On the Compressibility of $\mathcal{NP}$ Instances and Cryptographic Applications[‡]

Danny Harnik[*]
Dept. of Computer Science
Technion
Haifa, Israel
harnik@cs.technion.ac.il.

Moni Naor[†]
Dept. of Computer Science and App. Math.
Weizmann Institute of Science
Rehovot, Israel
moni.naor@weizmann.ac.il.

Abstract

We initiate the study of compression that preserves the solution *to an instance of a problem rather than preserving the instance itself. Our focus is on the compressibility of $\mathcal{NP}$ decision problems. We consider $\mathcal{NP}$ problems that have long instances but relatively short witnesses. The question is, can one efficiently compress an instance and store a shorter representation that maintains the information of whether the original input is in the language or not. We want the length of the compressed instance to be polynomial in the length of the* witness *rather than the length of original input. Such compression enables to succinctly store instances until a future setting will allow solving them, either via a technological or algorithmic breakthrough or simply until enough time has elapsed.*

We give a new classification of $\mathcal{NP}$ with respect to compression. This classification forms a stratification of $\mathcal{NP}$ that we call the $\mathcal{VC}$ hierarchy. The hierarchy is based on a new type of reduction called W-reduction and there are compression-complete problems for each class.

Our motivation for studying this issue stems from the vast cryptographic implications compressibility has. For example, we say that SAT is compressible if there exists a polynomial $p(\cdot,\cdot)$ so that given a formula consisting of m clauses over n variables it is possible to come up with an equivalent (w.r.t satisfiability) formula of size at most $p(n, \log m)$. Then given a compression algorithm for SAT we provide a construction of collision resistant hash functions from any *one-way function. This task was shown to be impossible via black-box reductions [41], and indeed the construction presented is inherently non-black-box. Another application of SAT compressibility is a cryptanalytic result concerning the limitation of everlasting security in the bounded storage model when mixed with (time) complexity based cryptography. In addition, we study an approach to constructing an Oblivious Transfer Protocol from* any *one-way function. This approach is based on compression for SAT that also has a property that we call* witness retrievability. *However, we mange to prove severe limitations on the ability to achieve witness retrievable compression of SAT.*

1 Introduction

In order to deal with difficult computational problems several well established options were developed, including: approximation algorithms, subexponential algorithms, parametric complexity and average-case complexity. In this paper we explore our favorite approach for dealing with problems: *postpone* them (hopefully, without cluttering our desk or disk). We initiate the study of the compressibility of NP problems for their resolution in some future setting. Rather than solving a given instance, we ask whether a shorter instance with the same solution can be found efficiently. We emphasize that we are not interested in maintaining the information about the original instance (as is the case in typical notions of compression), but rather maintain the solution only. The solution can possibly be much shorter than the input (as short as a yes/no answer), thus the potential of such a compression is high.

Specifically, we consider $\mathcal{NP}$ problems that have long instances but relatively short witnesses. An $\mathcal{NP}$ language L is defined by an efficiently computable relation R_L such that an input (or instance) x is in L if and only if there exists a witness w such that $R_L(x, w) = 1$. Throughout the paper, an $\mathcal{NP}$ instance is characterized by two parameters m and n: The length of the instance x is denoted by m and the length of the witness w is denoted by n. The problems of interest are those having short witnesses, i.e. $n << m$. Traditionally, the study of $\mathcal{NP}$ languages evolves around the ability or inability to efficiently decide if an instance is in the language or not, or to find a witness w for an instance x within polynomial time. We introduce the question of compressibility of such instances.

[*]This research was conducted while at the Weizmann Institute.
[†]Incumbent of the Judith Kleeman Professorial Chair.
[‡]Research supported by a grant from the Israel Science Foundation.

Compressing SAT Instances: To illustrate the relevant setting, we use the well known example of SAT. An instance Φ for SAT consists of a CNF formula over n variables and we define that $\Phi \in SAT$ if there exists an assignment to the n variables that satisfies all the clauses of Φ. We begin with compressibility with respect to decision, and discuss compressibility of the search variant later in the paper. The question of compressibility of SAT is the following:

Example 1.1 (Compression of SAT instances)
Does there exist an efficient algorithm and a polynomial $p(\cdot,\cdot)$ with the following input and output:
Input: *A CNF formula Φ with m clauses over n variables (we are interested in $m >> n$).*
Output: *A formula Ψ of size $p(n, \log m)$ such that Ψ is satisfiable if and only if Φ is satisfiable.*

The idea is that the length of Ψ should be essentially unrelated to the original length m, but rather to the number of variables (or in other words, to the size of the witness). Typically, we think of the parameters m and n as related by some function, and it is instructive (but not essential) to think of m as larger than any polynomial in n. So potentially, the length of Ψ can be significantly shorter than that of Φ.[1]

In general, one cannot expect to compress all the formulas, or else we would have an efficient algorithm for all $\mathcal{NP}$ problems.[2] However, once we introduce the setting of a shorter witness, then compression becomes plausible. Note that if $\mathcal{P} = \mathcal{NP}$ and we actually know the algorithm for SAT then clearly compression is trivial, simply by solving the satisfiability of Φ and outputting 1 if $\Phi \in SAT$ and 0 otherwise.

