Chapter 1

Introduction

One of the core goals of cryptography is to be able to offer security and privacy without sacrificing functionality. To do this, cryptographers define notions of both correctness and security. We can then build systems that we prove satisfy both of these definitions. Correctness states that the system provides the functionality we want, while security is essentially the notion that we can effectively hide information from an adversary. Depending on the notion of security, this adversary may be all powerful (information-theoretic or computationally unbounded), may run only in polynomial time (computationally bounded), or even bounded by a specific polynomial runtime like $O(n^2)$ (a fine-grained adversary). This thesis will focus on realizing three different functionalities, each one hiding different information, and three different notions of security.

In the first part of this thesis, we discuss results in the area of Topology Hiding Computation (THC). THC is a generalization of secure multiparty computation. In this model, parties are in an incomplete but connected communication network, each party with its own private inputs. The functionality these parties want to realize is evaluating some function (computation) over all of their inputs. On the security and privacy side, these parties want to hide both their private inputs and who their neighbors are. So, the goal is for these parties to run a protocol, communicating only with their neighbors, so that by the end of the protocol, they learn the output of the function on their inputs and *nothing else*, including information about what the communication network looks like other than their own neighborhood.

It turns out that THC is a difficult notion to realize, mired in impossibility results. We circumvent two impossibility results in THC in two different ways. First, there is an impossibility result for THC stating that it is impossible to design THC against adversaries that can “turn off” parties during the protocol [MOR15a]. In essence, an adversary that has this power can always learn something about the graph. So, instead of designing a protocol that would attempt to leak no information, we designed a protocol to limit the amount of information leaked as much as possible. We ran into similar impossibility results when considering a model more similar to the real-world: channels between parties now have unknown delays on them, and an adversary may be able to control the delay. We had to find a model that mirrored this real-world complication, and this was also not impossible to achieve.

In the second part of the thesis, we focus on Adversarially Robust Property Preserving Hashes (PPH). PPHs are a generalization of collision resistant hash functions (CRHFs). Recall that a CRHF is a family of hash functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ that is compressing ($m < n$), and has the property that no Probabilistic Polynomial-Time (PPT) adversary given h can find two inputs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$ except with negligible probability. Looking at CRHFs from a slightly different perspective, their functionality is to preserve the equality predicate between two inputs while compressing them, preventing an adversary from coming up with inputs where the equality predicate is not preserved (even though these inputs exist, they are hard to find). A PPH is also a compressing function that preserves some other property. For example, we focus on the property of "gap-hamming" distance: if two inputs are close in hamming distance, then their hashes are also close, and if two inputs are far, their hashes are also far. Note that this example is of a promise predicate since we do not care about inputs that are neither close nor far.

Property preserving hashes were a new cryptographic primitive that we designed, and so it was important to understand what was possible and what was not in our new model. We found strong connections between One-Way Communication Complexity (OWC complexity) and our primitive. Fortunately for us, OWC is a rich, well-studied area of complexity. In many cases, we could not directly port OWC results to our setting, but we could use their proof techniques and get lower bounds. Once we had these lower bounds, we had a much better sense of what was and was not possible, and could construct PPHs more easily.

In the final chapter, for a different perspective, we design a public-key cryptography functionality against weak adversaries. This functionality allows for a party to publish a public key while they keep the secret key, and any other party to use that public key to encrypt a message that only the party with the secret key can decrypt, *hiding* that message from any eavesdroppers. Unlike in standard models for cryptography, however, our eavesdroppers much less powerful: their runtime is bounded by an explicit polynomial instead of "probabilistic polynomial-time", e.g. can only run in time $O(n^{100})$. This, of course, is only interesting as a cryptographic notion if the adversary has the same or more time to run than the honest parties. So, we get a notion of cryptography we call *fine-grained*: for examples, honest parties might only need to run in $O(n^2)$ time, while any adversary running in time $O(n^{100})$ time still cannot glean any useful information with some probability. Motivating the study of this cryptography is the fact that it does not rely on any of the normal cryptographic assumptions, including $P \neq NP$, $BPP \neq NP$, or even the assumption that one-way functions exist.

However, due to its fine-grained nature many standard cryptographic reductions (like Goldreich-Levin hardcore bits [GL89]) no longer work. And so, in this thesis we demonstrate variations that can work in our setting, including a notion of fine-grained one-way functions, fine-grained hardcore-bits, a fine-grained key exchange, and finally fine-grained public-key cryptography.

1.1 Results

In this thesis, we explore these three different notions of hiding while preserving a functionality:

1. hide private inputs and network topology while preserving computation,
2. hide as much information as possible while preserving a property between inputs,
3. and hide messages from a weak adversary while preserving the communication and fine-grained runtime.

Topology-Hiding Computation To address the first perspective, we explore the realm of topology-hiding computation (THC). THC is a generalization of secure multiparty computation (MPC), where we hide not only each party's input, but also the communication graph; parties know who their neighbors are, but learn nothing else about the structure of the graph. In 2017, we showed that THC is possible against a very weak adversary [ALM17b], but there were still many open questions surrounding the nature of THC.

- **Fail-stop Adversaries.** Fail-stop adversaries can turn off parties during the protocol. As shown by Moran, Orlav, and Richelson in 2015 [MOR15a], it is impossible to achieve perfect topology hiding when giving the adversary this power. The next question is how to get around such an impossibility while still providing meaningful privacy and security. Since the adversary was going to learn something in any protocol we could devise, we decided to use a model that would incorporate this leakage.

  Our results here were a definition of security that included access to a "leakage oracle" and a protocol that used this oracle in the security proof. This oracle would spit out yes or no answers to specific queries, leaking one bit of information at a time. In the security reduction of our protocol, a simulator would only need to call this oracle at most once, leaking at most one bit of information to an adversary. Concretely, we could bound the probability that the simulator would have to call the leakage oracle, meaning that we would leak only a polynomial-fraction of a bit. Our protocol ran in $\tilde{O}(\kappa n^4/\rho)$ rounds, where κ is the security parameter, n is the number of parties, and ρ is the probability that we leak a bit.

  These results are detailed in our paper at TCC 2018 [LZM$^+$18].

- **Probabilistic Delay Model.** This model was the result of first trying to build THC in a purely asynchronous setting, only to prove that it was impossible to do so. So, to get around this without just reducing ourselves to the fail-stop case, we designed a new model that more closely resembled an asynchronous, real-world setting.

More formally, each edge between parties in the Probabilistic Delay Model has its own distribution over delays. Each time a party sends a message to another party, that delay is sampled *independently.* This simulates real world traffic having different delays as it gets sent from one server to another.

Our results also included two new constructions in this model. The first is a construction that works under standard cryptographic assumptions, but only on trees, cycles, or graphs with small circumference. The second is a construction that requires secure hardware boxes/tokens, but works for arbitrary communication graphs. Although this is a very powerful assumption, there are still many challenges involved since an adversarial party can query this hardware at any point with any input.

Adversarially Robust Property Preserving Hashes and One-Way Communication Lower Bounds Addressing the second perspective, we look at a new cryptographic primitive: adversarially robust Property Preserving Hashes (PPH). The inspiration here is two-fold. First, collision-resistant hash functions (CRHFs) are a staple in cryptography: no PPT adversary can produce an $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$ except with negligible probability over h sampled from the CRHF family and coins of the adversary, and hence one can preserve the "equality" predicate on compressed inputs even against adversarially chosen inputs. We want to compress inputs while maintaining some property other than equality between them, even in the presence of adversarially chosen inputs. The properties we considered were those that already had non-robust constructions, in particular locality-sensitive hashing (LSH) [IM98]. An LSH family is a hash function family $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ that compresses inputs ($m < n$) and is *mostly* correct. For example, say we were preserving a gap-ℓ_p norm for a gap between r and cr for some constant $c > 1$ (note that in our case, gap-hamming is equivalent to gap-ℓ_0 norm). Then, we have a threshold τ so that for any pair $x_1, x_2 \in \{0,1\}$

- if $||x_1 - x_2||_p < r$, $||h(x_1) - h(x_2)||_p < \tau$, and
- if $||x_1 - x_2||_p > cr$ then $||h(x_1) - h(x_2)||_p \geq \tau$

with high probability over our choice of h sampled from $\mathcal{H}$.

We can use almost the same language to define PPHs for some property $P : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, except our probability will be negligible and taken over both our sampled hash function and the coins of a PPT adversary. We also generalize the predicate evaluation: instead of using the same predicate on hashed values, we can use an "evaluation function" called Eval. So, even in the presence of (PPT) adversarially chosen values of x_1 and x_2,

- if $P(x_1, x_2) = 0$, then $\mathsf{Eval}(h(x_1), h(x_2)) = 0$, and
- if $P(x_1, x_2) = 1$, then $\mathsf{Eval}(h(x_1), h(x_2)) = 1$

with all but negligible probability. In other words, for any PPT adversary given h sampled from $\mathcal{H}$, the probability that the adversary produces x_1 and x_2 such that $P(x_1, x_2) \neq \mathsf{Eval}(h(x_1), h(x_2))$ is negligible.

The link between robust PPH and LSH was very clear, but less obvious was the link between PPH and one-way communication (OWC): in essence, compressing an input as much as possible for a OWC protocol is akin to compressing an input as much as possible while preserving some property of that input as we would want to do for a hash function. These connections led to the following results, which will be included in the thesis.

- OWC is a rich field with a lot of work detailing lower bounds on the complexity of certain predicates. These lower bounds *almost* directly translated to impossibility results for PPHs. However, we needed to be careful because OWC lower bounds dealt with constant error, where as cryptographers, we care about negligible error. We characterized these OWC lower bounds and how they mapped to PPH lower bounds. We showed that a class of predicates we call "reconstructing predicates" could not have PPHs. This class includes useful predicates like GreaterThan, Index, and ExactHamming.

- With these lower bounds and previous results from LSH in mind, we turned to constructing a PPH for Gap-Hamming: the promise predicate where we can distinguish between pairs of points that are $(1-\epsilon)d$ *close* in hamming distance or $(1+\epsilon)d$ *far*. For all pairs within the gap, we "don't care" how our PPH behaves. We build two constructions for this predicate, each with different drawbacks and benefits. Our first construction relies only on collision-resistant hashing. Our second uses a new assumption related to the hardness of syndrome decoding [AHI+17].

- Rounding out this paper and its myriad of definitions, we demonstrate that even for very simple predicates, we require collision resistance, and even if we weaken our main definition of a robust PPH, we still need one-way functions.

These results were published in ITCS 2019 [BLV19].

In unpublished follow-up research with Cyrus Rashtchian, we explored the intricacies of the CRHF method for constructing gap-hamming PPHs. We fleshed out the relationship between d, the "center" of our gap, and the size of the hash function output. We also discussed how to use this kind of construction to get a gap-ℓ_1-norm PPH and found that with standard methods of transforming ℓ_1 into ℓ_0, it could not work.

Fine-Grained Cryptography and Barriers Addressing the final perspective, this thesis will discuss our work in Fine-Grained cryptography. With my coauthors Lincoln and Vassilevska Williams, we built the first fine-grained key exchange built from fine-grained assumptions. The premise was to explore what cryptography could be like in "Pessiland," a world in which there are no cryptographic one-way functions (which also implies no public key cryptography). We looked at this through the lens of fine-grained complexity, using both assumptions and reduction techniques from that field.

While designing cryptography for this setting, we came across multiple barriers. The first was that some fine-grained problems would not lend themselves to build cryptography without refuting NSETH [CGI+16]. So, we would need to use a fine-grained problem that was not associated with that barrier. Another form of barrier we ran into was worst-case to average-case reductions for problems that would make for good cryptography. Even now, the only worst-case to average-case reductions for fine-grained problems are for counting versions of these problems. Unfortunately, counting-style problems do not lend themselves well to building cryptography, as there are no longer small enough witnesses. Finally, many standard cryptographic reductions no longer work in a fine-grained world, since they take a non-trivial polynomial amount of time, and so a challenge we faced was to ensure all of our reductions were fine-grained and build different types of primitives based off of these restricted reductions.

Despite these difficulties, we achieved the following results:

- Assuming that an average-case version of Zero-k-Clique requires $n^{k-o(1)}$ time, we constructed a non-interactive key exchange that required honest parties to run in $O(N)$ time and an adversary to run in time $\tilde{\Omega}(N^{1.5-\epsilon})$, where ϵ decreases with respect to k.[1]

- Generalize the properties of Zero-k-Clique to construct this key exchange: plantable, list-hard, and splittable.

- Define and construct fine-grained hard-core bits.

This work was published in CRYPTO 2019 [LLW19].

In unpublished follow-up work, we also show how to use another property of Zero-k-Clique to construct a key exchange where the adversary now needs $\Omega(N^{2-\epsilon})$ time. This N^2 gap is the best we are able to do since we base these constructions on Merkle puzzles [BM09].

1.2 Related Works

1.2.1 Related Work to Topology Hiding Computation

Hiding the network topology is more of a concern in the literature on *anonymous communication* [Cha81, RR98, SGR97]. Here, the goal is to hide the identity of the sender and receiver in a message transmission. A classical technique to achieve anonymity is the so-called mix-net technique, introduced by Chaum [Cha81]. Here, *mix* servers are used as proxies which shuffle messages sent between peers to disable an eavesdropper from following a message's path. The onion routing technique [SGR97, RR98] is

[1]The tilde in $\tilde{\Omega}$ ignores any *subpolynomial* factors.

perhaps the most known instantiation of the mix-technique. Another anonymity technique known as *Dining Cryptographers networks*, in short DC-nets, was introduced in [Cha88] (see also [Bd90, GJ04]).

Synchronous Networks. Existing constructions to achieve *topology-hiding communication* over an incomplete network focus mainly on the cryptographic setting for passive corruptions. The first protocol was given in [MOR15b]. Here, the authors provide a feasibility result for a broadcast protocol secure against static, passive corruptions. At a very high level, [MOR15b] uses a series of nested multi-party computations, in which each node is emulated by a secure computation of its neighbor. This emulation then extends to the entire graph recursively. In [HMTZ16a], the authors improve this result and provide a construction that makes only black-box use of encryption and where the security is based on the DDH assumption. However, both results are feasible only for graphs with logarithmic diameter. Topology hiding communication for certain classes of graphs with large diameter was described in [AM17]. This result was finally extended to allow for arbitrary (connected) graphs in [ALM17a].

The fail-stop setting was first considered in [MOR15b] where the authors give a construction for a fail-stop adversary with a very limited corruption pattern: the adversary is not allowed to corrupt any complete neighborhood of a party and is also not allowed an abort pattern that disconnects the graph. In this work, the authors also prove that topology-hiding communication without leakage is impossible if the adversary disconnects the graph. A more recent work, [BBMM18], provides a construction for a fail-stop adversary with one bit/non-negligible leakage but requires that parties have access to secure hardware modules which are initialized with correlated, pre-shared keys.

In the information-theoretic setting, the main result is negative [HJ07]: any MPC protocol in the information-theoretic setting inherently leaks information about the network graph. They also show that if the routing table is leaked, one can construct an MPC protocol which leaks no additional information.

Other Network Models. Katz et al. [KMTZ13] introduce eventual-delivery and channels with a fixed known upper bound. These functionalities implement communication between two parties, where the adversary can set, for each message, the delay after which it is delivered. For reasons stated at the beginning of this section, such functionalities cannot be used directly to model topology-hiding computation. Instead of point-to-point channels we need to model the whole communication network, and we cannot allow the adversary to set the delays. Intuitively, $\mathcal{F}_{\text{NET}}$ implements a number of bounded-delay channels, each of which is modified so that the delay is chosen once and independently of the adversary. If we did not consider hiding the topology, our modified channels would be a stronger assumption.

Cohen et al. [CCGZ16] define different channels with probabilistic delays, for example point-to-point channels (the SMT functionalities) and an all-to-all channel (parallel SMT, or PSMT). However, their PSMT functionality cannot be easily modified to model THC, since the delivery time is sampled once for all parties. One could modify the SMT functionalities and use their parallel composition, but we find our formulation simpler and much better suited for THC.

1.2.2 Related Work to Property Preserving Hashing

1.2.3 Related Work to Fine-Grained Cryptography

1.3 Outline of Thesis

TODO

Chapter 4

Fine-Grained Cryptography

In this chapter, we identify sufficient properties for a fine-grained average-case assumption that imply cryptographic primitives such as fine-grained public key cryptography (PKC). We build a novel cryptographic key exchange using a problem with a small number of relatively weak structural properties, such that if a computational problem satisfies them, our key exchange has provable fine-grained security guarantees, based on the hardness of the problem. We then show that a natural and plausible average-case assumption for the key problem Zero-k-Clique from fine-grained complexity satisfies our properties. We also develop fine-grained one-way functions and hardcore bits even under these weaker assumptions.

Where previous works had to assume random oracles or the existence of strong one-way functions to get a key-exchange computable in $O(n)$ time secure against $O(n^2)$ adversaries (see [Merkle'78] and [BGI'08]), our assumptions seem much weaker. Our key exchange has a similar gap, $O(n)$ secure against $O(n^{1.5-\epsilon})$ adversaries, between the computation of the honest party and the adversary as prior work, while being non-interactive, implying fine-grained PKC. We also show that using more specific properties of Zero-k-Clique, we can build a key exchange computable in $O(n)$ time secure against $O(n^{2-\epsilon})$ adversaries, approaching Merkle's construction guarantees.

This chapter is based on [LLW19].

4.1 Overview

Modern cryptography has developed a variety of important cryptographic primitives, from One-Way Functions (OWFs) to Public-Key Cryptography to Obfuscation. Except for a few more limited information theoretic results [Sha79, CKGS98, RW02], cryptography has so far required making a computational assumption, P $\neq$ NP being a baseline requirement. Barring unprecedented progress in computational complexity, such hardness hypotheses seem necessary in order to obtain most useful primitives. To alleviate this reliance on unproven assumptions, it is good to build cryptography from a variety of extremely different, believable assumptions: if a technique disproves one hypothesis, the unrelated ones might still hold. Due to this, there are many different cryptographic assumptions: on factoring, discrete logarithm, shortest vector in lattices and many more.

Unfortunately, almost all hardness assumptions used so far have the same quite stringent requirements: not only that NP is not in BPP, but that we must be able to efficiently sample polynomially-hard instances whose solution we know. Impagliazzo [Imp95, RR94] defined five worlds, which capture the state of cryptography, depending on which assumptions happen to fail. The three worlds worst for cryptography are Algorithmica (NP in BPP), Heuristica (NP is not in BPP but NP problems are easy on average) and Pessiland (there are NP problems that are hard on average but solved hard instances are hard to sample, and OWFs do not exist). This brings us to our main question.

Can we have a meaningful notion of cryptography even if we live in Pessiland (or Algorithmica or Heuristica)?

This question motivates a weaker notion of cryptography: cryptography that is secure against n^k-time bounded adversaries, for a constant k. Let us see why such cryptography might exist even if P = NP. In complexity, for most interesting computational models, we have time hierarchy theorems that say that there are problems solvable in $O(n^2)$ time (say) that cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$ [HS65, HS66, Tse56]. In fact, such theorems exist also for the average case time complexity of problems [Lev73]. Thus, even if P=NP, there are problems that are hard on average for specific runtimes, i.e. *fine-grained* hard on average. *Can we use such hard problems to build useful cryptographic primitives?*

Unfortunately, the problems from the time hierarchy theorems are difficult to work with, a common problem in the search for unconditional results. Thus, let us relax our requirements and consider hardness assumptions, but this time on the exact running time of our problems of interest. One simple approach is to consider all known constructions of Public Key Cryptography (PKC) to date and see what they imply if the hardness of the underlying problem is relaxed to be $n^{k-o(1)}$ for a fixed k (as it would be in Pessiland). Some of the known schemes are extremely efficient. For instance, the RSA and Diffie-Hellman cryptosystems immediately imply weak PKC if one changes their assumptions to be about polynomial hardness [RSA78, DH06]. However, these cryptosystems have other weaknesses – for instance, they are completely broken in a postquantum world as Shor's algorithm breaks their assumptions in essentially quadratic time [Sho94]. Thus, it makes sense to look at the cryptosystems based on other assumptions. Unfortunately, largely because cryptography has mostly focused on the gap between polynomial and superpolynomial time, most reductions building PKC have a significant (though polynomial) overhead; many require, for example, multiple rounds of Gaussian elimination. As a simple example, the Goldreich-Levin construction for hard-core bits uses n^ω (where $\omega \in [2, 2.373)$ is the exponent of square matrix multiplication [Wil12][Gal14]) time and n calls to the hard-core-bit distinguisher [GL89]. The polynomial overhead of such reductions means that if the relevant problem is only $n^{2-o(1)}$ hard, instead of super-polynomially hard, the reduction will not work anymore and won't produce a meaningful cryptographic primitive. Moreover, reductions with fixed polynomial overheads are no longer composable in the same way when we consider weaker, polynomial gap cryptography. Thus, new, more careful cryptographic reductions are needed.

Ball et al. [BRSV17, BRSV18] recently began to address this issue through the lens of the recently blossoming field of *fine-grained complexity.* Fine-grained complexity is built upon "fine-grained" hypotheses on the (worst-case) hardness of a small number of key problems. Each of these key problems K, has a simple algorithm using a combination of textbook techniques, running in time $T(n)$ on instances of size n, in, say, the RAM model of computation. However, despite decades of research, no $\tilde{O}(T(n)^{1-\epsilon})$ algorithm is known for any $\epsilon > 0$ (note that the tilde suppresses sub-polynomial factors). The fine-grained hypothesis for K is then that K requires $T(n)^{1-o(1)}$ time in the RAM model of computation. Some of the main hypotheses in fine-grained complexity (see [Vas18]) set K to be CNF-SAT (with $T(n) = 2^n$, where n is the number of variables), or the k-Sum problem (with $T(n) = n^{\lceil k/2 \rceil}$), or the All-Pairs Shortest Paths problem (with $T(n) = n^3$ where n is the number of vertices), or one of several versions of the k-Clique problem in weighted graphs. Fine-grained uses fine-grained reductions between problems in a very tight way (see [Vas18]): if problem A has requires running time $a(n)^{1-o(1)}$, and one obtains an $(a(n), b(n))$-fine-grained reduction from A to B, then problem B needs runtime $b(n)^{1-o(1)}$. Using such reductions, one can obtain strong lower bounds for many problems, conditioned on one of the few key hypotheses.

