently high pseudoentropy given Y.

Let $c' > 0$ be a constant to be determined by the analysis and let $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of random variables over $\{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ jointly distributed with X, Y such that $\mathrm{H}(Z_\lambda|Y_\lambda) \geq \mathrm{H}(X_\lambda|Y_\lambda) + n(\lambda) - 1/\lambda^{c'}$ for large enough $\lambda \in \mathbb{N}$. Fix a sufficiently large value $\lambda \in \mathcal{I}$. To avoid cluttering, we omit λ from the notation.

By a standard hybrid argument, it holds that[39]

$$\left|\Pr\left[\widehat{\mathsf{D}}^{O_{X',Y',Z'}}(X,Y) = 1\right] - \Pr\left[\widehat{\mathsf{D}}^{O_{X',Y',Z'}}(Z,Y) = 1\right]\right| \geq \frac{1}{k} \cdot \left|\Pr\left[\mathsf{D}\left(X^k, Y^k\right) = 1\right] - \Pr\left[\mathsf{D}\left(Z^k, Y^k\right) = 1\right]\right|. \tag{34}$$

By Theorem 5.6, it holds that

$$\mathrm{H}^\varepsilon_\infty(Z^k|Y^k) \geq k \cdot \Big(\mathrm{H}(X|Y) + n - 1/\lambda^{c'} - \delta\Big) = k \cdot \Big(\mathrm{H}(X|Y) + n - \delta\Big) - k/\lambda^{c'},$$

where $\varepsilon = 2^{-\frac{k\cdot\delta^2}{2\log^2(|\mathcal{X}|+3)}}$.

Let $c'' > 0$ be the minimal integer such that $k/\lambda^{c''} \leq 1/\lambda^c$. Eqs. (33) and (34) now yield that

$$\left|\Pr\left[\widehat{\mathsf{D}}^{O_{X',Y',Z'}}(X,Y) = 1\right] - \Pr\left[\widehat{\mathsf{D}}^{O_{X',Y',Z'}}(Z,Y) = 1\right]\right| > \frac{1}{k\cdot\lambda^c} \geq \frac{1}{\lambda^{c''}}.$$

Finally, since $k = \mathsf{poly}(\lambda)$, there exists $\hat{c} > 0$ such that $\widehat{\mathsf{D}}$'s running time is at most $\lambda^{\hat{c}}$. Set $c' = \max\{c'', \hat{c}\}$. We have that X does not have $(\lambda^{c'}, 1/\lambda^{c'})$-conditional pseudoentropy at least $\mathrm{H}(X|Y) + n - 1/\lambda^{c'}$ given Y, a contradiction to pseudoentropy of X given Y. □

[39] Recall that the oracle $O_{X',Y',Z'}$ returns random samples (jointly) distributed according to (X, Y, Z). See Definition 2.15.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il