The main question that Ball et al. set out to answer is: *Can one use fine-grained reductions from the hard problems from fine-grained complexity to build useful cryptographic primitives?* Their work produced worst-case to average-case fine-grained reductions from key problems to new algebraic average case problems. From these new problems, Ball et al. were able to construct fine-grained proofs of work, but they were not able to obtain stronger cryptographic primitives such as fine-grained one-way-functions or public key encryption. In fact, they gave a barrier for their approach: extending their approach would falsify the Nondeterministic Strong Exponential Time Hypothesis (NSETH) of Carmosino et al. [CGI$^+$16]. Because of this barrier, one would either need to develop brand new techniques, or use a different hardness assumption.

What kind of hardness assumptions can be used to obtain public-key cryptography (PKC) even in Pessiland?

A great type of theorem to address this would be: for every problem P that requires $n^{k-o(1)}$ time on average, one can construct a public-key exchange (say), for which Alice and Bob can exchange a $\lg(n)$ bit key in time $O(n^{ak})$, whereas Eve must take $n^{(a+g)k-o(1)}$ time to learn Alice and Bob's key, where g is large, and a is small. As a byproduct of such a theorem, one can obtain not just OWFs, but even PKC in Pessiland under fine-grained assumptions via the results of Ball et al. Of course, due to the limitations given by Ball et al. such an ideal theorem would have to refute NSETH, and hence would be at the very least difficult to prove. Thus, let us relax our goal, and ask

What properties are sufficient for a fine-grained average-case assumption so that it implies fine-grained PKC?

If we could at least resolve this question, then we could focus our search for worst-case to average-case reductions in a useful way.

4.1.1 Our Results

Our main result is a fine-grained key-exchange that can be formed from any problem that meets three structural conditions in the word-RAM model of computation. This addresses the question of what properties are sufficient to produce fine-grained Public Key Encryption schemes (PKEs).

For our key exchange, we describe a set of properties, and any problem that has those properties implies a polynomial gap PKE. An informal statement of our main theorem is as follows.

Theorem. [Fine-Grained Key-Exchange (informal)] Let P be a computational problem for which a random instance can be generated in $O(n^g)$ time for some g, and that requires $n^{k-o(1)}$ time to be solved on average for some fixed $k > g$. Additionally, let P have three key structural properties of interest: (1) "plantable": we can generate a random-looking instance, choosing either to have or not to have a solution in the instance, and if there is a solution, we know what/where it is; (2) "average-case list-hard": given a list of n random instances of the problem, returning which one of the instances has a solution requires essentially solving all instances; (3) "splittable": when given an instance with a solution, we can split it in $O(n^g)$ time into two slightly smaller instances that both have solutions.

Then a public key-exchange can be built such that Alice and Bob exchange a $\lg(n)$ bit key in time n^{2k-g}, where as Eve must take $\tilde{\Omega}(n^{3k-2g})$ time to learn Alice and Bob's key.

Notice that as long as there is a gap between the time to generate a random instance and the time to solve an instance on average, there is a gap between $N = n^{2k-g}$ and $n^{3k-2g} = N^{3/2-1/(4(k/g)-2)}$ and the latter goes to $N^{3/2}$, as k/g grows. The key exchange requires no interaction, and we get a *fine-grained* public key cryptosystem. While our key exchange construction provides a relatively small gap between the adversary and the honest parties ($O(N^{1.5})$ vs $O(N)$), the techniques required to prove security of this scheme are novel and the result is generic as long as the three assumptions are satisfied. In fact, we will show an alternate method to achieve a gap approaching $O(N^2)$ in the full version of this paper.

Our main result above is stated formally and in more generality in Theorem 31. We will explain the formal meaning of our structural properties *plantable*, *average-case list-hard*, and *splittable* later.

We also investigate what plausible average-case assumptions one might be able to make about the key problems from fine-grained complexity so that the three properties from our theorem would be satisfied. We consider the Zero-k-Clique problem as it is one of the hardest worst-case problems in fine-grained complexity. For instance, it is known that if Zero-3-Clique is in $O(n^{3-\epsilon})$ time for some $\epsilon > 0$, then both the 3-Sum and the APSP hypotheses are violated [Vas18, WW13]. It is important to note that while fine-grained problems like Zero-k-Clique and k-Sum are suspected to take a certain amount of time in the worst case, when making these assumptions for any constant k does not seem to imply $P \neq NP$ since all of these problems are still solveable in polynomial time.[1]

An instance of Zero-k-Clique is a complete k-partite graph G, where each edge is given a weight in the range $[0, R-1]$ for some integer R. The problem asks whether there is a k-clique in G whose edge weights sum to 0, modulo R. A standard fine-grained assumption (see e.g. [Vas18]) is that in the worst case, for large enough R, say $R \geq 10n^{4k}$, Zero-k-Clique requires $n^{k-o(1)}$ time to solve. Zero-k-Clique has no non-trivial average-case algorithms for natural distributions (uniform for a range of parameters, similar to k-Sum and Subset Sum). Thus, Zero-k-Clique is a natural candidate for an average-case fine-grained hard problem.

Our other contribution addresses an open question from Ball et al.: can a fine-grained one-way function be constructed from worst case assumptions? While we do not fully achieve this, we generate new plausible average-case assumptions from fine-grained problems that imply fine-grained one-way functions.

4.1.2 Previous Works

There has been much prior work leading up to our results. First, there are a few results using assumptions from fine-grained complexity and applying them to cryptography. Second, there has been work with the kind of assumptions that we will be using.

Fine-Grained Cryptography

Ball et al. [BRSV17, BRSV18] produce fine-grained wost-case to average-case reductions. Ball et al. leave an open problem of producing a one-way-function from a worst case assumption. They prove that from some fine-grained assumptions building a one-way-function would falsify NSETH [CGI+16][BRSV17]. We avoid their barrier in this paper by producing a construction of both fine-grained OWFs and fine-grained PKE from an *average-case* assumption.

Fine-Grained Key Exchanges. Fine-grained cryptography is a relatively unexplored area, even though it had its start in the 1970's with Merkle puzzles: the gap between honestly participating in the protocol versus breaking the security guarantee was only quadratic [Mer78]. Merkle originally did not describe a plausible hardness assumption under which the security of the key exchange can be based. 30 years later, Biham, Goren, and Ishai showed how to implement Merkle puzzles by making an assumption of the existence of either a random oracle or an exponential gap one way function [BGI08]. That is, Merkle puzzles were built under the assumption that a one-way function exists which takes time $2^{n(1/2+\delta)}$ to invert for some $\delta > 0$. So while prior work indeed succeeded in building a fine-grained key-exchange, it needed a very

[1]Assuming the hardness of these problems for more general k will imply $P \neq NP$, but that is not the focus of our work.

strong variant of OWFs to exist. It is thus very interesting to obtain fine-grained public key encryption schemes based on a fine-grained assumption (that might even work in Pessiland and below).

Another notion of Fine-Grained Cryptography. In 2016, work by Degwekar, Vaikuntanathan, and Vasudevan [DVV16] discussed fine-grained complexity with respect to both honest parties and adversaries restricted to certain circuit classes. They obtained constructions for some cryptographic primitives (including PKE) when restricting an adversary to a certain circuit class. From the assumption $\mathsf{NC1} \neq \oplus L/\text{poly}$ they show Alice and Bob can be in $AC^0[2]$ while being secure against $\mathsf{NC1}$ adversaries. While [DVV16] obtains some unconditional constructions, their security relies on the circuit complexity of the adversary, and does not apply to arbitrary time-bounded adversaries as is usually the case in cryptography. That is, this restricts the types of algorithms an adversary is allowed to use beyond just how much runtime these algorithms can have. It would be interesting to get similar results in the low-polynomial time regime, without restricting an adversary to a certain circuit class. Our results achieve this, though not unconditionally.

Tight Security Reductions and Fine-Grained Crypto. Another area the world of fine-grained cryptography collides with is that of tight security reductions in cryptography. Bellare et.al. coined the term "concrete" security reductions in [BKR94, BGR95]. Concrete security reductions are parametrized by time (t), queries (q), size (s), and success probability (ϵ). This line of work tracks how a reduction from a problem to a construction of some cryptographic primitive effects the four parameters of interest. This started a rich field of study connecting theory to practical cryptographic primitives (such as PRFs, different instantiations of symmetric encryption, and even IBE for example [BCK96, BDJR97, KW03, BR09]). In fine-grained reductions we also need to track exactly how our adversary's advantage changes throughout our reductions, however, we also track the running time of the honest parties. So, unlike in the concrete security literature, when the hard problems are polynomially hard (perhaps because $P = NP$), we can track the gap in running times between the honest and dishonest parties. This allows us to build one way functions and public key cryptosystems when the hard problems we are given are only polynomially hard.

Similar Assumptions

This paper uses hypotheses on the running times of problems that, while solvable in polynomial time, are variants of natural NP-hard problems, in which the size of the solution is a fixed constant. For instance, k-Sum is the variant of Subset Sum, where we are given n numbers and we need to find exactly k elements that sum to a given target, and Zero-k-Clique is the variant of Zero-Clique, in which we are given a graph and we need to find exactly k nodes that form a clique whose edge weights sum to zero.

With respect to Subset Sum, Impagliazzo and Naor showed how to directly obtain OWFs and PRGs assuming that Subset Sum is hard on average [IN02]. The OWF is $f(\mathbf{a}, \mathbf{s}) = (\mathbf{a}, \mathbf{a} \cdot \mathbf{s})$, where $\mathbf{a}$ is the list of elements (chosen uniformly at random

Paper	Assumptions	Crypto	Runtime	Power of Adversary
[Mer78]	Random Oracles*	Key Exchange	$O(N)$	$O(N^2)$
[BGI08]	Exponentially-Strong OWFs	Key Exchange	$O(N)$	$O(N^2)$
[BRSV18]	WC 3-Sum, OV, APSP, or SETH	Proof of Work	$O(N^2)$	N/A
[This work]	Zero-k-Clique or k-Sum	OWFs, Key Exch. & PKE	$O(N)$ $O(N)$	$O(N^{1+\delta})$ $O(N^{1.5-\delta})$
[DVV16]	$\mathsf{NC1} \neq \oplus L/\text{poly}$	OWFs, and PRGs with sublinear stretch, CRHFs, and PKE	$\mathsf{NC1}$	$\mathsf{NC1}$
	$\mathsf{NC1} \neq \oplus L/\text{poly}$	PKE and CRHFs	$\mathsf{AC}^0[2]$	$\mathsf{NC1}$
	Unconditional	PRGs with poly stretch, Symmetric encryption, and CRHFs	AC^0	AC^0

Figure 4-1: A table of previous works' results in this area. There have been several results characterizing different aspects of fine-grained cryptography. *It was [BGI08] who showed that Merkle's construction could be realized with a random oracle. However, Merkle presented the construction.

from the range R) and $s \in \{0,1\}^n$ represents the set of elements we add together. In addition to Subset Sum, OWFs have also been constructed from planted Clique, SAT, and Learning-Parity with Noise [Lin17, JP00]. The constructions from the book of Lindell and the chapter written by Barak [Lin17] come from a definition of a "plantable" NP-hard problem that is assumed to be hard on average.

Although our OWFs are equivalent to scaled-down, polynomial-time solvable characterizations of these problems, we also formalize the property that allows us to get these fine-grained OWFs (plantability). We combine these NP constructions and formalizations to lay the groundwork for fine-grained cryptography.

In the public-key setting, there has been relatively recent work taking NP-hard problems and directly constructing public-key cryptosystems [ABW10]. They take a problem that is NP-hard in its worst case and come up with an average-case assumption that works well for their constructions. Our approach is similar, and we also provide evidence for why our assumptions are correct.

In recent work, Subset Sum was also shown to directly imply public-key cryptography [LPS10]. The construction takes ideas from Regev's LWE construction [Reg05], turning a vector of subset sum elements into a matrix by writing each element out base q in a column. The subset is still represented by a 0-1 matrix, and error is handled by the lack of carrying digits. It is not clear how to directly translate this construction into the fine-grained world. First, directly converting from Subset Sum to k-Sum just significantly weakens the security without added benefit. More importantly, the security reduction has significant polynomial overhead, and would not apply in a very pessimistic Pessiland where random planted Subset Sum instances can be solved in quadratic time, say.

While it would be interesting to reanalyze the time-complexity of this construction (and others) in a fine-grained way, this is not the focus of our work. Our goal is to obtain novel cryptographic approaches exploiting the fine-grained nature of the problems, going beyond just recasting normal cryptography in the fine-grained world, and obtaining somewhat generic constructions.

4.1.3 Technical Overview

Here we will go into a bit more technical detail in describing our results. First, we need to describe our hardness assumptions. Then, we will show how to use them for our fine-grained key exchange, and finally, we will talk briefly about fine-grained OWFs and hardcore bits.

Our Hardness Assumption

We generate a series of properties where if a problem has these properties then a fine-grained public key-exchange can be built.

One property we require is that the problem is hard on average, in a fine-grained sense. Intuitively, a problem is average case indistinguishably hard if given an instance that is drawn with probability 1/2 from instances with no solutions and with probability 1/2 from instances with one solution, it is computationally hard on average to distinguish whether the instance has 0 or 1 solutions. The rest of the properties are structural; we need a problem that is *plantable*, *average-case list-hard*, and *splittable*. Informally,

- The plantable property roughly says that one can efficiently choose to generate either an instance without a solution or one with a solution, knowing where the solution is;
- The average case list-hard property says that if one is given a list of instances where all but one of them are drawn uniformly over instances with no solutions, and a random one of them is actually drawn uniformly from instances with one solution, then it is computationally hard to find the instance with a solution;
- Finally, the splittable property says that one can generate from one average case instance, two new average case instances that have the same number of solutions as the original one.

These are natural properties for problems and hypotheses to have. We will demonstrate in Section 4.5.3 that Zero-k-Clique has all of these properties. We need our problem to have all three of these qualities for the key exchange. For our one-way function constructions we only need the problem to be plantable.

The structural properties are quite generic, and in principle, there could be many problems that satisfy them. We exhibit one: the Zero-k-Clique problem.

Because no known algorithmic techniques seem to solve Zero-k-Clique even when the weights are selected independently uniformly at random from $[0, cn^k]$ for a constant c, folklore intuition dictates that the problem might be hard on average for this distribution: here, the expected number of k-Cliques is $\Theta(1)$, and solving the decision problem correctly on a large enough fraction of the random instances seems difficult. This intuition was formally proposed by Pettie [Pet15] for the very related k-Sum problem which we also consider.

We show that the Zero-k-Clique problem, together with the assumption that it is fine-grained hard to solve on average, satisfies all of our structural properties, and thus, using our main theorem, one can obtain a fine-grained key exchange based on Zero-k-Clique.

Key Exchange Assumption. We assume that when given a complete k-partite graph with kn nodes and random weights $[0, R-1]$, $R = \Omega(n^k)$, any adversary running in time $n^{k-\Omega(1)}$ time cannot distinguish an instance with a zero-k-clique solution from one without with more than 2/3 chance of success. In more detail, consider a distribution where with probability 1/2 one generates a random instance of size n with no solutions, and with probability 1/2 one generates a random instance of size n with exactly one solution. (We later tie in this distribution to our original uniform distribution.) Then, consider an algorithm that can determine with probability 2/3 (over the distribution of instances) whether the problem has a solution or not. We make the conjecture that such a 2/3-probability distinguishing algorithm for Zero-k-Clique, which can also exhibit the unique zero clique whenever a solution exists, requires time $n^{k-o(1)}$.

Public Key Exchange

So, what does the existence of a problem with our three properties, *plantable*, *average-case list-hard*, and *splittable*, imply?

The intuitive statement of our main theorem is that, if a problem has the three properties, and is n^k hard to solve on average and can be generated in n^g time (for Zero-k-Clique $g = 2$), then a key exchange exists that takes $O(N)$ time for Alice and Bob to execute, and requires an eavesdropper Eve $\tilde{\Omega}(N^{(3k-2g)/(2k-g)})$ time to break. When $k > g$ Eve takes super linear time in terms of N. When $k = 3$ and $g = 2$, an important case for the Zero-k-Clique problem, Eve requires $\tilde{\Omega}(N^{5/4})$ time.

For the rest of this overview we will describe our construction with the problem Zero-k-Clique.

To describe how we get our key exchange, it is first helpful to consider Merkle Puzzles [Mer78, BGI08, BM09]. The idea is simple: let f be a one way permutation over n bits (so a range of 2^n values) requires $2^{n(\frac{1}{2}+\epsilon)}$ time to invert for some constant $\epsilon > 0$. Then, Alice and Bob could exchange a key by each computing $f(v)$ on $10 \cdot 2^{n/2}$ random element $v \in [2^n]$ and sending those values $f(v)$ to each other. With .9 probability, Alice and Bob would agree on at least one pre-image, v. It would take an eavesdropper Eve $\Omega(2^{n(\frac{1}{2}+\epsilon)})$ time before she would be able to find the v agreed upon by Alice and Bob. So, while Alice and Bob must take $O(2^{n/2})$ time, Eve must take $O(2^{n(\frac{1}{2}+\epsilon)})$ time to break it.

Our construction will take on a similar form: Alice and Bob will send several problems to each other, and some of them will have planted solutions. By matching up where they both put solutions, they get a key exchange.

Concretely, Alice and Bob will exchange m instances of the Zero-k-Clique problem and in $\sqrt{m}$ of them (chosen at random), plant solutions. The other $m - \sqrt{m}$ will not have solutions (except with some small probability). These m problems will be indexed, and we expect Alice and Bob to have both planted a solution in the same index. Alice can check her $\sqrt{m}$ indices against Bob's, while Bob checks his, and by the end, with constant probability, they will agree on a single index as a key. In the end, Alice and Bob require $O(mn^g + \sqrt{m}n^k)$ time to exchange this index. Eve must take time $\tilde{\Omega}(n^k m)$. When $m = n^{2k-2g}$, Alice and Bob take $O(n^{2k-g})$ time and Eve takes $\tilde{\Omega}(n^{3k-2g})$. We therefore get some gap between the running time of Alice and Bob as compared to Eve for any value of $k \geq g$. Furthermore, for all $\delta > 0$ there exists some large enough k such that the difference in running time is at least $O(T(n))$ time for Alice and Bob and $\tilde{\Omega}(T(n)^{1.5-\delta})$ time for Eve. Theorem 31 is the formal theorem statement.

To show hardness for this construction we combine techniques from both fine-grained complexity and cryptography (see Figure 4-2). We take a single instance and use a self-reduction to produce a list of ℓ instances where one has a solution whp if the original instance has a solution. In our reductions ℓ will be polynomial in the input size. Then, we take this list and produce two lists that have a solution in the same location with high probability if the original instance has a solution. Finally, we plant $\sqrt{\ell}$ solutions into the list, to simulate Alice and Bob's random solution planting.

One Way Functions First, and informally, a fine-grained OWF is a function on n bits that requires $\tilde{O}(T(n)^{1-\delta})$ time to evaluate for some constant $\delta > 0$, and if any

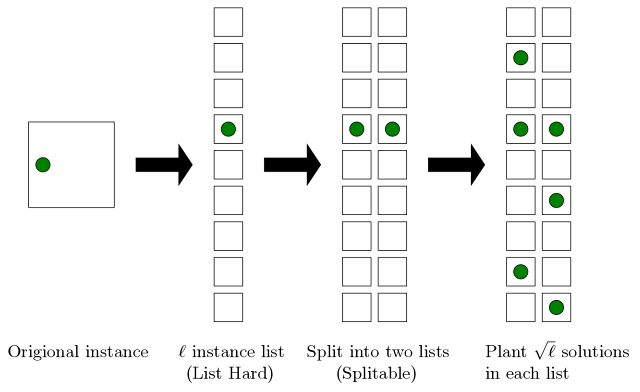
Origional instance
ℓ instance list (List Hard)
Split into two lists (Splitable)
Plant $\sqrt{\ell}$ solutions in each list

Figure 4-2: A depiction of our reduction showing hardness for our fine-grained key exchange.

adversary attempts to invert f in time $\tilde{O}(T(n)^{1-\delta'})$ for *any* constant $\delta' > 0$, she only succeeds with probability at most $\epsilon(n)$, where ϵ is considered "insignificant."

Ball et al. [BRSV17] defined fine-grained OWFs, keeping track of the time required to invert and the probability of inversion in two separate parameters. We streamline this definition by fixing the probability an adversary inverts too an insignificant function of input size, which we define in Section 4.2.

For this overview, we will focus on the intuition of using specific problems k-Sum-R (k-Sum modulo R) or Zero-k-Clique-R (Zero-k-Clique modulo R) to get fine-grained OWFs, though in section 4.6, we construct fine-grained OWFs from a general class of problems. Let N be the size of the input to these problems. Note that if R is too small (e.g. constant), then these problems are solvable quickly and the assumptions we are using are false. So, we will assume $R = \Omega(n^k)$.

OWF Assumptions. Much like for our key exchange, our assumptions are about the difficulty of distinguishing an instance of k-Sum or Zero-k-Clique with probability more than $2/3$ in time faster than $n^{k/2}$ or n^k respectively. Formally, randomly generating a k-Sum-R instance is creating a k lists of size n with values randomly chosen from $[0, R-1]$. Recall that a random Zero-k-Clique instance is a complete k-partite graph where weights are randomly chosen from $[0, R-1]$. Our 'weak' k-Sum-R and Zero-k-Clique-R assumptions state that for any algorithm running in $O(n)$ time, it cannot distinguish between a randomly generated instance with a planted solution and one without with probability greater than $2/3$.

Note that these assumptions are much weaker than the previously described key-exchange assumption, where we allowed the adversary $O(n^{k-\Omega(1)})$ time instead of just super-linear.

Theorem 21 (Fine-Grained OWFs (informal))**.** *If for some constant $\delta > 0$ and range $R = \Omega(n^k)$ either k-Sum-R requires $\Omega(N^{1+\delta})$ time to solve with probability $> 2/3$ or Zero-k-Clique-R requires $\Omega(N^{(1+\delta)})$ time to solve with probability $> 2/3$ then a fine-grained OWF exists.*

The formal theorem is Theorem 28, stated and proved in Section 4.6.2.

Intuitively our construction of a fine-grained OWF runs a planting procedure on a random instance in time $O(N)$. By our assumptions finding this solution takes time $\Omega(N^{1+\delta})$ for some constant $\delta > 0$, and thus inverting this OWF takes $\Omega(N^{1+\delta})$.

We also get a notion of hardcore bits from this. Unlike in traditional crypto, we can't immediately use Goldreich-Levin's hardcore bit construction [GL89]. Given a function on N bits, the construction requires at least $\Omega(N)$ calls to the adversary who claims to invert the hardcore bit. When one is seeking super-polynomial gaps between computation and inversion of a function, factors of N can be ignored. However, in the fine-grained setting, factors of N can completely eliminate the gap between computation and inversion, and so having a notion of fine-grained hardcore bits is interesting.

We show that for our concrete constructions of fine-grained OWFs, there is a subset of the input of size $O(\lg(N))$ (or any sub-polynomial function) which itself requires $\Omega(N^{1+\delta})$ time to invert. From this subset of bits we can use Goldreich-Levin's hardcore bit construction, only losing a factor of $N^{o(1)}$ which is acceptable in the fine-grained setting.

Theorem 22 (Hardcore Bits (informal))**.** *If for some constant $\delta > 0$ and range $R = \Omega(n^k)$ either k-Sum-R requires $\Omega(N^{1+\delta})$ time to solve with probability $> 2/3$ or Zero-k-Clique-R requires $\Omega(N^{1+\delta})$ time to solve with probability $> 2/3$ then a fine-grained OWF exists with a hardcore bit that can not be guessed with probability greater than $\frac{1}{2} + 1/q(n)$ for any $q(n) = n^{o(1)}$.*

The formal theorem is Theorem 29 and is stated and proved in Section 4.6.3.

Intuitively, solutions for k-Sum-R and Zero-k-Clique-R can be described in $O(\log(n))$ bits — we just list the locations of the solution. Given a solution for the problem, we can just change one of the weights and use the solution location to produce a correct preimage. So, now using Goldreich-Levin, we only need to make $O(\log(n))$ queries during the security reduction.

4.1.4 Organization of Chapter

In section 4.2 we define our notions of fine-grained crypto primitives, including fine-grained OWFs, fine-grained hardcore bits, and fine-grained key exchanges. In section 4.3, we describe a few classes of general assumptions (plantable, splittable, and average-case list hard), and then describe the concrete fine-grained assumptions we use (k-Sum and Zero-k-Clique). Next, in section 4.4 we show that the concrete assumptions we made imply certain subsets of the general assumptions. In section 4.7, we show that using an assumption that is plantable, splittable, and average-case list hard, we can construct a fine-grained key exchange.

In Section 4.6, we show how to use a plantable problem to get a fine-grained OWF. In supplementary materials section 4.5 we show that Zero-k-Clique has all of the desired properties (plantable, splittable, and average-case list hard).

4.2 Preliminaries: Model of Computation and Definitions

The running times of all algorithms are analyzed in the word-RAM model of computation, where simple operations such as $+, -, \cdot,$ bit-shifting, and memory access all require a single time-step.

Just as in normal exponential-gap cryptography we have a notion of probabilistic polynomial-time (PPT) adversaries, we can similarly define an adversary that runs in time less than expected for our fine-grained polynomial-time solvable problems. This notion is something we call probabilistic fine-grained time (or PFT). Using this notion makes it easier to define things like OWFs and doesn't require carrying around time parameters through every reduction.

Definition 24. *An algorithm $\mathcal{A}$ is an $T(n)$ probabilistic fine-grained time, $\mathsf{PFT}_{T(n)}$, algorithm if there exists a constant $\delta > 0$ such that $\mathcal{A}$ runs in time $O(T(n)^{1-\delta})$.*

Note that in this definition, assuming $T(n) = \Omega(n)$, any sub-polynomial factors can be absorbed into δ.

Additionally, we will want a notion of *negligibility* that cryptography has. Recall that a function negl(n) is negligible if for all polynomials $Q(n)$ and sufficiently large n, negl$(n) < 1/Q(n)$. We will have a similar notion here, but we will use the words *significant* and *insignificant* corresponding to non-negligible and negligible respectively.

Definition 25. *A function* $\mathsf{sig}(n)$ *is* significant *if*

$$\mathsf{sig}(n) \geq \frac{1}{p(n)}$$

for all polynomials p. A function $\mathsf{insig}(n)$ *is* insignificant *if for all significant functions* $\mathsf{sig}(n)$ *and sufficiently large n,*

$$\mathsf{insig}(n) < \mathsf{sig}(n).$$

Note that for every polynomial f, $1/f(n)$ is insignificant. Also notice that if a probability is significant for an event to occur after some process, then we only need to run that process a sub-polynomial number of times before the event will happen almost certainly. This means our run-time doesn't increase even in a fine-grained sense; i.e. we can boost the probability of success of a randomized algorithm running in $\tilde{O}(T(n))$ from $1/\log(n)$ to $O(1)$ just by repeating it $O(\log(n))$ times, and still run in $\tilde{O}(T(n))$ time (note that '$\tilde{}$' suppresses all sub-polynomial factors in this work).

4.2.1 Fine-Grained Symmetric Crypto Primitives

Ball et all defined fine-grained one-way functions (OWFs) in their work from 2017 [BRSV17]. They parameterize their OWFs with two functions: an inversion-time function $T(n)$ (how long it takes to *invert* the function on n bits), and an probability-of-inversion function ϵ; given $T(n)^{1-\delta'}$ time, the probability any adversary can invert is $\epsilon(T(n)^{1-\delta'})$. The computation time is implicitly defined to be anything noticeably less than the time to invert: there exists a $\delta > 0$ and algorithm running in time $T(n)^{1-\delta}$ such that the algorithm can evaluate f.

Definition 26 ((δ, ϵ)-one-way functions)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is* (δ, ϵ)-one-way *if, for some $\delta > 0$, it can be evaluated on n bits in $O(T(n)^{1-\delta})$ time, but for any $\delta' > 0$ and for any adversary $\mathcal{A}$ running in $O(T(n)^{1-\delta'})$ time and all sufficiently large n,*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathcal{A}(f(x)) \in f^{-1}(f(x))\right] \leq \epsilon(n, \delta).$$

Using our notation of $\mathsf{PFT}_{T(n)}$, we will similarly define OWFs, but with one fewer parameter. We will only be caring about $T(n)$, the time to invert, and assume that the probability an adversary running in time less than $T(n)$ inverts with less time is insignificant. We will show later, in section 4.6, that we can compile fine-grained one-way functions with probability of inversion $\epsilon \leq 1 - \frac{1}{n^{o(1)}}$ into ones with insignificant probability of inversion. So, it makes sense to drop this parameter in most cases.

Definition 27. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is $T(n)$ fine-grained one-way (is an $T(n)$-FGOWF) if there exists a constant $\delta > 0$ such that it takes time $T(n)^{1-\delta}$ to evaluate f on any input, and there exists a function $\epsilon(n) \in \mathsf{insig}(n)$, and for all $\mathsf{PFT}_{T(n)}$ adversaries $\mathcal{A}$,*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathcal{A}(f(x)) \in f^{-1}(f(x))\right] \leq \epsilon(n).$$

With traditional notions of cryptography there was always an exponential or at least super-polynomial gap between the amount of time required to evaluate and invert one-way functions. In the fine-grained setting we have a polynomial *gap* to consider.

Definition 28. *The (relative)* gap *of an $T(n)$ fine-grained one-way function f is the constant $\delta > 0$ such that it takes $T(n)^{1-\delta}$ to compute f but for all $\mathsf{PFT}_{T(n)}$ adversaries $\mathcal{A}$,*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathcal{A}(f(x)) \in f^{-1}(f(x))\right] \leq \mathsf{insig}(n).$$

4.2.2 Fine-Grained Asymmetric Crypto Primitives

In this paper, we will propose a fine-grained key exchange. First, we will show how to do it in an interactive manner, and then remove the interaction. Removing this interaction means that it implies fine-grained public key encryption! Here we will define both of these notions: a fine-grained non-interactive key exchange, and a fine-grained, CPA-secure public-key cryptosystem.

First, consider the definition of a key exchange, with interaction. This definition is modified from [BGI08] to match our notation. We will be referring to a transcript generated by Alice and Bob and the randomness they used to generate it as a "random transcript".

Definition 29 (Fine-Grained Key Exchange)**.** *A $(T(n), \alpha, \gamma)$-$\mathsf{FG\text{-}KeyExchange}$ is a protocol, Π, between two parties A and B such that the following properties hold*

- *Correctness. At the end of the protocol, A and B output the same bit ($b_A = b_B$) except with probability γ;*

$$\Pr_{\Pi,A,B}[b_A = b_B] \geq 1 - \gamma$$

  This probability is taken over the randomness of the protocol, A, and B.

- *Efficiency. There exists a constant $\delta > 0$ such that the protocol for both parties takes time $\tilde{O}(T(n)^{1-\delta})$.*

- *Security. Over the randomness of Π, A, and B, we have that for all $\mathsf{PFT}_{T(n)}$ eavesdroppers E has advantage α of guessing the shared key after seeing a random transcript. Where a transcript of the protocol Π is denoted $\Pi(A, B)$.*

$$\Pr_{A,B}[E(\Pi(A, B)) = b_B] \leq \frac{1}{2} + \alpha$$

A Strong $(T(n))$-$\mathsf{FG\text{-}KeyExchange}$ is a $(T(n), \alpha, \gamma)$-$\mathsf{FG\text{-}KeyExchange}$ where α and γ are insignificant. The key exchange is considered weak *if it is not strong.*

This particular security guarantee protects against chosen plaintext attacks. But first, we need to define what we mean by a fine-grained public key cryptosystem.

Definition 30. *An $T(n)$*-fine-grained public-key cryptosystem *has the following three algorithms.*

$\mathsf{KeyGen}(1^\kappa)$ *Outputs a public-secret key pair (pk, sk).*

$\mathsf{Encrypt}(pk, m)$ *Outputs an encryption of m, c.*

$\mathsf{Decrypt}(sk, c)$ *Outputs a decryption of c, m.*

These algorithms must have the following properties:

- *They are efficient. There exists a constant $\delta > 0$ such that all three algorithms run in time $O\left(T(n)^{1-\delta}\right)$.*

- *They are correct. For all messages m,*

$$\Pr_{\mathsf{KeyGen},\mathsf{Encrypt},\mathsf{Decrypt}}[\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m)) = m | (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)] \geq 1 - \mathsf{insig}(n).$$

The cryptosystem is CPA-secure *if any $\mathsf{PFT}_{T(n)}$ adversary $\mathcal{A}$ has an insignificant advantage in winning the following game:*

1. *Setup. A challenger $\mathcal{C}$ runs $\mathsf{KeyGen}(1^n)$ to get a pair of keys, (pk, sk), and sends pk to $\mathcal{A}$.*

2. *Challenge. $\mathcal{A}$ gives two messages m_0 and m_1 to the challenger. The challenger chooses a random bit $b \overset{\$}{\leftarrow} \{0, 1\}$ and returns $c \leftarrow \mathsf{Encrypt}(pk, m_b)$ to $\mathcal{A}$.*

3. *Guess. $\mathcal{A}$ outputs a guess b' and wins if $b' = b$.*

4.3 Average Case Assumptions

Below we will describe four general properties so that any assumed-to-be-hard problem that satisfies them can be used in our later constructions of one-way functions and cryptographic key exchanges. We will also propose two concrete problems with believable fine-grained hardness assumptions on it, and we will prove that these problems satisfy some, if not all, of our general properties.

Let us consider a search or decision problem P. Any instance of P could potentially have multiple witnesses/solutions. We will restrict our attention only to those instances with no solutions or with exactly one solution. We define the natural uniform distributions over these instances below.

Definition 31 (General Distributions). *Fix a size n and a search problem P. Define $D_0(P, n)$ as the uniform distribution over the set S_0, the set of all P-instances of size n that have no solutions/witnesses. Similarly, let $D_1(P, n)$ denote the uniform distribution over the set S_1, the set of all P-instances of size n that have exactly one unique solution/witness. When P and n are clear from the context, we simply use D_0 and D_1.*

4.3.1 General Useful Properties

We now turn our attention to defining the four properties that a fine-grained hard problem needs to have, in order for our constructions to work with it.

To be maximally general, we present definitions often with more than one parameter. The four properties are: *average case indistinguishably hard, plantable, average case list-hard* and *splittable.*

We state the formal definitions. In these definitions you will see constants for probabilities. Notably $2/3$ and $1/100$. These are arbitrary in that the properties we need are simply that $1/2 < 2/3$ and $2/3$ is much less than $1 - 1/100$. We later boost these probabilities and thus only care that there are constant gaps.

Definition 32 (Average Case Indistinguishably Hard). *For a decision or search problem P and instance size n, let D be the distribution drawing with probability $1/2$ from $D_0(P, n)$ and $1/2$ from $D_1(P, n)$.*

Let $val(I) = 0$ if I is from the support of D_0 and let $val(I) = 1$ if I is from the support of D_1.

P is Average Case Indistinguishably Hard in time $T(n)$ ($T(n)$-ACIH) if $T(n) = \Omega(n)$ and for any $\mathsf{PFT}_{T(n)}$ algorithm A

$$\Pr_{I \sim D}[A(I) = val(I)] \leq 2/3.$$

We also define a similar notion for search problems. Intuitively, it is hard to find a 'witness' for a problem with a solution, but we need to define what a witness is and how to verify a witness in the fine-grained world.

Definition 33 (Average Case Search Hard). *For a search problem P and instance size n, let $D_1 = D_1(P, n)$.*

Let $wit(I)$ denote an arbitrary witness of an instance I with at least one solution. P is Average Case Search Hard in time $T(n)$ if $T(n) = \Omega(n)$ and

- *there exists a $\mathsf{PFT}_{T(n)}$ algorithm V (a fine-grained verifier) such that $V(I, wit(I)) = 1$ if I has a solution and $wit(I)$ is a witness for it and 0 otherwise*
- *and for any $\mathsf{PFT}_{T(n)}$ algorithm A*

$$\Pr_{I \sim D_1}[A(I) = wit(I)] \leq 1/100.$$

Note that ACIH implies ACSH, but not the other way around. In fact, given difficulties in dealing with problems in the average case, getting search-to-decision reductions seems very difficult.

Our next definition describes a fine-grained version of a problem (or relation) being 'plantable' [Lin17]. The definition of a plantable problem from Lindell's book states that a plantable NP-hard problem is hard if there exists a PPT sampling algorithm G. G produces both a problem instance and a corresponding witness (x, y), and over the randomness of G, any other PPT algorithm has a negligible chance of finding a witness for x.

There are a couple of differences between our definition and the plantable definition from Lindell's book the [Lin17]. First, we will of course have to put a fine-grained spin on it: our problem is solvable in time $T(n)$ and so we will need to be secure against $\mathsf{PFT}_{T(n)}$ adversaries. Second, we will be focusing on a decision-version of our problems, as indicated by definition 32. Intuitively, our sampler ($\mathsf{Generate}$) will also take in a bit b to determine whether or not it produces an instance of the problem that has a solution or does not.

Definition 34 (Plantable ($(G(n), \epsilon)$-Plantable))**.** *A $T(n)$-ACIH or $T(n)$-ACSH problem P is* plantable *in time $G(n)$ with error ϵ if there exists a randomized algorithm* $\mathsf{Generate}$ *that runs in time $G(n)$ such that on input n and $b \in \{0, 1\}$,* $\mathsf{Generate}(n, b)$ *produces an instance of P of size n drawn from a distribution of total variation distance at most ϵ from $D_b(P, n)$.*

If it is a $T(n) - ACSH$ problem, then $\mathsf{Generate}(n, 1)$ *also needs to output a witness $wit(I)$, in addition to an instance I.*

We now introduce the List-Hard property. Intuitively, this property states that when given a list of length $\ell(n)$ of instances of P, it is almost as hard to determine if there exists one instance with a solution as it is to solve an instance of size $\ell(n) \cdot n$.

Definition 35 (Average Case List-hard ($(T(n), \ell(n), \delta_{LH})$-ACLH))**.** *A $T(n)$-ACIH or $T(n)$-ACSH problem P is* Average Case List Hard *in time $T(n)$ with list length $\ell(n)$ if $\ell(n) = n^{\Omega(1)}$, and for every $\mathsf{PFT}_{\ell(n) \cdot T(n)}$ algorithm A, given a list of $\ell(n)$ instances, $\mathbf{I} = I_1, I_2, \ldots, I_{\ell(n)}$, each of size n distributed as follows: $i \xleftarrow{\$} [\ell(n)]$ and $I_i \sim D_1(P, n)$ and for all $j \neq i$, $I_j \sim D_0(P, n)$;*

$$\Pr_{\mathbf{I}}[A(\mathbf{I}) = i] \leq \delta_{LH}.$$

It's worth noting that this definition is nontrivial only if $\ell(n) = n^{\Omega(1)}$. Otherwise $\ell(n)T(n) = \tilde{O}(T(n))$, since $\ell(n)$ would be sub-polynomial.

We now introduce the splittable property. Intuitively a splittable problem has a process in the average case to go from one instance I into a pair of average looking problems with the same number of solutions. We use the splittable property to enforce that a solution is shared between Alice and Bob, which becomes the basis of Alice and Bob's shared key (see Figure 4-2).

Definition 36 ((Generalized) Splittable)**.** *A $T(n)$-ACIH problem P is generalized splittable with error ϵ, to the problem P' if there exists a $\mathsf{PFT}_{T(n)}$ algorithm* Split *and a constant m such that*

- *when given a P-instance $I \sim D_0(P, n)$,* $\mathsf{Split}(I)$ *produces a list of length m of pairs of instances $\{(I_1^1, I_2^1), \ldots, (I_1^m, I_2^m)\}$ where $\forall i \in [1, m]$ I_1^i, I_2^i are drawn from a distribution with total variation distance at most ϵ from $D_0(P', n) \times D_0(P', n)$.*

- *when given an instance of a problem $I \sim D_1(P, n)$,* $\mathsf{Split}(I)$ *produces a list of length m of pairs of instances $\{(I_1^1, I_2^1), \ldots, (I_1^m, I_2^m)\}$ where $\exists i \in [1, m]$ such that I_1^i, I_2^i are drawn from a distribution with total variation distance at most ϵ from $D_1(P', n) \times D_1(P', n)$.*

4.3.2 Concrete Hypothesis

Problem Descriptions Two key problems within fine-grained complexity are the k-Sum problem and the Zero-k-Clique problem.

Given k lists of n numbers $L_1, \ldots, L_k$, the k-Sum problem asks, are there $a_1 \in L_1, \ldots, a_k \in L_k$ so that $\sum_{j=1}^{k} a_j = 0$. The fastest known algorithms for k-Sum run in $n^{\lceil k/2 \rceil - o(1)}$ time, and this running time is conjectured to be optimal, in the worst case (see e.g. [Pat10, AW14, Vas18]).

The Zero-k-Clique problem is, given a graph G on n vertices and integer edge weights, determine whether G contains k vertices that form a k-clique so that the sum of all the weights of the clique edges is 0. The fastest known algorithms for this problem run in $n^{k-o(1)}$ time, and this is conjectured to be optimal in the worst case (see e.g. [BT16], [AVW14], [LWW18], [BGMW18]). As we will discuss later, Zero-k-Clique and k-Sum are related. In particular, it is known [WW10] that if 3-Sum requires $n^{2-o(1)}$ time, then Zero-3-Clique requires $n^{3-o(1)}$ time. Zero-3-Clique is potentially even harder than 3-Sum, as other problems such as All-Pairs Shortest Paths are known to be reducible to it, but not to 3-Sum.

A folklore conjecture states that when the 3-Sum instance is formed by drawing n integers uniformly at random from $\{-n^3, \ldots, n^3\}$ no PFT_{n^2} algorithm can solve 3-Sum on a constant fraction of the instances. This, and more related conjectures were explicitly formulated by Pettie [Pet15].

We propose a new hypothesis capturing the folklore intuition, while drawing some motivation from other average case hypotheses such as Planted Clique. For convenience, we consider the k-Sum and Zero-k-Clique problems modulo a number; this variant is at least as hard to solve as the original problems over the integers: we can reduce these original problems to their modular versions where the modulus is only k (for k-Sum) or $\binom{k}{2}$ (for Zero-k-Clique) times as large as the original range of the numbers.

We will discuss and motivate our hypotheses further in Section 4.4.

Definition 37. *An instance of the k-Sum problem over range R, k-Sum-R, consists of kn numbers in k lists $L_1, \ldots, L_k$. The numbers are chosen from the range $[0, R-1]$. A solution of a k-Sum-R instance is a set of k numbers $a_1 \in L_1, \ldots, a_k \in L_k$ such that their sum is zero mod R, $\sum_{i=1}^{k} a_i \equiv 0 \mod R$.*

We will also define the uniform distributions over k-Sum instances that have a certain number of solutions. We define two natural distributions over k-Sum-R instances.

Definition 38. *Define $D^{ksum}_{uniform}[R, n]$ be the distribution of instances obtained by picking each integer in the instance uniformly at random from the range $[0, R-1]$.*

Define $D^{ksum}_0[R, n] = D_0($ k-Sum-$R, n)$ to be the uniform distribution over k-Sum-R instances with no solutions. Similarly, let $D^{ksum}_1[R, n] = D_1($ k-Sum-$R, n)$ to be the uniform distribution over k-Sum-R instances with 1 solution.

The distribution $D_{ksum}[R, i, n]$ is the uniform distribution over k-Sum instances with n values chosen modulo R and where there are exactly i distinct solutions.

Let $D^{ksum}_0[R, n] = D_{ksum}[R, 0, n]$, and $D^{ksum}_1[R, n] = D_{ksum}[R, 1, n]$.

We now proceed to define the version of Zero-k-Clique that we will be using. In addition to working modulo an integer, we restrict our attention to k-partite graphs. In the worst case, the Zero-k-Clique on a general graph reduces to Zero-k-Clique on a complete k-partite graph [2][AYZ16].

Definition 39. *An instance of Zero-k-Clique-R consists of a k-partite graph with kn nodes and partitions $P_1, \ldots, P_k$. The k-partite graph is complete: there is an edge between a node $v \in P_i$ and a node $u \in P_j$ if and only if $i \neq j$. Thus, every instance has $\binom{k}{2}n^2$ edges. The weights of the edges come from the range $[0, R-1]$.*

A solution in a Zero-k-Clique-R instance is a set of k nodes $v_1 \in P_1, \ldots, v_k \in P_k$ such that the sum of all the weights on the $\binom{k}{2}$ edges in the k-clique formed by $v_1, \ldots, v_k$ is congruent to zero mod R: $\sum_{i\in[1,k]} \sum_{j\in[1,k] \text{ and } j\neq i} w(v_i, v_j) \equiv 0 \mod R$. A solution is also called a zero k-clique.

We now define natural distributions over Zero-k-Clique-R instances, similar to those we defined for k-Sum-R. We will additionally define the distributions of these instances in which a certain number of solutions appear.

Definition 40. *Define $D^{zkc}_{uniform}[R, n]$ to be the distribution of instances obtained by picking each integer edge weight in the instance uniformly at random from the range $[0, R-1]$.*

[2]This reduction is done using color-coding ([AYZ16]), an example of this lemma exists in the paper "Tight Hardness for Shortest Cycles and Paths in Sparse Graphs" [LWW18].

Define $D_0^{zkc}[R,n] = D_0(\text{Zero-}k\text{-Clique-}R, n)$ to be the uniform distribution over Zero-k-Clique-R instances with no solutions. Similarly, let $D_1^{zkc}[R,n] = D_1(\text{Zero-}k\text{-Clique-}R, n)$ to be the uniform distribution over Zero-k-Clique-R instances with 1 solution.

The distribution is $D_{zkc}[R,i,n]$ the uniform distribution over zero k-clique instances on kn nodes with weights chosen modulo R and where there are exactly i distinct zero k-cliques in the graph. Let $D_0^{zkc}[R,n] = D_{zkc}[R,0,k]$ and $D_1^{zkc}[R,n] = D_{zkc}[R,1,k]$.

Weak and Strong Hypotheses The strongest hypothesis that one can make is that the average case version of a problem takes essentially the same time to solve as the worst case variant is hypothesized to take. The weakest but still useful hypothesis that one could make is that the average case version of a problem requires *super-linear* time. We formulate both such hypotheses and derive meaningful consequences from them.

We state the weak versions in terms of decision problems and the strong version in terms of search problems. This is for convenience of presenting results. Our fine-grained one-way functions and fine-grained key exchanges can both be built using the search variants. We make these choices for clarity of presentation later on.

Definition 41 (Weak k-Sum-R Hypothesis)**.** *There exists some large enough constant c such that for all constants $c' > c$, distinguishing $D_0^{ksum}[c'R,n]$ and $D_1^{ksum}[c'R,n]$ is $n^{1+\delta}$-ACIH for some $\delta > 0$.*

Definition 42 (Weak Zero-k-Clique-R Hypothesis)**.** *There exists some large enough constant c such that for all constants $c' > c$, distinguishing $D_0^{zkc}[c'R,n]$ and $D_1^{zkc}[c'R,n]$ is $n^{2+\delta}$-ACIH for some $\delta > 0$.*

Notice that the Zero-k-Clique-R problem is of size $O(n^2)$.

Definition 43 (Strong Zero-k-Clique-R Hypothesis for range n^{ck})**.** *For all $c > 1$, given an instance I drawn from the distribution $D_1^{zkc}[n^{ck}, n]$ where the witness (solution) is the single zero k-clique is formed by nodes $\{v_1, \ldots, v_k\}$, finding $\{v_1, \ldots, v_k\}$ is n^k-ACSH.*

Some may find the assumption with range n^k to be the most believable assumption. This is where the probability of a Zero-k-Clique existing at all is a constant.

Definition 44 (Random Edge Zero-k-Clique Hypothesis)**.** *Let $sol(I)$ be a function over instances of Zero-k-Clique problems where $sol(I) = 0$ if there are no zero k-cliques and $sol(I) = 1$ if there is at least one zero k-clique. Let $wit(I)$ be a zero k-clique in I, if one exists. Given an instance I drawn from the distribution $D_{uniform}^{zkc}[n^k, n]$ there is some large enough n such that for any PFT_{n^k} algorithm A*

$$\Pr_{I \sim D}[A(I) = wit(I) | sol(I) = 1] \leq 1/200.$$

Theorem 23. *Strong Zero-k-Clique-R Hypothesis for range $R = n^{ck}$ is implied by the Random Edge Random Edge Zero-k-Clique Hypothesis if $c > 1$ is a constant.*[3]

[3]Thank you to Russell Impagliazzo for discussions related to the sizes of ranges R.

Proof. Create $n^{(1-1/c)}$ random partitions of the nodes where each partition is of size $n^{1/c}$. Then generate $n^{(1-1/c)k}$ graphs by choosing every possible choice of k partitions.

This results in $n^{(1-1/c)k}$ problems of size $n^{1/c}$ with range n^k.

If an algorithm A violated Strong Zero-k-Clique-R Hypothesis for range n^{ck} then it must have some running time of the form $O(n^{k-\delta})$ for $\delta > 0$. We could run A on all $n^{(1-1/c)k}$ problems, resulting in a running time of $n^{k/c-\delta/c}n^{(1-1/c)}k = O(n^{k-\delta/c})$ for the Random Edge problem. If we find a valid zero-k-clique then we return 1. If we don't we return 0 with probability 1/2 and 1 with probability 1/2.

Let p_1 be the probability that any one of the $n^{k/c}$ has exactly one zero clique, conditioned on $val(I) = 1$ (that there is at least one solution).

If Strong Zero-k-Clique-R Hypothesis is violated then we return the correct answer with the probability at least $p_1\frac{1}{100} + (1 - p_1/100)/2 = 1/2 + p_1\frac{1}{200}$. So, we now want to lower bound the value of p_1.

The probability that there are more than 2 cliques in a subproblem of size $n^{1/c}$, conditioned on there being at least one clique is at most $n^{-k(1-1/c)}$. Because to generate the distribution of problems of size $n^{1/c}$ with at least one clique one can plant a clique (randomly choose k nodes and randomly choose $\binom{k}{2} - 1$ edge weights, then choose the final edge weight such that this is a zero k-clique). The expected number of cliques other than the planted clique is $\frac{n^{1/c}-1}{n^k}$ and the number of cliques other than the planted clique is a non-negative integer.

So conditioned $val(I) = 1$ we have that $p_1 \geq 1 - n^{-k(1-1/c)}$. So the probability of success, conditioned on $val(I) = 1$ is at least $p_1/100 \geq 1/200$. □

4.4 Our assumptions - background and justification

In this section, we justify making average-case hardness assumptions for k-SUM and Zero k-Clique — and why we do not for other fine-grained problems. We start with some background on these problems, and then justify why our hypotheses are believable.

4.4.1 Background for Fine-Grained Problems

Among the most popular hypotheses in fine-grained complexity is the one concerning the 3-Sum problem defined as follows: given three lists A, B and C of n numbers each from $\{-n^t, \ldots, n^t\}$ for large enough t, determine whether there are $a \in A, b \in B, c \in C$ with $a + b + c = 0$. There are multiple equivalent variants of the problem (see e.g. [GO12]).

The fastest 3-Sum algorithms run in $n^2(\log\log n)^{O(1)}/\log^2 n$ time (Baran, Demaine and Patrascu for integer inputs [BDP08], and more recently Chan'18 for real inputs [Cha18]). Since the 1990s, 3-Sum has been an important problem in computational geometry. Gajentaan and Overmars [GO12] formulated the hypothesis that 3-Sum requires quadratic time (nowadays this means $n^{2-o(1)}$ time on a word-RAM with $O(\log n)$ bit words), and showed via reductions that many geometry problems also require quadratic time under this hypothesis. Their work spawned many more within geometry. In recent years, many more consequences of this hypothesis have been derived, for a variety of non-geometric problems, such as sequence local alignment [AVW14], triangle enumeration [Pat10, KPP16], and others.

As shown by Vassilevska Williams and Williams [WW10], 3-Sum can be reduced to a graph problem, 0-Weight Triangle, so that if 3-Sum requires $n^{2-o(1)}$ time on inputs of size n, then 0-Weight Triangle requires $N^{3-o(1)}$ time in N-node graphs. In fact, Zero-Weight Triangle is potentially harder than 3-Sum, as one can also reduce to it the All-Pairs Shortest Paths (APSP) problem, which is widely believed to require essentially cubic time in the number of vertices. There is no known relationship (via reductions) between APSP and 3-Sum.

The Zero-Weight Triangle problem is as follows: given an n-node graph with edge weights in the range $\{-n^c, \ldots, n^c\}$ for large enough c, denoted by the function $w(\cdot, \cdot)$, are there three nodes p, q, r so that $w(p, q) + w(q, r) + w(r, p) = 0$? Zero-Weight Triangle is just Zero-3-Clique where the numbers are from a polynomial range.

An equivalent formulation assumes that the input graph is tripartite and complete (between partitions).

Both 3-Sum and Zero-Weight Triangle have generalizations for $k \geq 3$: k-Sumand Zero-Weight k-Clique, defined in the natural way: (1) given k lists of n numbers each from $\{-n^{ck}, \ldots, n^{ck}\}$ for large c, are there k numbers, one from each list, summing to 0? and (2) given a complete k-partite graph with edge weights from $\{-n^{kc}, \ldots, n^{kc}\}$ for large c, is there a k-clique with total weight sum 0?

4.4.2 Justifying the Hardness of Some Average-Case Fine-Grained Problems

The k-Sum problem is conjectured to require $n^{\lceil k/2 \rceil - o(1)}$ time (for large enough weights), and the Zero-Weight k-Clique problem is conjectured to require $n^{k-o(1)}$ time (for large enough weights), matching the best known algorithms for both problems (see [Vas18]). Both of these conjectures have been used in fine-grained complexity to derive conditional lower bounds for other problems (e.g. [BT16], [AVW14], [LWW18], [BGMW18]).

It is tempting to conjecture average-case hardness for the key hard problems within fine-grained complexity: Orthogonal Vectors (OV), APSP, 3-Sum. However, it is known that APSP is not hard on average, for many natural distributions (see e.g. [PSSZ13, CFMP00]), and OV is likely not (quadratically) hard on average (see e.g. [KW17]).

On the other hand, it is a folklore belief that 3-Sum is actually hard on average. In particular, if one samples n integers uniformly at random from $\{-cn^3, \ldots, cn^3\}$ for constant c, the expected number of 3-Sums in the instance is $\Theta(1)$, and there is no known truly subquadratic time algorithm that can solve 3-Sum reliably on such instances. The conjecture that this is a hard distribution for 3-Sum was formulated for instance by Pettie [Pet15].

The same folklore belief extends to k-Sum. Here a hard distribution seems to be to generate k lists uniformly from a large enough range $\{-cn^k, \dots, cn^k\}$, so that the expected number of solutions is constant.

Due to the tight relationship between 3-Sum and Zero-Weight Triangle, one might also conjecture that uniformly generated instances of the latter problem are hard to solve on average. In fact, if one goes through the reductions from the worst-case 3-Sum problem to the worst-case Zero-Weight Triangle, via the 3-Sum Convolution problem [Pat10, WW13] starting from an instance of 3-Sum with numbers taken uniformly at random from a range, then one obtains a list of Zero-Weight Triangle instances that are essentially average-case. This is easier to see in the simpler but less efficient reduction in [WW13] which from a 3-Sum instance creates $n^{1/3}$ instances of (complete tripartite) Zero-Weight Triangle on $O(n^{2/3})$ nodes each and whose edge weights are exactly the numbers from the 3-Sum instance. Thus, at least for $k = 3$, average-case hardness for 3-Sum is strong evidence for the average-case hardness for Zero-Weight Triangle.

Previously, for Theorem 23, we gave a reduction between uniform instances of uniform Zero-Weight k-Clique with range $\Theta(n^k)$ and instances of planted Zero-Weight k-Clique with large range. Working with instances of planted Zero-Weight k-Clique with large range is easier for our hardness constructions, so we use those in most of this paper.

Justifying the Hardness of Distinguishing. Now, our main assumptions consider distinguishing between the distributions D_0 and D_1 for 3-Sum and Zero-Weight Triangle. Here we take inspiration from the Planted Clique assumption from Complexity [HK11, Jer92, Kuc95]. In Planted Clique, one first generates an Erdös-Renyi graph that is expected to not contain large cliques, and then with probability 1/2, one plants a clique in a random location. Then the assertion is that no polynomial time algorithm can distinguish whether a clique was planted or not.

We consider the same sort of process for Zero-k-Clique. Imagine that we first generate a uniformly random instance that is expected to have no zero k-Cliques, by taking the edge weights uniformly at random from a large enough range, and then we plant a zero k-Clique with probability 1/2 in a random location. Similarly to the Planted Clique assumption, but now in a fine-grained way, we can assume that distinguishing between the planted and the not-planted case is computationally difficult.

Our actual hypothesis is that when one picks an instance that has no zero k-Cliques at random with probability 1/2 and picks one that has a zero k-Clique with probability 1/2, then distinguishing these two cases is hard. As we show later, this hypothesis is essentially equivalent to the planted version (up to some slight difference between the underlying distributions).

Similarly to Planted Clique, no known approach for Zero-k-Clique seems to work in this average-case scenario, faster than essentially n^k, so it is natural to hypothesize that the problem is hard. We leave it as a tantalizing open problem to determine whether the problem is actually hard, either by reducing a popular worst-case hypothesis to it, or by providing a new algorithmic technique.

4.5 Properties of k-Sum and Zero-k-Clique Hypotheses

In this section, we will prove the properties that k-Sum and Zero-k-Clique have that will make them useful in constructing fine-grained OWFs and our fine-grained key exchange.

4.5.1 k-Sum is Plantable from a Weak Hypothesis

Here we will show that by assuming the Weak k-Sum hypothesis (see definition 41), we get that k-Sum is plantable and $n^{2+\delta}$-ACIH. The proof is relatively straightforward: just show that planting a solution in a random k-Sum-R instance is easy while making sure that the distributions are close to what you expect.

Theorem 24. *Assuming the weak k-Sum-R hypothesis, k-Sum-R is plantable with error $\leq 2n^k/R$ in $O(n)$ time.*

Proof. First, we will define $\mathsf{Generate}(n, b)$:

- $b = 0$: choose all kn entries uniformly at random from $[0, R-1]$, taking time $O(n)$.
- $b = 1$: choose all kn entries uniformly at random from $[0, R-1]$, then choose values $v_1, \ldots, v_k$, each v_i at random from partition P_i, and choose $i \xleftarrow{\$} [k]$. Set $v_i = -\sum_{j\neq i} v_j \mod R$. This takes time $O(n)$.

We need to show that $\mathsf{Generate}(n, 0)$ is ϵ-close to D_0 and $\mathsf{Generate}(n, 1)$ is ϵ-close to D_1.

First, we note that $\mathsf{Generate}(n, 0)$ has the following property: $\Pr_{I\sim\mathsf{Generate}(n,0)}[I = I'|I \text{ has no solutions}] = \Pr_{I\sim D_0}[I = I']$. This is because $\mathsf{Generate}(n, 0)$ samples uniformly over the support of D_0. So, the total variation distance between $\mathsf{Generate}(n, 0)$ and D_0 is the probability $\mathsf{Generate}(n, 0)$ samples outside of the support of D_0, that is, the probability $\mathsf{Generate}(n, 0)$ samples an I with a value 1 or greater. Let TVD denote Total Variation Distance between two distributions. Now, a union bound gives us

$$\begin{aligned}\mathrm{TVD}(\mathsf{Generate}(n,0), D_0) &= \Pr_{I\sim\mathsf{Generate}(n,0)}[I \text{ has at least 1 solution}]\\ &\leq \sum_{\text{all } n^k \text{ sums } \mathbf{s}\in[n]^k} \Pr_{I\sim\mathsf{Generate}(n,0)}[\mathbf{s} \text{ is a } k\text{-Sum}]\\ &= \frac{n^k}{R}.\end{aligned}$$

Now, to show that $\mathsf{Generate}(n, 1)$ is ϵ-close to D_1, we will use the fact that total-variation distance (TVD) is a metric and the triangle inequality. Let $\mathsf{Generate}(n, 0)+$ Plant and D_0+ Plant denote sampling from the first distribution and planting a k-Sum solution at random (so $\mathsf{Generate}(n, 0)+$ Plant $=$ $\mathsf{Generate}(n, 1)$). We have that

$$\begin{aligned}\mathrm{TVD}(\mathsf{Generate}(n,1), D_1) \leq\ & \mathrm{TVD}(\mathsf{Generate}(n,0) + \text{Plant}, D_0 + \text{Plant})\\ & + \mathrm{TVD}(D_0 + \text{Plant}, D_1).\end{aligned}$$

The distance Generate$(n, 0)+$ Plant from D_0+ Plant is equal to the distance from Generate$(n, 0)$ and D_0, since the planting does not change between distributions. As previously shown, this distance is at most $\frac{n^k}{R}$. The distance from D_0+ Plant and D_1 is just the chance that we introduce more than one clique by planting. We are only changing one value in the D_0 instance, v_i. There are $n^{k-1} - 1 \leq n^{k-1}$ possible sums involving v_i, so the chance that we accidentally introduce an unintended k-Sum solution is at most $\frac{n^{k-1}}{R}$. Therefore,

$$\text{TVD}(\textsf{Generate}(n, 1), D_1) \leq \frac{n^k}{R} + \frac{n^{k-1}}{R} < \frac{2n^k}{R}$$

□ □

Note that when $R > 6n^k$, Generate$(n, 1)$ has total variation distance $< 1/3$ from $D_1(k-\text{SUM}-R, n)$.

4.5.2 Zero-k-Clique is also Plantable from Weak or Strong Hypotheses

The proof in this section mirrors of the proof that k-Sum-R is plantable. Note that the size of a k-Clique instance is $O(n^2)$, and so the fact that this requires $O(n^2)$ time is just that it is linear in the input size. Here we will just list what the Generate functionality is:

- Generate$(n, 0)$ outputs a complete k-partite graph with n nodes in each partition, and edge weights drawn uniformly from $\mathbb{Z}_R$. This takes $O(n^2)$ time.

- Generate$(n, 1)$ starts with Generate$(n, 0)$, and then plants a clique by choosing a node from each partition, $v_1 \in P_1, \ldots, v_k \in P_k$, choosing an $i \neq j \xleftarrow{\$} [k]$, and setting the weight $w(v_i, v_j) = -\sum_{(i',j') \neq (i,j)} w(v_{i'}, v_{j'}) \mod R$. This also takes $O(n^2)$ time.
  If assuming the strong hypothesis (search problem), we can also output a witness, $(v_1, \ldots, v_k)$, of size $O(\log n)$.

Unfortunately for it seems difficult to show that k-Sum is average-case list-hard or splittable. However, we will show that if we assume that Zero-k-Clique is only *search* hard (a strictly weaker assumption than being indistinguishably hard), we can get the plantable, list-hard, and splittable properties — the caveat is that we need to assume that Zero-k-Clique requires $\tilde{\Omega}(n^k)$ time to solve (not just super-linear in time).

Before proving the theorem, we need a couple of helper lemmas to characterize the total variation distance, etc. These lemmas will be useful later on as well.

Lemma 23. *The distribution $D_0^{zkc}[R, n]$ has total variation distance $\leq n^k/R$ from the distribution of instances drawn from* Generate$(n, 0)$.

Proof. $D_0^{zkc}[R,n]$ is uniform over all instances of size n where there are no solutions. $\mathsf{Generate}(n,0)$ is uniform over all instances of size n.

Let D be the distribution of instances in $\mathsf{Generate}(n,0)$ which are in the support of $D_0^{zkc}[R,n]$. Because both $\mathsf{Generate}(n,0)$ and $D_0^{zkc}[R,n]$ are uniform over the support of $D_0^{zkc}[R,n]$, $D = D_0^{zkc}[R,n]$.

So the total variation distance between $D_0^{zkc}[R,n]$ and $\mathsf{Generate}(n,0)$ is just

$$Pr_{I\sim\mathsf{Generate}(n,0)}[I \notin \text{ the support of } D_0^{zkc}[R,n]].$$

The expected number of zero k-cliques is n^k/R, every set of k nodes has a chance of $1/R$ of being a zero k-clique. Thus, the probability that an instance has a non-zero number of solutions is $\leq n^k/R$. So, the total variation distance is $\leq n^k/R$. □

Lemma 24. *The distribution $D_1^{zkc}[R,n]$ has total variation distance $\leq n^k/R + n^{k-2}/R$ from the distribution of* $\mathsf{Generate}(n,1)$.

Proof. We want to first show that $\mathsf{Generate}(n,1)$ is uniform over the support of $D_1^{zkc}[R,n]$. Consider an instance I in the support of $D_1^{zkc}[R,n]$. Let $S(I) = a_1,\ldots,a_k$ be the set of k nodes in which there is a zero k-clique. $Pr_{I'\sim\mathsf{Generate}(n,1)}[I' = I]$ is given by the chance that

- the nodes chosen in I' $(a'_1,\ldots,a'_k)$ to plant a clique are the same as those in $S(I)$,
- the edges in the clique have the same weights in I' and I and,
- all edges outside the clique have the same weight in I' and I.

$$Pr_{I'\sim\mathsf{Generate}(n,1)}[I' = I] = \left(n^{-k}\right)\left(R^{-\binom{k}{2}-1}\right)\left(R^{-\binom{k}{2}(n^2-1)}\right).$$

This is the same probability for all instances I in the support of $D_1^{zkc}[R,n]$.

So, we need only bound the probability

$$Pr_{I\sim\mathsf{Generate}(n,1)}[I \notin \text{ the support of } D_1^{zkc}[R,n]].$$

By Lemma 23 the initial process of choosing edges the probability of producing a clique is $\leq n^k/R$. We then change one edge's weight, this introduces a clique. It introduces an expected number of additional cliques $\leq n^{k-2}/R$ (this is the number of cliques it participates in). Thus, we can bound the probability of more than one clique by $\leq n^k/R + n^{k-2}/R$. □

Theorem 25. *Assuming the weak Zero-k-Clique hypothesis (ACIH) over range R, Zero-k-Clique is $(O(n^2), 2n^k/R)$-Plantable.*
Assuming the strong Zero-k-Clique hypothesis (ACSH) over range R, Zero-k-Clique is also $(O(n^2), 2n^k/R)$-Plantable.

Proof. This proof simply combines the two previous lemmas: Lemma 23 and Lemma 24.

$\mathsf{Generate}(n,0)$ has total variation distance n^k/R from $D_0^{zkc}[R,n]$ by Lemma 23, and $\mathsf{Generate}(n,1)$ has total variation distance $n^k/R + n^{k-2}/R < 2n^k/R$ from $D_1^{zkc}[R,n]$ by Lemma 24. So, in both cases the error is bounded above by $2n^k/R$.

Finally note that $\mathsf{Generate}(n,1)$ also can output the planted solution, the clique it chose to set to 0, and so can output a witness. □

4.5.3 Zero-k-Clique is Plantable, Average Case List-Hard and, Splittable from the Strong Zero-k-Clique Hypothesis

Here we will focus on the Strong Zero-k-Clique assumption, see Definition 43. Recall that this is the search version of the problem: given a graph with weights on its edges drawn uniformly from the k-partite graphs with exactly one zero k-clique, it is difficult to find the clique in time less than $\tilde{O}(n^k)$.

We already proved that Zero-k-Clique was Plantable in Theorem 24. So, now we will focus on the other two properties we want: list-hardness and splittability. These will give us the properties we need for our key exchange.

Zero-k-Clique is Average Case List-Hard

We present the proof that Zero-k-Clique is average case list-hard.

The intuition of the proof is as follows. There is an efficient worst case self-reduction for the Zero-k-Clique problem. This self-reduction results in $\ell'(n)^k$ sub-problems of size $n/\ell'(n)$. One can choose $\ell'(n)$ of these instances such that they are generated from non-overlapping parts of the original instance. They will then look uniformly randomly generated.

Now we will have generated many, $(\ell'(n))^k$, of these list versions of the Zero-k-Clique problem, where only one of them has the unique solution. We show that the problem is Average Case List-Hard by demonstrating that we can make many independent calls to the algorithm despite correlations between the instances called. Specifically, we only care about the response on one of these instances, so as long as that instance is random then we can solve the original problem.

Theorem 26. *Given the strong Zero-k-Clique-R Hypothesis , Zero-k-Clique is Average Case List-Hard with list length $\ell(n)$ for any $\ell(n) = n^{\Omega(1)}$.*

Proof. Let $\ell = \ell(n)$ for the sake of notation. $I \sim D_1(\text{Zero-}k\text{-Clique}, \ell \cdot n)$ with k partitions, $P_1, \dots, P_k$ of $\ell \cdot n$ nodes each and with edge weights generated uniformly at random from $\mathbb{Z}_R$.

Randomly partition each P_i into ℓ sets $P_i^1, \dots, P_i^\ell$ where each set contains n nodes. Now, note that if we look for a solution in all ℓ^k instances formed by taking every possible choice of $P_1^{i_1}, P_2^{i_2}, \dots, P_k^{i_k}$, this takes time $O((\ell \cdot n)^k)$, which is how long the original size ℓn problem takes to solve.

Sadly, not all ℓ^k instances are independent. We want to generate sets of independent instances. Note that if we choose ℓ of these sub-problems such that the nodes don't overlap, then the edges were chosen independently between each instance! Specifically consider all vectors of the form $\mathbf{x} = \langle x_2, \dots, x_k \rangle \in \mathbb{Z}_\ell^{k-1}$. Then let

$$S_{\mathbf{x}} = \{P_1^i \cup P_2^{i+x_2} \cup \ldots \cup P_k^{i+x_k} | \forall i \in [1, \ell]\}$$

be the set of all independent partitions. Now, note that $\cup_{\mathbf{x} \in \mathbb{Z}_\ell^{k-1}} S_{\mathbf{x}}$ is the full set of all possible ℓ^k subproblems, and the total number of problems in all $S_{\mathbf{x}}$ is ℓ^k, so once again brute-forcing each $S_{\mathbf{x}}$ takes time $O((\ell \cdot n)^k)$. We depict this splitting in Figure 4-3.

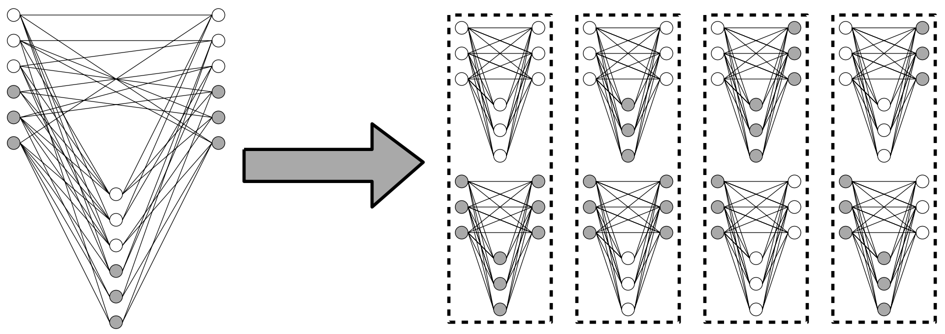

Figure 4-3: A depiction of splitting the subproblems for a case where $\ell = 2$ and $k = 3$.

Note that producing these each of these ℓ instances is efficient, it takes time $O(n^2)$, which is just the input size.

Next, we will show that the correct number of solutions are generated. If I has only one solution then exactly one I_j in exactly one $S_{\mathbf{x}}$ has a solution. This is because any zero-k-clique in I must involve exactly one node from each partition P_i. So, if there is one zero-k-clique it will only appear in subproblems where the node from partition P_i is in P_i^j and P_j^i appears in that subproblem. There is exactly one subproblem generated with a specific choice of k sub-partitions. So, exactly one I_j in exactly one $S_{\mathbf{x}}$ has a solution.

Let S^* be the list $S_{\mathbf{x}}$ that contains a Zero-k-Clique. We have that the $S_{\mathbf{x}}$ which actually contains an instance with a solution is drawn from

$$\{I_1, \ldots, I_x\}_{I_i \sim D_1, \wedge \forall j \neq i, I_j \sim D_0}.$$

This distribution is exactly what we require for a list-problem. All that is left to show is if we have $\mathsf{PFT}_{\ell \cdot n^k}$ adversary $\mathcal{A}$ that can identify for which index i there is a zero k-clique in S^* (with probability at least 7/10), we can use $\mathcal{A}$ to find the clique.

Now, recall that we are trying to solve a search problem: we need to be able to turn an index pointing to partitions into a witness for the original problem. According to the strong Zero-k-Clique hypothesis, this search requires $O(n^k)$ time. However, as long as $\ell = n^{\Omega(1)}$, this is still faster in a fine-grained sense.

On an input I from D_1, algorithm $\mathcal{B}$ uses $\mathcal{A}$ as follows:

- Randomly partition each P_i from I into ℓ parts.
- For every $\mathbf{x} \in \mathbb{Z}_\ell^{k-1}$:
  - Generate the list $S_{\mathbf{x}}$.
  - Run $\mathcal{A}(S_{\mathbf{x}})$ to get output i.
  - Brute force search the size-n^2 Zero-k-Clique instance $S_{\mathbf{x}}[i] = (P_1^i, \ldots, P_k^{i+x_k})$ for a solution. If one exists, output it, otherwise, continue.

The first step only takes $O(\ell \cdot n)$ time since we are only divvying up ℓn nodes. The second step requires a bit more analysis. The loop runs at most ℓ^{k-1} times.

Each time the loop runs, it only takes $O(\ell \cdot n)$ time to construct $S_{\mathbf{x}}$, while $\mathcal{A}$ takes $O((\ell \cdot n^k)^{(1-\epsilon)})$ (since it is $\mathsf{PFT}_{\ell \cdot n^k}$), and our brute force check takes $O(n^k)$ time. Putting this together, the algorithm takes a total time of

$$O(\ell \cdot n + \ell^{k-1}((\ell \cdot n^k)^{(1-\epsilon)} + n^k + \ell \cdot n)) = O(\ell^{k-\epsilon} n^{k(1-\epsilon)} + \ell^{k-1} n^k) + \ell^k \cdot n.$$

Both terms in this sum are strictly less than the hypothesized $\ell^k n^k$ time, and so $\mathcal{B}$ is $\mathsf{PFT}_{(\ell n)^k}$, contradicting the strong Zero-k-Clique hypothesis.

The reason we require $\ell(n) = n^{\Omega(1)}$ is because if it were less than polynomial in n, we would not get noticeable improvement through this method of splitting up the problem into several sub-problems — the brute force step would take as long as solving the original problem via brute force. □ □

Zero-k-Clique is Splittable

Next we show that zero-k-clique is splittable. We start by proving this for a convenient range and then show we can use a reduction to get more arbitrary ranges.

Splitting the problem over a convenient range. Intuitively we will split the weights in half bit-wise, taking the first half of the bits of each edge weight, and then we take the second half of the bits of each edge weight to make another instance. If the $\binom{k}{2}$ weights on a k clique sum to zero then the first half of all the weights sum to zero, up to carries, and the second half of all the weights sum to zero, also up to carries. We simply guess the carries.

Lemma 25. *Zero-k-Clique is generalized splittable with error* $\leq 4(\binom{k}{2} + 1)n^k/\sqrt{R}$ *when* $R = 4^x$ *for some integer* x.

Proof. We are given an instance of Zero-k-Clique I with k partitions, $P_1, \ldots, P_k$ of n nodes and with edge weights generated uniformly at random from $[0, R-1]$, where $R = 2^{2x}$ for some positive integer x.

First we will define some helpful notation to describe our procedure.

- Let ZkC$[R]$ denote the Zero-k-Clique problem over range R.
- Let $w(P_i[a], P_j[b])$ be the weight of the edge in instance I between the a^{th} node in P_i and the b^{th} node in P_j.
- Let u be some number in the range $[0, R-1]$. Let $u^{\uparrow}$ be the high order $\lg(R)/2$ bits of the number u (this will be an integer because R is a power of 4). Let $u_{\downarrow}$ be the low order $\lg(R)/2$ bits of the number u.
  For the sake of notation, $w^{\uparrow}(P_i[a], P_j[b])$ denotes $[w(P_i[a], P_j[b])]^{\uparrow}$, and same for $w_{\downarrow}(P_i[a], P_j[b])$ denoting $[w(P_i[a], P_j[b])]_{\downarrow}$.

Here is the reduction to take one instance of ZkC$[R]$ and create a list of pairs of instances of ZkC$[\sqrt{R}]$.

1. Take the ZkC$[R]$ instance I and create two instances of ZkC$[\sqrt{R}]$, I_{low} and I_{high} by the following:

- For every edge $(P_i[a], P_j[b])$ in I, let the corresponding edge in I_{low} have weight $w_{\downarrow}(P_i[a], P_j[b])$ and the edge in I_{high} have weight $w^{\uparrow}(P_i[a], P_j[b])$.

2. For every $c \in [0, \binom{k}{2}]$ (we need only check $\binom{k}{2}$ possible carries):

   (a) Let I_1^c be a copy of I_{low}, but randomly permute all nodes.

   (b) Let I_2^c be a copy of I_{high}, but choose a random pair of a partitions P_i and P_j: for all edges $e_2 \in I_2^c$ between P_i and P_j, a copy of edge $e \in I_{high}$, let $w(e_2) = w(e) + c \mod \sqrt{R}$.

3. Output the list $[(I_1^{(0)}, I_2^{(0)}), \ldots, (I_1^{(\binom{k}{2})}, I_2^{(\binom{k}{2})})]$

For a visual aid, see figure 4-4 for a depiction of the splittable triangles.

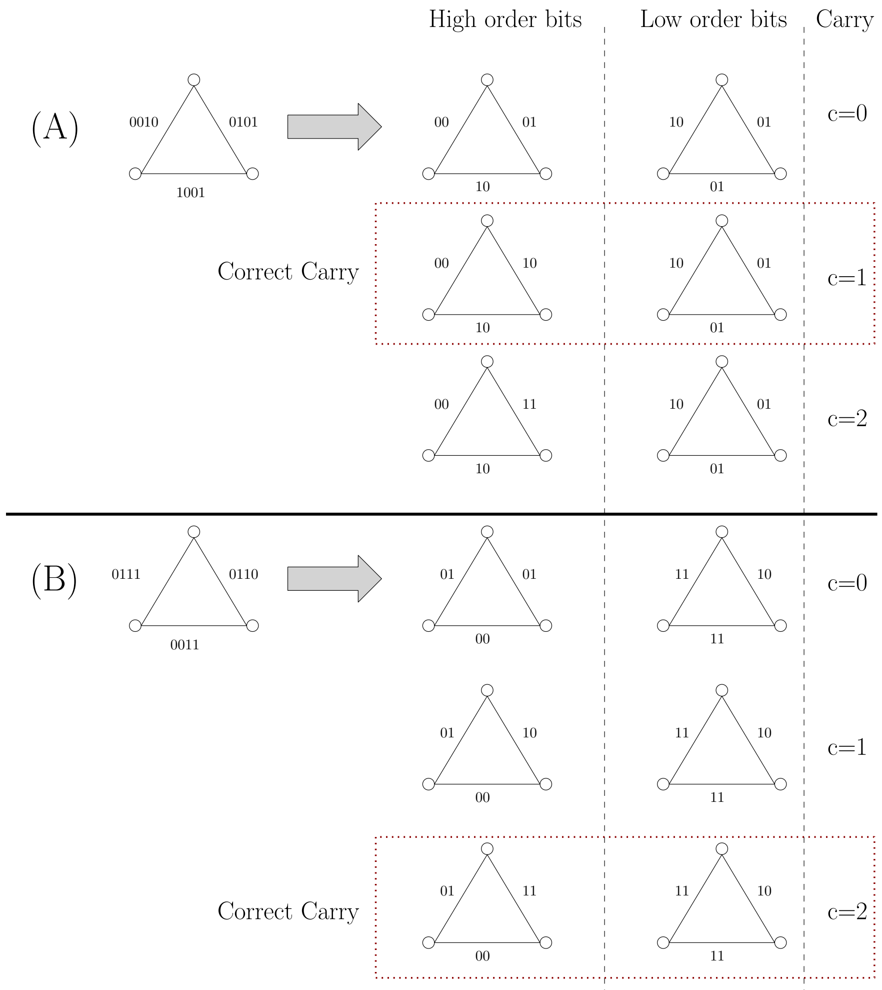

Figure 4-4: An example of splitting the edges of triangles whose edges sum to 16.

We will now show that we get the desired distributions in our list of instances depending on whether $I \sim D_0(\text{ZkC}[R], n)$ or $I \sim D_1(\text{ZkC}[R], n)$.

- $I \sim D_0(\mathrm{ZkC}[R], n)$. We need to show that every pair $(I_1^{(c)}, I_2^{(c)})$ is sampled from a distribution total variation distance $\leq 2n^k/\sqrt{R}$ from $D_0(\mathrm{ZkC}[\sqrt{R}], n)^2$. Note that every pair is correlated very heavily with every other pair with respect to edge weights. But, within each pair, they are close to $D_0(\mathrm{ZkC}[\sqrt{R}], n)^2$.

  From lemma 23, this is TVD at most $\frac{n^k}{R}$ from just choosing edge-weights uniformly at random. So, consider $I' \sim \mathsf{Generate}(n, 0)$, and do the same operations as for I in the reduction: every bit in every edge weight will be chosen uniformly at random, meaning that the edge-weights in I'_{low} and I'_{high} will also be uniform over $\sqrt{R}$. Permuting the nodes in I'_{low} does not change this distribution, and neither does adding (any) c to a subset of edges in I'_{high}. Therefore, using lemma 23, *both* $I_1'^{(c)}$ and $I_2'^{(c)}$ are TVD at most $\frac{n^k}{\sqrt{R}}$ from $D_0(\mathrm{ZkC}[\sqrt{R}], n)$. Since TVD is a metric, this implies that $I_1^{(c)}$ is TVD at most $n^k/\sqrt{R}$ from the distribution of $I_1'^{(c)}$, and thus at most $n^k/\sqrt{R} + n^k/R$ from $D_0(\mathrm{ZkC}[\sqrt{R}], n)$ — the same is true for $I_2^{(c)}$, even when conditioned on $I_1^{(c)}$. Therefore, the pair, for every c, is TVD at most $2(n^k/\sqrt{R} + n^k/R) \leq \frac{4n^k}{\sqrt{R}}$.

- $I \sim D_1(\mathrm{ZkC}[R], n)$. We want to show that we get a list in which exactly one of the pairs of instances is distributed close to $D_1(\mathrm{ZkC}[\sqrt{R}], n)^2$.

  We will take a similar approach here, considering the planted distribution of I instead of the true one. Let $I' \sim \mathsf{Generate}(n, 1)$, so by lemma 24, I' is TVD at most $2n^k/R$ from D_1. We will first show that I'_{low} is also drawn from a planted distribution over the range $\sqrt{R}$. Let e' be the edge's weight that was changed to plant a zero clique. Now, for every edge except e'_{low}, the edges of I'_{low} are distributed uniformly. e' is a randomly chosen edge corresponding to a randomly chosen clique in I', and therefore e'_{low} is also a randomly chosen edge corresponding to a randomly chosen clique in I'_{low}. The act of making that clique sum to 0 mod R also requires that the low-order bits sum to 0 mod $\sqrt{R}$ — otherwise the high-order bits cannot cancel out anything left over. Therefore, by setting $w(e')$ to the value making the clique sum to 0, we are exactly planting a clique in I'_{low}. This distribution has TVD $\leq \frac{2n^k}{\sqrt{R}}$ from $D_1(\mathrm{ZkC}[\sqrt{R}], n)$. Because $I_1'^{(c)}$ is just a permutation on the nodes of I'_{low} for every c, $I_1'^{(c)}$ will have TVD at most $\frac{2n^k}{\sqrt{R}}$ from D_1 as well.

  Now, we need that at least one of the pairs in this list to be close to $D_1(\mathrm{ZkC}[\sqrt{R}], n) \times D_1(\mathrm{ZkC}[\sqrt{R}], n)$. It will turn out that there exists a c such that $I_2'^{(c)}$ will also be close to D_1 (whereas $I_1'^{(c)}$ is distributed close to D_1 for every c). Let c^* be the correct carry — that is for the clique planted in I', $\sum_{e \in clique} w_\downarrow(e) = \sqrt{R}c^*$ mod R. Now, without loss of generality, we can assume that in the plant of I', the edge e^* chosen to complete the zero-k clique was between partitions P_i and P_j. So, considering every other edge in $I_2'^{(c^*)}$, it is distributed uniformly at random (adding c^* will not change that distribution). Now, for that special clique C^* that was planted in I', we have that

$$\begin{aligned}\sum_{e\in C^*} w(e) &= \sqrt{R}\cdot \sum_{e\in C^*} w^{\uparrow}(e) + \sum_{e\in C^*} w_{\downarrow}(e)\\ &= \sqrt{R}(\sum_{e\in C^*} w^{\uparrow}(e) + c^*)\\ &= \sqrt{R}(w^{\uparrow}(e^*) + c^* + \sum_{e\in C^*, e\neq e^*} w^{\uparrow}(e)) = 0 \mod R\end{aligned}$$

Since the quantity $\sqrt{R}(w^{\uparrow}(e^*)+c^*+\sum_{e\in C^*,e\neq e^*} w^{\uparrow}(e))$ is 0 mod R, then $w^{\uparrow}(e^*)+c^*+\sum_{e\in C^*,e\neq e^*} w^{\uparrow}(e) = 0 \mod \sqrt{R}$.

This means that $I_2'^{(c^*)}$ is drawn from $\mathsf{Generate}(n,1)$ over the range $\sqrt{R}$. Since TVD is a metric, we have that for $I \sim D_1(\text{ZkC}[\sqrt{R}], n)$ (TVD at most $\frac{n^k}{R}$ from $\mathsf{Generate}(n,1)$), there exists a c^* such that $I_2^{(c^*)}$ is TVD at most $\frac{2n^k}{\sqrt{R}}$ from D_1 — even when dependent on $I_1^{(c^*)}$. Therefore, the TVD of $(I_1^{(c^*)}, I_2^{(c^*)}) = \mathsf{Split}(I)$ to D_1^2 is at most $\frac{4n^k}{\sqrt{R}}$.

Therefore, when $I \sim D_0(\text{ZkC}[R], n)$, we get a list of pairs of instances TVD $\leq 4n^k/\sqrt{R}$ from $D_0(\text{ZkC}[R], n)^2$; the probability that any of these pairs here err is $\leq (\binom{k}{2}+1)\cdot\frac{4n^k}{\sqrt{R}}$ by a union bound. Similarly, when $I \sim D_1(\text{ZkC}[R], n)$, we get there exists a pair in this list of the form $D_1(\text{ZkC}[R], n)^2$; the probability of erring here is $\leq \frac{4n^k}{\sqrt{R}}$.

Therefore, the total error here is $\leq (\binom{k}{2}+1)\cdot\frac{4n^k}{\sqrt{R}}$. □ □

Zero-k-Clique is Splittable Over Any Large Enough Range. Our techniques also generalize to any large enough range (even ones not of the form 4^x). For example, if you believe that the problem is hard only over a prime range, we can prove that as well. As stated, our error is $\binom{k}{2}4^{\binom{k}{2}}3n^k/\sqrt{R} = O(n^k/\sqrt{R})$. For this to be meaningful, $R = \Omega(n^{2k})$, and in our constructions, R is $\Omega(n^{6k})$. We will show in the next section why the zero k-clique problem is still hard over these larger ranges.

Theorem 27. *Zero-k-clique is generalized splittable over any range R, with error* $\leq \binom{k}{2}4^{\binom{k}{2}}3n^k/\sqrt{R}$.

Proof. Given an instance I with range R we will produce $\leq \binom{k}{2}4^{\binom{k}{2}}$ instances, corresponding to guesses over what ranges the clique edge weights fall into.

Take the next smallest power $R' = \max\{2^{2x} | 2^{2x} < R \text{ and } x \in \mathbb{Z}\}$. Now let $c = \lceil R/R' \rceil$. We will now create c subsets of R each of size R'. $S_i = [R'i, R'(i+1)-1]$ for $i \in [0, c-2]$ and $S_{c-1} = [R-R', R-1]$. Note that these subsets completely cover the range $[0, R-1]$ and are each of size $\leq R'$. Let $\Delta_i = R'i$ for $i \in [0, c-2]$ and $\Delta_{c-1} = R - R'$.

Let the partitions of I be $P_1, \ldots, P_k$. Let the set of edges between P_i and P_j be $E_{i,j}$. For all i,j pairs $i \neq j$ we will choose a number between $[0, c-1]$. Call these numbers $g_{i,j}$ and the full list of them $\mathbf{g}$. For all possible choices of $\mathbf{g} \in \mathbb{Z}_c^{\binom{k}{2}}$ and $d \in [0, \binom{k}{2}-1]$ we will generate an instance $I_{\mathbf{g},d}$ over range R' as follows:

For edge set $E_{i,j}$ that isn't $E_{1,2}$, for every edge in that edge set $e \in E_{i,j}$ if the weight of e, $w(e) \in S_{g_{i,j}}$ then set $w_{\mathbf{g},d}(e) = w(e) \mod R'$, if $w(e) \notin S_{g_{i,j}}$ then set $w_{\mathbf{g},d}$ to be a weight chosen uniformly at random from $[0, R'-1]$. Now note that these values are completely uniform over the range from $[0, R'-1]$.

For $E_{1,2}$, for every edge in that edge set $e \in E_{1,2}$ if the weight of e, $w(e) \in S_{g_{1,2}}$ then set $w_{\mathbf{g},d}(e) = w(e) + dR \mod R'$, if $w(e) \notin S_{g_{1,2}}$ then set $w_{\mathbf{g}}$ to be a weight chosen uniformly at random from $[0, R'-1]$. Now note that these values are also completely uniform over the range from $[0, R'-1]$.

If no clique existed in the original instance then the chance that one is produced here is bounded by $n^k/R' \leq n^k 4/R'$ by Lemma 23. Because we make so many queries this chance that any of them induce a clique is $\leq \binom{k}{2} 4^{\binom{k}{2}} n^k 4/R'$.

If the original instance was drawn from $D_1^{zkc}[R, n]$ then by Lemma 24 this is only $\leq n^k/R + n^{k-2}/R$ total variation distance away from the instance generated by choosing each edge at random and then planting a clique. Then the procedure produces uniformly looking edges except for the planted edge. In the generated instance where the original zero clique edge weights are in $\mathbf{g}$ and the zero k-clique sums to dR then the instance $I_{\mathbf{g},d}$ will have that planted edge set to the value such that zero k-clique from the original is a planted instance. So, that produced instance is drawn from a distribution with total variation distance $\leq n^k/R' + n^{k-2}/R'$ from $D_1^{zkc}[R', n]$.

Then we use the splitting procedure from Lemma 25 to generate two instances from each of our generated instances. The probability of a no instance becoming a yes instance is $\leq \binom{k}{2} 4^{\binom{k}{2}} 3n^k/\sqrt{R}$, if there is a yes instance then it will generate a yes instance and have total variation distance at most $\leq \binom{k}{2} 4^{\binom{k}{2}} 3n^k/\sqrt{R}$ from $D_1^{zkc}[\sqrt{R'}, n]^2$. □

4.6 Fine-Grained One-Way Functions

In this section, we give a construction of fine-grained OWFs (FGOWF) based on plantable $T(n)$-ACIH problems. We first show that even though the probability of inversion may be constant (we call this "medium" fine-grained one-way), we can do some standard boosting in the same way weak OWFs can be transformed into strong OWFs in the traditional sense. Then, given such a plantable problem, we will prove that Generate$(n, 1)$ is a medium $T(n)$-FGOWF. Then, from this medium FGOWF, we can compile a strong FGOWF using this boosting trick. Then, since Zero-k-Clique is plantable (see Theorem 25), this implies that assuming Zero-k-Clique is hard yields fine-grained OWFs.

Finally, we discuss the possibility of fine-grained hardcore bits and pseudorandom generators. It turns out that the standard Goldreich-Levin [GL89] approach to creating hardcore bits works in a similar fashion here, but requires some finessing; it will not work for *all* fine-grained OWFs.

We will be using $\tilde{O}(\cdot)$ to suppress sub-polynomial factors of n (as opposed to only $\lg(n)$ factors).

4.6.1 Weak and Strong OWFs in the Fine-Grained Setting

Traditional cryptography has notions of weak and strong OWFs. Weak OWFs can be inverted most of the time, but a polynomial-fraction of the time, they cannot be. These weak OWFs can be compiled into strong OWFs (showing that weak OWFs imply strong OWFs), where there is a negligible chance that the resulting strong OWF is invertible over the choice of inputs.

Here we will briefly define "medium" $T(n)$-FGOWFs, and show how they can imply a "strong" $T(n)$-FGOWF, where "strong" refers to definition 27

Definition 45. *A function f is a medium $T(n)$-FGOWF if there exists a sub-polynomial function $Q(n)$ such that for all $\mathsf{PFT}_{T(n)}$ adversaries $\mathcal{A}$,*

$$\Pr_{x \overset{\$}{\leftarrow} \{0,1\}^n}\Big[\mathcal{A}(f(x)) \in f^{-1}(f(x))\Big] \leq 1 - \frac{1}{Q(n)}.$$

Claim 14. *Medium $T(n)$-FGOWFs imply strong $T(n)$-FGOWFs for any polynomial $T(n)$ that is at least linear.*

Proof. The structure of this proof will follow Yao's original argument augmenting weak OWFs to strong ones. Intuitively, we are able to use this argument because sub-polynomial functions compose well with each other.

Assume for the sake of contradiction that no strong $T(n)$-FGOWF exists. Then, there exists some function $p'(n)$ such that $p'(n)$ is sub-polynomial and for all functions f computable in $T(n)^{1-\epsilon}$ time there exists a $PFT_{T(n)}$ adversary that can invert the function f with probability $1 - \frac{1}{p'(n)}$.

Let f be a medium $T(n)$-FGOWF, where the probability any $\mathsf{PFT}_{T(n)}$ adversary inverts it is $1 - \frac{1}{Q(n)}$. The basic idea will be to produce g that is just a concatenation of many fs, just as in the traditional cryptographic case.

For any positive-integer function $c(n)$, let $g(x_1||\ldots||x_{c(n)}) = f(x_1)||\ldots||f(x_{c(n)})$, where $||$ denotes concatenation. Let $c(n) = 4\left(Q(n)r(n)\right)^2$ (or the ceiling of $4\left(Q(n)r(n)\right)^2$, if not an integer). Where $r(n)$ is a subpolynomial function such that $r(n) \geq p'(c(n) \cdot n)$. Note that $p'(n)$ is subpolynomial, and that as a result some such function $r(n)$ exists. Specifically, setting $r(n) = p'(n^2)$ satisfies both criteria.

Now note that $c(n) \cdot n = \tilde{O}(n)$, and so $T(c(n) \cdot n) = \tilde{O}(T(n))$. Furthermore, g is a $T(c(n) \cdot n) = O(T(n))$-FGOWF since $c(n)$ is subpolynomial.

Now, for sake of contradiction, let $\mathcal{A}$ be a $\mathsf{PFT}_{T(n)}$ such that there exists a sub-polynomial function p' where

$$\Pr\Big[\mathcal{A}(g(x_1||\ldots||x_c)) \in g^{-1}(g(x_1||\ldots||x_c))\Big] \geq \frac{1}{p'(c(n) \cdot n)}.$$

Let $p(n) = p'(c(n) \cdot n)$. Because $c(n) \cdot n = O(n^2)$ and $p'(n)$ is sub-polynomial, $p'(c(n) \cdot n) = p(n)$ is also sub-polynomial. Therefore,

$$\Pr\Big[\mathcal{A}(g(x_1||\ldots||x_c)) \in g^{-1}(g(x_1||\ldots||x_c))\Big] \geq \frac{1}{p(n)}.$$

We will define a $\mathsf{PFT}_{T(n)}$ function $\mathcal{A}_0$ that makes a single call to $\mathcal{A}$: on input $y = f(x)$

1. Choose $i \xleftarrow{\$} [c(n)]$.
2. Let $z_i = y$
3. For all $j \in [c(n)]$, $j \neq i$, $x_j \xleftarrow{\$} \{0,1\}^n$ and $z_j = f(x_j)$.
4. Run $\mathcal{A}$ on $(z_1, \ldots, z_{c(n)})$ to get output $(x_1, \ldots, x_{c(n)})$ if $\mathcal{A}$ succeeds.
5. If $\mathcal{A}$ succeeded, output x_i.

Because all operations in $\mathcal{A}_0$ are either calling $\mathcal{A}$ (once) or take time $O(n \cdot c(n))$,[4] $\mathcal{A}_0$ is a $\mathsf{PFT}_{T(n)}$ algorithm. Now, we will let $\mathcal{B}$ be an algorithm calling $\mathcal{A}_0$ $d(n) = 4c(n)^2(n)p(n)Q(n)$ times, returning a valid inversion of $f(x)$ if $\mathcal{A}$ succeeded at least once.

We will call $x \in \{0,1\}^n$ 'good' if $\mathcal{A}_0$ inverts it with probability at least $\frac{1}{2c^2(n)p(n)}$; x is 'bad' otherwise. Notice that if x is good, then $\mathcal{B}$, which runs $\mathcal{A}$ many times, succeeds with high probability:

$$\Pr\left[\mathcal{B}(f(x)) \text{ fails}|x \text{ is good}\right] \leq \left(1 - \frac{1}{2c^2(n)p(n)}\right)^{d(n)} \sim e^{-2Q(n)} < \frac{1}{2Q(n)}.$$

We will show that there are at least $2^n(1 - \frac{1}{2p(n)})$ good elements.

Claim 15. *There are at least $2^n(1 - \frac{1}{2p(n)})$ good elements.*

Proof. For a contradiction, assume there are at least $2^n(\frac{1}{2p(n)})$ bad elements. We will end up contradicting the inversion probability of $\mathcal{A}$ (which is at least $1/p(n)$). For notation, let $\mathbf{x} = (x_1, \ldots, x_{c(n)}) \in \{0,1\}^{n \cdot c(n)}$, and $\mathbf{x}$ will be chosen uniformly at random over the input space.

$$\begin{aligned}\Pr_{\mathbf{x}}[\mathcal{A}(\mathbf{z} = g(\mathbf{x})) \text{ succeeds}] = \Pr[&\mathcal{A}(\mathbf{z}) \text{ succeeds} \wedge \exists \text{bad } x_j] \\ &+ \Pr\left[\mathcal{A}(\mathbf{z}) \text{ succeeds} \wedge \mathrm{x}_j \text{ good } \forall j \in [c(n)]\right]\end{aligned}$$

Now, for all $j \in [c(n)]$,

$$\begin{aligned}\Pr_{\mathbf{x}}[\mathcal{A}(\mathbf{z}) \text{ succeeds} \wedge x_j \text{ is bad}] &\leq \Pr_{\mathbf{x}}[\mathcal{A}(\mathbf{z}) \text{ succeeds}|x_j \text{ is bad}] \\ &\leq c(n) \Pr_{\mathbf{x}}[\mathcal{A}_0(f(x_j)) \text{ succeeds}|x_j \text{ is bad}] \\ &\leq \frac{c(n)}{2c^2(n)p(n)} = \frac{1}{2c(n)p(n)}\end{aligned}$$

So, if we just union bound over all j, we get

$$\Pr_{\mathbf{x}}[\mathcal{A}(\mathbf{z}) \text{ succeeds} \wedge \text{some } x_j \text{ are bad}] \leq \sum_{j=1}^{c(n)} \Pr_{\mathbf{x}}[\mathcal{A}_0(f(x_j)) \text{ succeeds} \wedge x_j \text{ is bad}] \leq \frac{1}{2p(n)}.$$

And one more quick upper bound yields

[4]It does not make much sense for $T(n)$ to be sublinear for our contexts

$$\begin{aligned}\Pr[\mathcal{A}(\mathbf{z}) \text{ succeeds} \wedge \text{all } x_j \text{ are good}] &\leq \Pr_{\mathbf{x}}[\text{all } x_j \text{ good}] \\ &< \left(1 - \frac{1}{2p(n)}\right)^{c(n)} \\ &\leq e^{-2(Q(n)r(n))^2/p(n)} \\ &\leq e^{-2(Q(n))^2 \cdot p(n)} < \frac{1}{2p(n)}.\end{aligned}$$

Finally, this yields the contradiction to the claim that there are at least $2^n(\frac{1}{2p(n)})$ bad elements:

$$\Pr_{\mathbf{x}}[\mathcal{A}(\mathbf{z}) \text{ succeeds}] < \frac{1}{p(n)}.$$

$\square$ $\square$

Note that $p(n) \geq Q(n)$ because $1/Q(n)$ is the maximum probability of inverting a single copy of $f(\cdot)$, where as $1/p(n)$ is assumed (for contradiction) to be the probability that a function inverts $c(n)$ copies of $f(\cdot)$ simultaneously. So, there are at most $2^n(\frac{1}{2Q(n)})$ bad elements.

Now that we know there is a high probability that we hit a good x, we can finish the rest of this proof.

$$\begin{aligned}\Pr_x[\mathcal{B}(f(x)) \text{ fails}] &= \Pr[\mathcal{B}(f(x)) \text{ fails}|x \text{ is good}] \Pr_x[x \text{ is good}] \\ &\qquad + \Pr[\mathcal{B}(f(x)) \text{ fails}|x \text{ is bad}] \Pr_x[x \text{ is bad}] \\ &\leq \Pr[\mathcal{B}(f(x)) \text{ fails}|x \text{ is good}] \Pr_x[x \text{ is good}] + \Pr_x[x \text{ is bad}] \\ &\leq \frac{1}{2Q(n)}\left(1 - \frac{1}{2Q(n)}\right) + \frac{1}{2Q(n)} < \frac{1}{Q(n)}\end{aligned}$$

Thus, the chance that $\mathcal{B}$ actually has of inverting f is strictly greater than $1 - \frac{1}{Q(n)}$, contradicting the claim that f was medium-hard with respect to $Q(n)$. $\square$ $\square$

Weaker Fine-Grained OWFs. Now, because we are in the fine-grained setting, we can talk about gaps. There is a notion of weak-OWFs in cryptography where we can say if there exists *any* polynomial such that we can invert with probability $1-1/\text{poly}$, we can construct strong OWFs. We want a similar notion for fine-grained OWFs. Here we can't just choose any polynomial — we have to choose a polynomial that respects the gap.

Formally, for an $T(n)$-FGOWF f that has $\mathsf{PFT}_{T(n)}$ adversaries inverting it with probability $1 - 1/P(n)$ for some $P(n)$, we can get that a $\mathsf{PFT}_{T(n)}$ adversary can invert f with probability $(1 - 1/P(n))^{c(n)}$. Now, as long as there exists δ' such that $T(n)^{1-\delta}P(n) = T(n)^{1-\delta'}$, there is still a gap ($\delta' < \delta$) even if we compute f $P(n)$ times to evaluate f. Therefore, we are able to get a strong fine-grained OWF from a weak one, as long as it's not *too* weak.

4.6.2 Building Fine-Grained OWFs from Plantable Problems

Here we show that one can generate fine-grained one way functions from plantable problems. Recall the definition of Plantable states that there exists an algorithm $\mathsf{Generate}(n, b)$ where when $b = 0$, an instance of the problem *without* a solution is generated, and when $b = 1$, an instance of a problem *with one* solution is generated with probability at least $1 - \epsilon$. This probabilistic element, ϵ, is actually a bound on the total variation distance of the distributions we are actually aiming to sample from: $\mathsf{Generate}(n, 0)$ and $\mathsf{Generate}(n, 1)$ have total variation distance at most ϵ from $D_0(P, n)$ and $D_1(P, n)$ respectively.

Theorem 28. *If there exists a Plantable $T(n)$-ACIH problem where $G(n)$ is $\mathsf{PFT}_{T(n)}$ with error some constant $\epsilon < 1/3$, then $T(n)$-FGOWFs exist.* [5]

Proof. Let P be a Plantable $T(n)$-ACIH problem where $G(n) = T(n)^{1-\delta}$ for some constant $\delta > 0$. So, the (randomized) algorithm $\mathsf{Generate}(n, 1)$ is $\mathsf{PFT}_{T(n)}$ and outputs an instance I that has at least one solution — we write this as $\mathsf{Generate}(n, 1; r)$ when explicitly noting which randomness was used.

We want to show that being able to invert $\mathsf{Generate}(n, 1; r)$, over the distribution from r, in a fine-grained sense, as solving the ACIH problem P. Let ϵ be the upper bound on the total variation distance between $\mathsf{Generate}(n, 1)$ and $D_1(P, n)$, as per Definition 34.

For sake of contradiction, assume that no *medium* $T(n)$-FGOWF exist. So, we can invert $\mathsf{Generate}(n, 1)$ with any probability $1 - \frac{1}{Q(n)}$ for any sub-polynomial $Q(n)$. Let $\mathcal{A}$ be a $\mathsf{PFT}_{T(n)}$ algorithm that inverts $\mathsf{Generate}(n, 1)$ with probability $\gamma > 1 - \frac{1}{\log(n)}$ (note that $\log(n)$ is significant). We will show that this violates the assumption that P is a $T(n)$-ACIH problem.

We now construct a $\mathsf{PFT}_{T(n)}$ algorithm $\mathcal{B}$ that distinguishes between $I \sim D_0(P, n)$ and $I \sim D_1(P, n)$ with probability greater than $2/3$, violating the hardness assumption on P.

- Given I from distribution D, $\mathcal{B}$ gives I to $\mathcal{A}$.
- $\mathcal{A}$ outputs r.
- If $\mathsf{Generate}(n, 1; r) == I$, output 1. Otherwise, output 0.

We will now compute the probability that $\mathcal{B}$ distinguishes between inputs from D_1 and D_0. Recall that D is just sampling with D_0 with probability $1/2$, and otherwise samples from D_1. For the sake of brevity let the notation $I \in D_0$ and $I \in D_1$ convey

[5] We would like to thank Chris Brzuska and his reading group for finding a bug in the original version of this proof. To correct this bug, we added the word ‘constant’ to the theorem statement, removed our severe abuse of notation, and fixed the proof accordingly.

that I is in the support of D_0 and the support of D_1 respectively. We have

$$\begin{aligned}\Pr_{I\sim D}[\mathcal{B}(I) \text{ distinguishes } D_0 \text{ from } D_1] &= \Pr_{I\sim D_1}[\mathcal{B}(I)=1]\cdot \Pr_{I\sim D}[I\in D_1] \\ &\quad + \Pr_{I\sim D_0}[\mathcal{B}(I)=0]\cdot \Pr_{I\sim D}[I\in D_0] \\ &= \frac{1}{2}\Pr_{I\sim D_1}[\mathcal{B}(I)=1] + \frac{1}{2}\Pr_{I\sim D_0}[\mathcal{B}(I)=0].\end{aligned}$$

First, we note that $\Pr_{I\sim D_0}[\mathcal{B}(I)=0]=1$ because $\mathsf{Generate}(n,1;r)$ is guaranteed to produce a witness for all randomness r. This means that any I sampled from D_0 is not in the image of $\mathsf{Generate}(n,1)$, and therefore, $\mathcal{A}$ cannot produce a valid inverse.

Then, we use the fact that D_1 is close in total variation distance to $\mathsf{Generate}(n,1)$ to show that $\Pr_{I\sim D_1}[\mathcal{B}=1]\ge \gamma-2\epsilon$. Let p_{G_1} be the pdf of $\mathsf{Generate}(n,1)$ and p_{D_1} be the pdf of D_1. Let $\mathcal{I}$ be the set of all instances of the problem. Let $S=\{I\in \mathrm{Im}(\mathsf{Generate}(n,1)) : \mathsf{Generate}(n,1;\mathcal{A}(I))=I\}$ be the set of instances produced by $\mathsf{Generate}$ that $\mathcal{A}$ can successfully invert. Recall that $\mathrm{TVD}(D_1,\mathsf{Generate}(n,1))\le\epsilon$ means $\sum_{I\in\mathcal{I}}|p_D(I)-p_G(I)|\le 2\epsilon$ by the definition of TVD.

$$\begin{aligned}2\epsilon &\ge \sum_{I\in\mathcal{I}}|p_{D_1}(I)-p_{G_1}(I)| \\ &\ge \sum_{I\in S}|p_{D_1}(I)-p_{G_1}(I)| \\ &\ge \sum_{I\in S}[p_{D_1}(I)] - \sum_{I\in S}[p_{G_1}(I)]\,.\end{aligned}$$

This implies $\sum_{I\in S}[p_{G_1}(I)]\ge\sum_{I\in S}[p_{D_1}(I)]-2\epsilon$, and therefore $\Pr_{I\sim D_1}[\mathcal{B}=1]\ge\gamma-2\epsilon$.

Notice that since ϵ is constant and less than $\frac{1}{3}$, $\frac{1}{3}-\alpha=\epsilon$ for some constant $\alpha>0$. Putting this together, we have that

$$\begin{aligned}\Pr_{I\sim D}[\mathcal{B}(I) \text{ distinguishes } D_0 \text{ from } D_1] &\ge \frac{1}{2}\cdot(\gamma-2\epsilon)+\frac{1}{2} \\ &= \frac{\gamma}{2}-\epsilon+\frac{1}{2} \\ &= 1-\frac{1}{2\log(n)}-\epsilon \\ &\ge 1-\frac{1}{2\log(n)}-\left(\frac{1}{3}-\alpha\right) \\ &> \frac{2}{3}.\end{aligned}$$

Note that $\frac{1}{2\log(n)}$ is less (asymptotically) that any constant α, the sum of these terms is greater than $\frac{2}{3}$.

So, assuming P is $T(n)$-ACIH, i.e. no adversary has better than a constant chance less than 1 of being able to invert $\mathsf{Generate}(n,1;r)$, then $\mathsf{Generate}$ is a medium $T(n)$-FGOWF. By Claim 14, this implies strong $T(n)$-FGOWFs exist. $\square$ $\square$

Note that k-Sum-R and Zero-k-Clique-R are plantable with error less than $1/3$ the when $R>6n^k$ by Theorem 25 and Theorem 24, these are both plantable and therefore can be used to build these fine-grained OWFs.

4.6.3 Fine-Grained Hardcore Bits and Pseudorandom Generators

One way functions serve as the building block for a lot of symmetric encryption, and are (usually) implied by any other cryptographic primitive, from collision-resistant hash functions to symmetric-key encryption, to any flavor of public key encryption, and so on. The next step to building more cryptographic primitives with one-way functions is to see if we can use them to construct pseudorandom generators. While we do not yet have a construction of a fine-grained pseudorandom generator that can generate some sub-polynomial many pseudorandom bits[6], we take the first steps, showing how to get hardcore bits.

Definition 46. *A function b is a fine-grained hardcore (FGHC) predicate for a $T(n)$-FGOWF if for all $\mathsf{PFT}_{T(n)}$ adversaries $\mathcal{A}$,*

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(f(x)) = b(x)] \leq \frac{1}{2} + \mathsf{insig}(n).$$

Recall that in traditional cryptography, any OWF implies the existence of another OWF with a hardcore bit with the Goldreich-Levin (GL) construction [GL89]. The bad news: the GL construction required a security reduction with $O(n)$ evaluations of the one-way function. Given how we define problems to be $T(n)$-ACIH hard, this security reduction would not be $\mathsf{PFT}_{T(n)}$.

Theorem 29 (Fine-Grained Goldreich-Levin)**.** *Let f be an $T(n)$-$\mathsf{FG\text{-}OWF}$ acting on strings (x, y), where $|x| = n$ and $|y| = Q(n)$ for some subpolynomial Q, and assume there exists a $\mathsf{PFT}_{T(n)}$ algorithm $\mathcal{L}$ such that*

$$\Pr[\mathcal{L}(f(x, y), y) \in f^{-1}(f(x, y))] \geq \mathsf{sig}(n).$$

Then, the function $f' : (x, y, r) \mapsto f(x, y)||r$ where $|r| = |y|$ has the hardcore bit $y \cdot r$.

Proof. Here we just trace through the GL reduction and show that as long as $|y|$ is sub-polynomial, the reduction will go through.

First, the size of y, $Q(n)$, cannot be any subpolynomial, it must be large enough so that it is as hard to guess y as it is to invert f (because guessing y yields a significant chance of inverting f). If f is $T(n)$-hard to invert, then the time it takes to randomly guess bits, $2^{Q(n)}$, must be at least $T(n)$. Since $T(n)$ is at least linear, we can assume $Q(n) \geq \log(n)$.

For a contradiction, assume that a $\mathsf{PFT}_{T(n)}$ adversary $\mathcal{A}$ has a significant advantage ϵ in determining $r \cdot y$ when given $f(x, y), r$. We will show this implies $\mathcal{B}$, a $\mathsf{PFT}_{T(n)}$ algorithm using $\mathcal{A}$, can invert f with significant probability.

$\mathcal{B}$ behaves as follows with parameter $m = 2Q(n)/\epsilon$ on input $x' = f(x, y)$:

- For every $i \in [Q(n)]$:

[6]Note that due to the nature of being fine-grained, we cannot generate polynomially-many bits without additional assumptions

1. Choose $\log(m)$ pairs $(b_1, r_1), \ldots, (b_{\log(m)}, r_{\log(m)}) \xleftarrow{\$} \{0,1\} \times \{0,1\}^{Q(n)}$.
2. For every I in the powerset of $[\log(m)]$, let $b'_I = \sum_{j \in I} b_j \mod 2$.
3. For every I in the powerset of $[\log(m)]$, let $r_I = \sum_{j \in I} r_j \mod 2$.
4. For every I in the powerset of $[\log(m)]$,
   - Let $s_I \leftarrow e_i \oplus r_I$ where e_i is the i^{th} standard basis vector (e_i is all zeros except for one 1 in the i^{th} index).
   - Let $g_I \leftarrow b'_I \oplus \mathcal{A}(x'||s_I)$
5. Let $z_i =$ the Majority bit over all $2^{\log(m)}$ bits g_I.

- Output $z = z_1, \ldots, z_{Q(n)}$.

First, consider the set $S = \left\{(x,y) | \Pr_r[\mathcal{A}(f(x,y)||r) = r \cdot y] \geq \frac{1}{2} + \frac{\epsilon}{2}\right\}$. A quick calculation yields $|S| > \epsilon \cdot 2^{n-1}$.

So, assume that our input $(x,y) \in S$. Now, assume that every pair we chose in step 1 has the property $b_i = r_i \cdot y$ (we correctly guessed the bit in question). This event occurs with probability $1/m$.

Next, notice that each pair of s_I, and s_J $(I \neq J)$ are independent, and so the whole set is pairwise independent. So, if $(x,y) \in S$, $\mathcal{A}$ will return the correct bit given $f(x,y)||r_I$ at least an $\epsilon/2$-fraction of the time for independent r's. Because of the pairwise independence, a Chebyshev bound yields $\mathcal{A}$ will return the correct bit a majority of the time after the m queries (so $\mathsf{Majority}(\{g_I\})$ outputs the correct bit y_i) with probability at least $1 - \frac{1}{m(\epsilon/2)^2}$.

Finally, we put all of these pieces together to get

$$\begin{aligned}
\Pr[\mathcal{B}(f(x,y)) = y] &\geq \Pr[\mathcal{B}(f(x,y)) = y | (x,y) \in S] \cdot \frac{\epsilon}{2} \\
&= \frac{\epsilon}{2}\left(1 - \Pr[\exists i \text{ s.t. } y_i \neq z_i | (x,y) \in S]\right) \\
&\geq \frac{\epsilon}{2}\left(1 - Q(n)\Pr[y_i \neq z_i | (x,y) \in S]\right) \\
&\geq \frac{\epsilon}{2}\left(1 - Q(n)\Pr[y_i \neq z_i | (x,y) \in S \wedge \text{guessed all } b_i \text{ correctly}] \cdot \frac{1}{m}\right) \\
&\geq \frac{\epsilon}{2}\left(1 - \frac{Q(n)}{m} \cdot \frac{1}{m(\epsilon/2)^2}\right) \\
&= \frac{\epsilon}{2} - \frac{4Q(n)}{m^2\epsilon}
\end{aligned}$$

Recall we set $m = 2Q(n)/\epsilon$, and since ϵ is significant and Q is subpolynomial, m is also subpolynomial. Importantly, because $\mathcal{B}$ runs in $O(Q(n) \cdot mT(n)^{1-\delta})$, $\mathcal{B}$ is $\mathsf{PFT}_{T(n)}$.

Therefore, the probability that $\mathcal{B}$ succeeds in finding y_i (and hence inverting $f(x,y)$ with significant probability), with probability $\frac{\epsilon}{2} - \frac{4Q(n)\epsilon}{4Q^2(n)} \geq \frac{\epsilon}{2} - \frac{4\epsilon}{Q(n)}$. Recall that $Q(n)$ is at least linear in n, and so we can assume $Q(n) > 16$. This implies the probability $\mathcal{B}$ succeeds is $\frac{\epsilon}{4}$.

Because ϵ is significant, $\mathcal{B}$ breaks the fine-grained one-wayness of f. This is a contradiction. Therefore, ϵ must be insignificant. □ □

Hardcore bits from k-Sum and Zero-k-Clique

For both of these problems, planting a solution is exactly choosing some number of values (k for k-Sum, and the edge weights of a k-clique for Zero-k-Clique) and changing one of them so that the values now give a solution.

Corollary 9. *Assuming either the Weak k-Sum hypothesis or weak Zero-k-Clique hypothesis, there exist FGOWFs with fine-grained hardcore bits.*

Proof. This is straightforward due to the nature of planting for both of these hypotheses. Informally, planting for these problems is choosing a location within the given instance to put a solution. If an adversary learns where that solution is supposed to be, generating an instance without that specific solution is easy.

First, let's prove this for k-Sum. The reason k-Sum is plantable is because $\mathsf{Generate}(n, 1)$ chooses k indices at random in the k-Sum instance, and then changes the value one of them to make those k instances form a solution the k-Sum. This randomness requires specifying k instances out of kn, and an edge-weight. Let y be the $k \log(n)$ bits required to describe the k locations of the solution; y is part of the total randomness r used in $\mathsf{Generate}(n, 1)$. Without loss of generality, we can write $r = y||r'$. Let $f'(y||r', s) = \mathsf{Generate}(n, 1; y||r')||s$. Since $|y|$ is sub-polynomial, by Theorem 29, the bit $y \cdot s$ is hardcore for f'.

Now, let's make the same argument for Zero-k-Clique. As before, $\mathsf{Generate}(n, 1; r)$ can be written as $\mathsf{Generate}(n, 1; y||r')$ where y is the location of the zero k-clique generated. This location is just $k \cdot \log(n)$ bits; one coordinate from n for each of the k partitions in the graph. Therefore, we can define $f'(y||r', s) = \mathsf{Generate}(n, 1; y||r')||s$, which, by Theorem 29, has the hardcore bit $y \cdot s$. $\square$

4.7 Fine-Grained Key Exchange

Now we will explain a construction for a *key exchange* using general distributions. We will then specify the properties we need for problems to generate a secure key exchange. We will finally generate a key exchange using the strong Zero-k-Clique hypothesis.

Before doing this, we will define a class of problems as being Key Exchange Ready (KER).

Definition 47 (Key Exchange Ready (KER)). *A problem P is $\ell(n)$-KER with generate time $G(n)$, solve time $S(n)$ and lower bound solving time $T(n)$ if*

- *there is an algorithm which runs in $\tilde{\Theta}(S(n))$ time that determines if an instance of P of size n has a solution or not,*
- *the problem is $(\ell(n), \delta_{LH})$-ACLH where $\delta_{LH} \leq \frac{1}{34}$,*
- *is Generalized Splittable with error $\leq 1/(128\ell(n))$ to the problem P' and,*
- *P' is plantable in time $G(n)$ with error $\leq 1/(128\ell(n))$.*

- $\ell(n)T(n) \in \tilde{\omega}\left(\ell(n)G(n) + \sqrt{\ell(n)}S(n)\right)$, *and*

- *there exists an n' such that for all $n \geq n'$, $\ell(n) \geq 2^{14}$.*

4.7.1 Description of a Weak Fine-Grained Interactive Key Exchange

The high level description of the key exchange is as follows. Alice and Bob each produce $\ell(n) - \sqrt{\ell(n)}$ instances using $\mathsf{Generate}(n, 0)$ and $\sqrt{\ell(n)}$ generate instances with $\mathsf{Generate}(n, 1)$. Alice then shuffles the list of $\ell(n)$ instances so that those with solutions are randomly distributed. Bob does the same thing (with his own private randomness). Call the set of indices that Alice chooses to plant solutions S_A and the set Bob picks S_B. The likely size of $S_A \cap S_B$ is 1. The index $S_A \cap S_B$ is the basis for the key.

Alice determines the index $S_A \cap S_B$ by brute forcing all problems at indices S_A that Bob published. Bob can brute force all problems at indices S_B that Alice published and learn the set $S_A \cap S_B$.

If after brute forcing for instances either Alice or Bob find a number of solutions not equal to 1 then they communicate this and repeat the procedure (using interaction). They only need to repeat a constant number of times.

More formally our key exchange does the following:

Construction 30 (Weak Fine-Grained Interactive Key Exchange)**.** *A fine-grained key exchange for exhanging a single bit key.*

- $\mathsf{Setup}(1^n)$*: output* $\textsc{mpk} = (n, \ell(n))$ *and* $\ell(n) > 2^{14}$.

- *KeyGen($\textsc{mpk}$): Alice and Bob both get parameters (n, ℓ).*

  – *Alice generates a random $S_A \subset [\ell]$, $|S_A| = \sqrt{\ell}$. She generates a list of instances $\mathbf{I}_A = (I_A^1, \ldots, I_A^\ell)$ where for all $i \in S_A$, $I_i = \mathsf{Generate}(n, 1)$ and for all $i \notin S_A$, $I_A^i = \mathsf{Generate}(n, 0)$ (using Alice's private randomness). Alice publishes $\mathbf{I}_A$ and a random vector $\mathbf{v} \xleftarrow{\$} \{0, 1\}^{\log \ell}$.*

  – *Bob computes $\mathbf{I}_B = (I_B^1, \ldots, I_B^\ell)$ similarly: generating a random $S_B \subset [\ell]$ of size $\sqrt{\ell}$ and for every instance $I_j \in \mathbf{I}_B$, if $j \in S_B$, $I_j = \mathsf{Generate}(n, 1)$ and if $j \notin S_B$, $I_j = \mathsf{Generate}(n, 0)$. Bob publishes $\mathbf{I}_B$.*

- *Compute shared key: Alice receives $\mathbf{I}_B$ and Bob receives $\mathbf{I}_A$.*

  – *Alice computes what she believes is $S_A \cap S_B$: for every $i \in S_A$, she brute force checks if I_B^i has a solution or not. For each i that does, she records in list L_A.*

  – *Bob computes what he thinks to be $S_B \cap S_A$: for every $j \in S_B$, he checks if I_A^j has a solution. For each that does, he records it in L_B.*

- *Check: Alice takes her private list L_A: if $|L_A| \neq 1$, Alice publishes that the exchange failed. Bob does the same thing with his list L_B: if $|L_B| \neq 1$, Bob publishes that the exchange failed. If either Alice or Bob gave or recieved a failure, they both know, and go back to the* KeyGen *step.*

  If no failure occurred, then $|L_A| = |L_B| = 1$. Alice interprets the index $i \in L_A$ as a vector and computes $i \cdot \mathbf{v}$ as her key. Bob uses the index in $j \in L_B$ and also computes $j \cdot \mathbf{v}$. With high probability, $i = j$ and so the keys are the same.

4.7.2 Correctness and Soundness of the Key Exchange

We want to show that with high probability, once the key exchange succeeds, both Alice and Bob get the same shared index.

Lemma 26. *After running construction 30, Alice and Bob agree on a key k with probability at least $1 - \frac{1}{10{,}000\ell e}$.*

Proof. Since we are allowing interaction, the only way Alice and Bob can fail is if one of Alice's $\mathsf{Generate}(n, 0)$ contains a solution that overlaps with S_B, one of Bob's $\mathsf{Generate}(n, 0)$ contains a solution that overlaps with S_A, *and* $S_A \cap S_B = \varnothing$.

First, let's compute $p_0 = \Pr[S_A \cap S_B = \varnothing]$. We have $p_0 = \prod_{i=0}^{\sqrt{\ell}} \left(\frac{\ell - \sqrt{\ell} - i}{\ell}\right)$, the chance that every time Bob chooses an element for S_B, he does not choose an element in S_A. Rearranging this expression, we have

$$p_0 = \prod_{i=0}^{\sqrt{\ell}-1} \left(\frac{\ell - \sqrt{\ell} - i}{\ell}\right) = \prod_{i=0}^{\sqrt{\ell}-1} (1 - \frac{\sqrt{\ell} + i}{\ell}) \leq \prod_{i=0}^{\sqrt{\ell}-1} \left(1 - \frac{1}{\sqrt{\ell}}\right) = \left(1 - \frac{1}{\sqrt{\ell}}\right)^{\sqrt{\ell}} \approx \frac{1}{e}$$

Now, assuming that S_A and S_B do not intersect, we need to compute the probability that *both* Alice and Bob see an incorrectly generated instance (generated by $\mathsf{Generate}(n, 0)$, but contains a solution). Let $\epsilon_{plant} \leq \frac{1}{100\ell}$ be the planting error. Since there is no overlap between S_A and S_B, these probabilities are independent. The probability that S_A overlaps is at most $\sqrt{\ell}\epsilon_{plant} \leq \frac{1}{100\sqrt{\ell}}$ via a union bound over all $\sqrt{\ell}$ instances corresponding to the indices in S_A. Therefore, the probability that this happens for both Alice and Bob is at most $\frac{1}{10{,}000\ell} = \left(\frac{\sqrt{\ell}}{10{,}000\ell}\right)^2$.

Thus, the probability that this event occurs is at most $\frac{1}{10{,}000\ell e}$, and it is the only way the protocol ends without Alice and Bob agreeing on a key.

Therefore, the probability Alice and Bob agree on a key at the end of the protocol is $1 - \frac{1}{10{,}000\ell e}$. □ □

We next show that the key-exchange results in gaps in running time and success probability between Alice and Bob and Eve. Then, we will show that this scheme can be boosted in a fine-grained way to get larger probability gaps (a higher chance that Bob and Alice exchange a key and lower chance Eve gets it) while preserving the running time gaps.

First, we need to show that the time Alice and Bob take to compute a shared key is less (in a fine-grained sense) than the time it takes Eve, given the public transcript, to figure out the shared key. This includes the number of times we expect Alice and Bob to need to repeat the process before getting a usable key.

Time for Alice and Bob.

Lemma 27. *If a problem P is $\ell(n)$-KER with plant time $G(n)$, solve time $S(n)$ and lower bound $T(n)$ when $\ell(n) > 100$, then Alice and Bob take expected time $O(\ell G(n) + \sqrt{\ell} S(n))$ to run the key exchange.*

Proof. First, we will compute a bound on the number of times Alice and Bob need to repeat the key exchange before they match on exactly one index. Alice and Bob repeat any time there isn't exactly one overlap between S_A and S_B or the key exchange fails, as described in the proof of Lemma 26. Since the probability of the bad event happening is small, $\leq 1/(10,000e\ell)$, we will ignore it. Instead, saying

$$\begin{aligned}&\Pr[\text{Key Exchange Stops after this round}]\\ &\quad = \Pr[\text{bad event}] + \Pr[\text{Exactly one overlap}|\text{ no bad event}] \cdot \Pr[\text{no bad event}]\\ &\quad \geq \frac{1}{2}\Pr[\text{Exactly one overlap}|\text{ no bad event}] = \Pr[\text{ Exactly one overlap}]/2.\end{aligned}$$

Computing the probability that there is exactly one overlap. Let p_0 and p_1 be the probability that there are zero overlaps and exactly 1 overlap respectively. First, using similar techniques as in the proof of Lemma 26, we show that $p_0 \geq \frac{1}{e^2}$

$$p_0 = \prod_{i=0}^{\sqrt{\ell}-1}(1 - \frac{\sqrt{\ell}+i}{\ell}) \geq \prod_{i=0}^{\sqrt{\ell}-1}\left(1 - \frac{2\sqrt{\ell}}{\ell}\right) = \left(1 - \frac{2}{\sqrt{\ell}}\right)^{\sqrt{\ell}} \approx \frac{1}{e^2}.$$

A combinatorial argument also tells us that $p_0 = \binom{\ell-\sqrt{\ell}}{\sqrt{\ell}}/\binom{\ell}{\sqrt{\ell}}$ since there are $\binom{\ell}{\sqrt{\ell}}$ possible ways to choose S_A independent of S_B, but if we want to ensure no overlap between S_A and S_B, we need to avoid the $\sqrt{\ell}$ locations in S_B, hence $\binom{\ell-\sqrt{\ell}}{\sqrt{\ell}}$ choices for S_A. Then, we have $p_1 = \sqrt{\ell} \cdot \binom{\ell-\sqrt{\ell}}{\sqrt{\ell}-1}/\binom{\ell}{\sqrt{\ell}}$ because there are $\sqrt{\ell}$ places to choose from to overlap S_A with S_B, and then we must avoid the $\sqrt{\ell}-1$ locations in S_B for the rest of the $\sqrt{\ell}$ elements in S_A.

Now we will compute a bound on p_1 by first showing $\frac{p_1}{p_0} \geq 1$:

$$\begin{aligned}\frac{p_1}{p_0} &= \frac{\sqrt{\ell}\binom{\ell-\sqrt{\ell}}{\sqrt{\ell}-1}}{\binom{\ell}{\sqrt{\ell}}} \cdot \frac{\binom{\ell}{\sqrt{\ell}}}{\binom{\ell-\sqrt{\ell}}{\sqrt{\ell}}}\\ &= \frac{\sqrt{\ell}(\ell-\sqrt{\ell})!}{(\sqrt{\ell}-1)!(\ell-2\sqrt{\ell}+1)!} \cdot \frac{(\sqrt{\ell})!(\ell-2\sqrt{\ell})!}{(\ell-\sqrt{\ell})!}\\ &= \frac{(\sqrt{\ell})^2}{\ell-2\sqrt{\ell}+1} = \frac{\ell}{\ell-2\sqrt{\ell}+1} \geq 1\end{aligned}$$

Now, we have that $p_1 = \frac{p_1}{p_0} \cdot p_0 \geq 1 \cdot \frac{1}{e^2} \geq 1/10$.

Finally, putting this all together, the probability that Alice and Bob stop after a round of the protocol is at least $\frac{1}{20}$. And so, we expect Alice and Bob to stop after a constant number of rounds. Each round consists of calling $\mathsf{Generate}$ ℓ times and solving $\sqrt{\ell}$ instances; so, each round takes $\ell G(n) + \sqrt{\ell} S(n)$ time. Therefore, Alice and Bob take $O(\ell G(n) + \sqrt{\ell} S(n))$. □ □

Time for Eve.

Lemma 28. *If a problem P is $\ell(n)$-KER with plant time $G(n)$, solve time $S(n)$ and lower bound $T(n)$ when $\ell(n) \geq 2^{14}$, then an eavesdropper Eve, when given the transcript $\mathbf{I}_T$, requires $\tilde{\Omega}(\ell(n)T(n))$ time to solve for the shared key with probability $\frac{1}{2} + \mathsf{sig}(n)$.*

Proof. This proof requires two steps: first, if Eve can figure out the shared key in time $\mathsf{PFT}_{\ell(n)T(n)}$ time with advantage δ_{Eve}, then she can also figure out the index in $\mathsf{PFT}_{\ell(n)T(n)}$ time with probability $\delta_{Eve}/4$. Then, if Eve can compute the index with advantage $\delta_{Eve}/4$, we can use Eve to solve the list-version of P in $\mathsf{PFT}_{\ell(n)T(n)}$ with probability $\delta_{Eve}/16$, which is a contradiction to the list-hardness of our problem.

Finding a bit finds the index. This is just the Goldreich-Levin (GL) trick used in classical cryptography to convert OWFs to OWFs with a hardcore bit. We have to be careful in this scenario since the security reduction for GL requires polynomial overhead ($O(N^2)$). However, this is only because we are trying to find N bits based off of linear combinations of those bits. If instead we were trying to find $\operatorname{poly}\log N$ bits, we would only require $\operatorname{poly}\log N$ time to do so with this trick. $i \in \ell(n)$ is an index, so $|i| = \log(\ell(n))$. Because $\ell(n)$ is polynomial in n, $|i|$ is polynomial in the log of n, therefore, using the same techniques as used in the proof of Theorem 29, being able to determine $i \oplus r$ with δ advantage allows us to determine i in the same amount of time, with probability $\delta/4$.

Finding the index solves P Now, let $\mathbf{I} = (I_1, \ldots, I_\ell)$ be an instance of the list problem for P: for a random index i, $I_i \leftarrow D_1$, and for all other $j \neq i, I_j \leftarrow D_0$. Because P is generalized splittable, we can take every I_i and turn it into a list of m instances. With probability $1 - \ell\epsilon_{split}$, we turn $\mathbf{I}$ to m different instances: for every $c \in [m]$, $\mathbf{I}^{(c)} = ((I_1^{(1,c)}, I_1^{(2,c)}), \ldots (I_\ell^{(1,c)}, I_\ell^{(2,c)}))$. For all c and $j \neq i$, $(I_j^{(1,c)}, I_j^{(2,c)}) \sim D_0 \times D_0$, and for at least one $c^* \in [m]$, $(I_i^{(1,c^*)}, I_i^{(2,c^*)}) \sim D_1 \times D_1$. Because P is plantable, for $\sqrt{\ell} - 1$ random coordinates $h \in [\ell]$, for all $c \in [m]$, we will change $I_h^{(1,c)}$ to $I_h'^{(1,c)} \sim \mathsf{Generate}(n, 1)$, and for $\sqrt{\ell} - 1$ random coordinates $g \in [\ell]$, disjoint from all h's, for every $c \in [m]$, we will similarly plant solutions in the second list, changing $I_g^{(2,c)}$ to $I_g'^{(2,c)} \sim \mathsf{Generate}(n, 1)$.

Note that there are ℓ instances and Eve returns a single index. We can verify the correctness by brute forcing a single instance in the list instance. When ℓ is polynomial in n then the time to brute force is polynomially smaller than the time required to solve the list instance. We will need to brute force m of these instances (one for each of the m produced pairs of lists). When $\ell/m = n^{-\Omega(1)}$, the total time for all the brute forces is polynomially smaller than the time required for solving a single list instance. This is how we deal with the "dummy" instances produced with by the splittable construction.

Now, notice that we have changed the list version of the problem into m different lists of pairs of instances, $\left\{\left(\mathbf{I}_1^{(c)}, \mathbf{I}_2^{(c)}\right)\right\}_{c\in[m]}$, and there exists a c^* such that the c^*'th list is distributed, with probability $O(1 - 1/\sqrt{\ell})$, indistinguishably to the transcript of a successful key exchange between Alice and Bob. We planted $\sqrt{\ell} - 1$ solutions into random indices, and as long as we avoided the index with the solution (which happens with probability $1 - \frac{2}{\sqrt{\ell}}$), the rest of the pairs will be of the form $D_0 \times D_0$ with exactly one coordinate of overlapping instances with solutions. That coordinate will be the same as the index in the list problem with the solution.

So, since we are assuming Eve can run in $\mathsf{PFT}_{\ell(n)T(n)}$ time and we can create instances that look like key-exchange transcripts from list-problems, we can run Eve on each of these m different list-pair problems, and as long as she answers correctly for the c^* instance, we can solve our original problem in time $O((\ell(n)T(n))^{1-\delta})$ for $\delta > 0$. This is a contradiction to the hardness of the list problem, meaning Eve's time is bounded by $\Omega(\ell(n)T(n))$.

Analyzing the error in this case, when the key exchange succeeds, the total variation distance between an instance of the list problem being split and the original key-exchange transcript is bounded above by the following two sides:

- For the c^* that splits the D_1 instance of the list into one sampled from $D_1 \times D_1$, this succeeds with probability $1 - \epsilon_{split} \cdot \ell$.

- Given that we successfully split, the distance between the generated pairs of lists *after* we plant $\sqrt{\ell} - 1$ instances with a solution between this and the idealized list of $(D_b, D_{b'})$ instances with one (D_1, D_1), $\sqrt{\ell} - 1$ of the form (D_1, D_0) and $\sqrt{\ell} - 1$ of the form (D_0, D_1) is at most $\frac{2}{\sqrt{\ell}} + (1 - \frac{2}{\sqrt{\ell}})(\sqrt{\ell} \cdot \epsilon_{plant}) \leq \frac{2}{\sqrt{\ell}} + \sqrt{\ell}\epsilon_{plant}$.

- For the generated instances generated in a successful key exchange transcript, the error between this and the idealized list-pairs (described above) is at most $\ell \cdot \epsilon_{plant}$.

- Recall that $\epsilon_{plant}, \epsilon_{split} \leq \frac{1}{100\ell}$ and that $\ell \geq 2^{14}$. So, combined, the key-exchange transcript distribution and splitting the list-hard problem distribution are indistinguishable with probability at most

$$\begin{aligned}
&1 - (\epsilon_{split}\ell + \frac{2}{\sqrt{\ell}} + \sqrt{\ell}\epsilon_{plant} + \ell\epsilon_{plant}) \\
&= 1 - (\ell(\epsilon_{split} + \epsilon_{plant}) + \sqrt{\ell}\epsilon_{plant} + \frac{2}{\sqrt{\ell}}) \\
&\geq 1 - (\ell(\frac{2}{128\ell}) + \frac{1}{128\sqrt{\ell}} + \frac{2}{\sqrt{\ell}}) \\
&\geq 1 - (\frac{2}{128} + \frac{2}{128} + \frac{1}{128^2}) = 1 - \frac{1}{32} - \frac{1}{2^{14}} \\
&> 1 - \frac{1}{31}
\end{aligned}$$

Therefore, the total variation distance between key-exchange transcripts and the transformed ACLH instances is at most $\frac{1}{31}$.

Now, recall that if we have a $\mathsf{PFT}_{\ell(n)T(n)}$ algorithm E that resolves the single-bit key with advantage δ, then there exists a $\mathsf{PFT}_{\ell(n)T(n)}$ algorithm E^* that resolves the index of the key exchange transcript with probability $\delta/4$. Let $Transf$ be the algorithm that transforms an ACLH instance $\mathbf{I}$ to the key-exchange transcript (with TVD from a successful key-exchange transcript of $\frac{1}{34}$) Therefore, the probability that we fool Eve into solving our ACLH problem is

$$\Pr[E^*(Transf(\mathbf{I})) = i] \geq \delta/4 - \frac{1}{31} \geq \frac{1}{16} - \frac{1}{31} > \frac{1}{34}$$

Now, since the ACLH problem P allows for $\mathsf{PFT}_{\ell(n)T(n)}$ adversaries to have advantage at most $\frac{1}{34}$, this is a contradiction. Therefore, there does not exist a $\mathsf{PFT}_{\ell(n)T(n)}$ eavesdropping adversary that can resolve the single bit key with advantage $\frac{1}{4}$ (so resolving the key with probability $1/2 + 1/4 = 3/4$). □

We note that the range, $R \approx n^{6k}$ in the above corollary may be considered to be "too large" if you believe the hardness in the problem comes from a range where were are expected to get one solution with probability 1/2 ($R = O(n^k)$). So, in the next corollary, we address that problem, getting the key exchange using this much smaller range.

Now, we can put all of these together to get a weak fine-grained key exchange. We will then boost it to be a strong fine-grained key exchange (see the Definition 29 for weak versus strong in this setting).

Theorem 31. *If a problem P is $\ell(n)$-KER with plant time $G(n)$, solve time $S(n)$ and lower bound $T(n)$ when $\ell(n) \geq 2^{14}$, then construction 30 is a $((\ell(n)T(n), \alpha, \gamma)$-*$\mathsf{FG\text{-}KeyExchange}$*, with $\gamma \leq \frac{1}{10{,}000\ell(n)e}$ and $\alpha \leq \frac{1}{4}$.*

Proof. This is a simple combination of the correctness of the protocol, and the fact that an eavesdropper must take more time than the honest parties. We have that the $\Pr[b_A = b_B] \geq 1 - \frac{1}{10{,}000\ell e}$, implying $\gamma \leq \frac{1}{10{,}000\ell e}$ from Lemma 26. We have that Alice and Bob take time $O(\ell(n)G(n) + \sqrt{\ell(n)}S(n))$ and Eve must take time $\tilde{\Omega}(\ell(n)T(n))$ to get an advantage larger than $\frac{1}{4}$ by Lemmas 27 and 28. Because P is KER , $\ell(n)T(n) \in \tilde{\omega}\left(\ell(n)G(n) + \sqrt{\ell(n)}S(n)\right)$, implying there exists $\delta > 0$ so that $\ell(n)G(n) + \sqrt{\ell(n)}S(n) \in \tilde{O}(\ell(n)T(n)^{1-\delta})$. So, we have correctness, efficiency and security. □

Next, we are going to amplify the security of this key exchange using parallel repetition, drawing off of strategies from [DNR04] and [BIN97].

Theorem 32. *If a weak $(\ell(n)T(n), \alpha, \gamma)$-*$\mathsf{FG\text{-}KeyExchange}$* exists where $\gamma = O\left(\frac{1}{n^c}\right)$ for some constant $c > 0$, but $\alpha = O(1)$, then a* Strong *$(\ell(n)T(n))$-*$\mathsf{FG\text{-}KeyExchange}$* also exists.*

Proof. Using techniques from [BIN97], we will phrase breaking this key exchange as an eavesdropper as an honest-verifier two-round parallel repetition game. The original game as a $\mathsf{PFT}_{\ell(n)T(n)}$ prover P and verifier V. V generates an honest transcript of the key exchange between Alice and Bob and sends this transcript to P. Note that the single-bit key sent in this protocol is uniformly distributed. P wins if P can output the key correctly. Now, because any eavesdropper running $\mathsf{PFT}_{\ell(n)T(n)}$ only has advantage α of determining the key, the prover P has probability at most $\frac{1}{2}+\alpha$ of winning this game. Let $\beta = \frac{1}{2}+\alpha$. By Theorem 4.1 of [BIN97], if we instead have V generate m parallel repetitions (m independent transcripts of the key exchange), then the probability that P can find *all* of the keys in $\mathsf{PFT}_{\ell(n)T(n)}$ is less than $\frac{16}{1-\beta}\cdot e^{-m(1-\beta^2)/256}$ (which is larger than $\frac{32}{(1-\beta)}\cdot e^{-mc(1-\beta^2)/256}$).

Let $m = \frac{512}{1-\beta^2}\cdot\log(n)$, which is sub-polynomial in n because β is at least constant. and we get that the probability the prover succeeds in finding all m keys is at most $\beta' = O\left(\frac{1}{n^2}\right)$. Now, we need this to work while there is some error γ. Note that γ is at most $1/n^c$, so the probability we get an error in any of the m instances of the key exchange is $(1-\gamma)^{d\cdot\log(n)}$ for some constant d. Asymptotically, this is $e^{-\gamma d\log(n)} = n^{-\gamma d}$. This is where we required γ to be so small: because $\gamma = O(1/n^c)$, this probability of failure is $o(\sqrt{\gamma})$, which is still insignificant.

Now, we need to turn this m parallel-repetition back into a key exchange. We will first do that by employing our fine-grained Goldreich-Levin method: the weak key-exchange will be run m times in parallel and Alice will additionally send a uniformly random m-length binary vector, $\mathbf{r}$. The key will be the m keys, $\mathbf{k} = (k_1,\ldots,k_m)$ dot-producted with $\mathbf{r}$: $\mathbf{k}\cdot\mathbf{r}$. Because m is sub-polynomial in n and the Goldreich-Levin security reduction only requires $\tilde{O}(m^2)$ time, $\mathbf{k}\cdot\mathbf{r}$ is a fine-grained hard-core bit for the transcript. Therefore, an eavesdropper will have advantage at most $\frac{1}{2}+\mathsf{insig}(n)$ in determining the shared key. □

Remark 4. *It is not obvious how to amplify correctness* and *security of a fine-grained key exchange at the same time. If we have a weak $(\ell(n)T(n),\alpha,\gamma)$-FG-KeyExchange, where $\alpha = \mathsf{insig}(n)$ but $\gamma = O(1)$, then we can use a standard repetition error-correcting code to amplify γ. That is, we can run the key exchange $\log^2(n)$ times to get $\log^2(n)$ keys (most of which will agree between Alice and Bob), and to send a message with these keys, send that message $\log^2(n)$ times. With all but negligible probability, the decrypted message will agree with the sent message a majority of the time. Since with very high probability the adversary cannot recover any of the keys in $\mathsf{PFT}_{\ell(n)T(n)}$ time, this repetition scheme is still secure.*

As shown in Theorem 32, we can also amplify a key exchange that has constant correctness and polynomial soundness to one with $1-\mathsf{insig}(n)$ correctness and polynomial soundness. However, it is unclear how to amplify both at the same time in a fine-grained manner.

Corollary 10. *If a problem P is $\ell(n)$-KER, then a Strong $(\ell(n)T(n))$-FG-KeyExchange exists.*

Proof. The probability of error in Construction 30 is at most $\frac{1}{10{,}000\ell(n)e}$, and $\ell(n) = n^{\Omega(1)}$ (due to the fact that $\ell(n)$ comes from the definition of list-hard, Definition 35.

The probability a $\mathsf{PFT}_{\ell(n)T(n)}$ eavesdropper has of resolving the key is $\frac{1}{2}+\alpha$ where $\alpha \leq \frac{1}{4}$. This means our $\beta \leq \frac{3}{4} = O(1)$. Since the clauses in theorem 32 are met, a Strong $(\ell(n)T(n))$-FG-$\mathsf{KeyExchange}$ exists. $\square$

Finally, using the fact that Alice and Bob do not use each other's messages to produce their own in Construction 30, we prove that we can remove all interaction through repetition and get a $T(n)$-fine-grained public key cryptosystem.

Lemma 29. *Construction 30 does not need interaction.*

Proof. There is a constant probability that the construction fails at each round. So, Alice and Bob will simply run the protocol $c \cdot \log(n)$ times in parallel and take the key generated from the first successful exchange. There are two errors to keep track of: the chance that Alice and Bob's S_A and S_B do not intersect in exactly one spot and the probability that an instance was generated with a false-positive. Since ϵ_{plant} is so small $(O(1/n^{\Omega(1)}))$, we do not need to worry about the false-positives (the chance of generating one is insignificant). So, the error we are concerned with is the chance that none of the $c\log(n)$ instances of the key exchange end up having S_A and S_B overlapping in exactly 1 entry. This happens with probability at most $\Pr[\text{no overlap } c\log(n) \text{ times}] = (\Pr[\text{no overlap once}])^{c\log n} \leq O(1/n^c)$, which is also insignificant. Therefore, the chance this key exchange fails is at most $1-(\frac{c\log(n)}{10{,}000\ell e}+\frac{1}{n^c}) = 1-\mathsf{insig}(n)$. $\square$

Theorem 33. *If a problem P is $\ell(n)$-KER, then a $\ell(n)\cdot T(n)$-fine-grained public key cryptosystem exists.*

Proof. First, consider an amplified, non-interactive version of Construction 30 (combination of Corollary 10 and lemma 29): Alice and Bob run the protocol m times in parallel, where $m = \frac{512}{1-\beta^2}\cdot\log(n)$, and Alice sends an additional random binary vector $\mathbf{r} \in \{0,1\}^m$. The key they agree on is the dot-product between $\mathbf{r}$ and the vector of keys exchanged. This is a Strong $(\ell(n)T(n))$-FG- $\mathsf{KeyExchange}$ (see Corollary 10). We now define the three algorithms for a fine-grained public key cryptosystem.

- $\mathsf{KeyGen}(1^n)$: run Bob's half of the non-interactive protocol m times, generating m collections of $c\log(n)$ lists of $\ell(n)$ instances of P: $\{(\mathbf{I}_B^{(1,i)},\ldots,\mathbf{I}_B^{(c\log(n),i)})\}_{i\in[m]}$. Each list $\mathbf{I}_B^{(j,i)}$ has a random set $S_B^{(j,i)} \subset [\ell]$ where instances $I_k \in \mathbf{I}_B^{(j,i)}$ are from $\mathsf{Generate}(n,1)$ if $k \in S_B^{(j,i)}$ and from $\mathsf{Generate}(n,0)$ otherwise. The public key is $pk = \{(\mathbf{I}_B^{(1,i)},\ldots,\mathbf{I}_B^{(c\log(n),i)})\}_{i\in[m]}$ and the secret key is $sk = \{(S_B^{(1,i)},\ldots,S_B^{(c\log(n),i)})\}_{i\in m}$.
- $\mathsf{Encrypt}(pk, m \in \{0,1\})$: run Alice's half of the protocol m times, solving for the shared key, and then encrypting a message under that key. More formally, generate m lists of $c\log(n)$ sets $S_A^{(j,i)} \subset [\ell]$ such that $|S_A^{(j,i)}| = \sqrt{\ell}$. Then, generate lists of instances $\mathbf{I}_A^{(j,i)}$ for $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,c\log(n)\}$, where for every instance $I_k^{(j,i)} \in \mathbf{I}_A^{(j,i)}$, $I_k^{(j,i)} = \mathsf{Generate}(n,1)$ if $k \in S_A^{(j,i)}$ and otherwise use $\mathsf{Generate}(n,0)$. Now, for each of these m instances, compute the shared key as in Construction the non-interactive version 30 (see lemma 29), to

get a vector of keys $\mathbf{k} = (k_1, \ldots, k_m)$. If any of these exchanges fail, output $\perp$. Now, compute a random binary vector $\mathbf{r} \in \{0,1\}^m$ and let $key = \mathbf{k} \cdot \mathbf{r}$. Finally, let the ciphertext $c = \left((\mathbf{I}_A^{(1)}, \ldots, \mathbf{I}_A^{(m)}), \mathbf{r}, key \oplus m\right)$.

- $\mathsf{Decrypt}(sk, c)$: Bob computes the shared key and decrypts the message. Formally, let $c = \left((\mathbf{I}_A^{(1)}, \ldots, \mathbf{I}_A^{(m)}), \mathbf{r}, c^*\right)$. Bob computes the shared key just like Alice, using $(\mathbf{k} \cdot \mathbf{r}) \oplus c^* = m'$. Output the bit m'.

Now we will prove this is indeed a fine-grained public key cryptosystem. First, because the key exchange took Alice and Bob $\mathsf{PFT}_{\ell(n)T(n)}$ time, this scheme is efficient, requiring at most $(\ell(n)T(n))^{1-\delta} \cdot m \cdot (c \log n) = \tilde{O}(\ell(n)T(n)^{(1-\delta)})$ for constant $\delta > 0$.

Next, the scheme is correct. This comes directly from the fact that the key exchange succeeds with probability $1 - \mathsf{insig}(n)$.

Lastly, the scheme is secure. This is a simple reduction from the security game to an eavesdropper. For sake of contradiction, let Eve be a $\mathsf{PFT}_{\ell(n)T(n)}$ adversary that can win the CPA-security game as in Definition 30 with probability $\frac{1}{2} + \epsilon$ where $\epsilon = \mathsf{sig}(n)$. Eve can then directly win the key exchange wi†h the same advantage because the message the challenger gives to Eve is simply a transcript of a key exchange. $\square$

Note that this encryption scheme can be used to send any sub-polynomial number of bits, just by running it in sequence sub-polynomially many times. We also want to note that the adversary's advantage cannot be any less than $\frac{1}{\text{poly}(n)}$ since, due to the fine-grained nature of the scheme, the adversary can always solve the hard problem via guessing.

Corollary 11. *Given the strong Zero-k-Clique-R Hypothesis over range $R = \ell(n)^2 n^{2k}$, there exists a $(\ell(n)T(n), 1/4, \mathsf{insig}(n))$-FG-KeyExchange, where Alice and Bob can exchange a sub-polynomial-sized key in time $\tilde{O}\left(n^k\sqrt{\ell(n)} + n^2\ell(n)\right)$ for every polynomial $\ell(n) = n^{\Omega(1)}$.*

There also exists a $\ell(n)T(n)$-fine-grained public-key cryptosystem, where we can encrypt a sub-polynomial sized message in time $\tilde{O}\left(n^k\sqrt{\ell(n)} + n^2\ell(n)\right)$.

Both of these protocols are optimized when $\ell(n) = n^{2k-4}$.

Proof. This comes from the fact that strong Zero-k-Clique-R Hypothesis implies that Zero-k-Clique is a KER problem by Theorem 27, Theorem 26, and Theorem 25. So we can use construction 30 to get the key-exchange by Theorem 31 and Claim 14.

The optimization comes from minimizing $\tilde{O}\left(n^k\sqrt{\ell(n)} + n^2\ell(n)\right)$, which is simply to set $n^k\sqrt{\ell(n)} = n^2\ell(n)$. This results in $\ell(n) = n^{2k-4}$.

The gap between honest parties and dishonest parties is computed as follows. Honest parties take $H(n) = \tilde{O}(\ell(n)n^2) = \tilde{O}(n^{2k-2})$. Dishonest parties take $E(n) = \tilde{O}(\ell(n)n^k) = \tilde{O}(n^{3k-4})$. We have that $E(n) = H(n)^t$ where $t = \frac{3k-4}{2k-2}$, which approaches 1.5 as $k \to \infty$. So, we have close to a 1.5 gap between honest parties and dishonest ones as long as we assume $T(n) = n^k$. $\square$ $\square$

The Zero-3-Clique hypothesis (the Zero Triangle hypothesis) is generally better believed than the Zero-k-Clique hypothesis for larger k. Note that even with the strong Zero-3-Clique hypothesis we get a key exchange with a gap in the running times of Alice and Bob vs Eve. In this case, the gap is $t = 5/4 = 1.2$.

Also, since the Zero-k-Clique-hypothesis may be more believable over a smaller range, say $R = n^k$, we can combine Corollary 11 and Theorem 23 to get the below result: even with a relatively small range, we can still build a fine-grained public-key cryptosystem.

Corollary 12. *Given the strong Zero-k-Clique-R Hypothesis over range $R = n^k$, where $\ell(n)$ is polynomial, there exists a $(\ell(n)T(n), 1/4, \mathsf{insig}(n))$-FG-KeyExchange, where Alice and Bob can exchange a sub-polynomial-sized key in time $\tilde{O}\left(n^k\sqrt{\ell(n)} + n^2\ell(n)\right)$ for every polynomial $\ell(n) = n^{\Omega(1)}$.*

There also exists a $\ell(n)T(n)$-fine-grained public-key cryptosystem, where we can encrypt a sub-polynomial sized message in time $\tilde{O}\left(n^k\sqrt{\ell(n)} + n^2\ell(n)\right)$.

Both of these protocols are optimized when $\ell(n) = n^{2k-4}$.

Proof. Using Corollary 11 and Theorem 23 we can use the hardness of strong Zero-k-Clique-R Hypothesis over range $R = n^k$ to show hardness for strong Zero-k-Clique-R Hypothesis over range $R = \ell(n)^2 n^{2k}$. □