Motivation for Compression: Compressing for the future is an appealing notion for various settings. There are numerous plausible scenarios that will give us more power to solve problems in the future. We could potentially find out that $\mathcal{P} = \mathcal{NP}$ and solve all our $\mathcal{NP}$ problems then. We may have faster computers or better means of computing such as quantum computers or any other physical method for solving problems (see Aaronson [1] for a list of suggestions). Above all, the future entails lots and lots of time, a resource that the present is usually short of. Saving the problems of today as they are presented is wasteful, and compression of problems will allow us to store a far greater number of problems for better days.

Our interest in studying the issue of compression stems from the vast cryptographic implications of compressibility. We demonstrate three questions in cryptography that compression algorithms would resolve (see Section 3). We are confident that the compression of problems implies further applications both within and outside of cryptography. For example, in subsequent works, Dubrov and Ishai [13] show the relevance of the notion of compression to derandomization and Dziembowski [15] shows that compression is related to the study of forward-secure storage (see Section 5 on related work). The concept of compression of problems is also interesting beyond the confines of $\mathcal{NP}$ problems, and makes sense in any setting where the compression requires much less resources than the actual solution of the problem.

1.1 Compression of NP instances

We define the notion of compression with respect to an $\mathcal{NP}$ language. For simplicity, we assume that an input to an $\mathcal{NP}$ language L includes an encoding of the parameter n that upper bounds the length of a potential witness.[3] We also associate with L a specific $\mathcal{NP}$ relation R_L that defines it (as mentioned above). We note that once the parameters m and n are explicit, it is in most cases immaterial what specific relation defines the language and the properties we discuss (such as compressibility) are properties of the language at hand (unless stated otherwise). In essence, a compression algorithm is a specialized Karp-reduction that also reduces the length of the instance.

Definition 1.2 (Compression Algorithm for $\mathcal{NP}$ Instances)
Let L be an $\mathcal{NP}$ language where m and n denote the instance length and the witness length respectively. A compression algorithm *for L is a probabilistic polynomial time machine Z along with a language L' in $\mathcal{NP}$ (or more accurately in $\mathcal{NP}(poly(m))$)[4] and a polynomial $p(\cdot,\cdot)$ such that for all large enough m:*

1. *For all $x \in \{0,1\}^m$ with parameter n the length of $Z(x)$ is at most $p(n, \log m)$.*
2. *$Z(x) \in L'$ if and only if $x \in L$*

The above definition is of an errorless *compression. We also consider a probabilistic variant called* ε-compression *for some real function $\varepsilon : \mathbb{N} \to [0,1]$. The probabilistic definition is identical to the errorless one except for the second property that is augmented to:*

2'. For large enough n, for all $x \in \{0,1\}^m$ with parameter n it holds that:

$$\Pr[(Z(x) \in L') \Leftrightarrow (x \in L)] > 1 - \varepsilon(n)$$

where probability is over the internal randomness of Z. Typically we require $\varepsilon(\cdot)$ to be negligible.

[1] Note, that since our requirement for compression is only relevant for problems where $m >> n$, then an $\mathcal{NP}$-complete problem such as 3-SAT (where all clauses have exactly 3 literals) is irrelevant for compression as in such formulas m is already at most $O(n^3)$.

[2] Suppose that every formula can be compressed by a single bit, then sequentially reapplying compression to the input will result in a very short formula that may be solved by brute enumeration.

[3] Typically, the parameter n is indeed part of the description of the problem (e.g. for Clique, SAT, Long-path and others).

[4] By $\mathcal{NP}(poly(m))$ we mean in nondeterministic-time $poly(m)$ (that is, verifiable in time $poly(m)$ when given a non-deterministic hint).

The paper consists of two parts: *Part I* is a study of the concept of compression of $\mathcal{NP}$ instances from a complexity point of view. *Part II* introduces the cryptographic applications of compression algorithms.

How much to compress: Definition 1.2 (of compression algorithms) requires a very strong compression, asking that the length of the compression is polynomial in n and $\log m$. For the purposes of part I of the paper (the complexity study), it is essential that the compression is at least sub-polynomial in m in order to ensure that the reductions defined with respect to compressibility (See Section 2) do compose.[5] Furthermore, for part II (the applications) this definition may be strongly relaxed, where even a compression to $m^{1-\varepsilon}$ for some constant ε suffices for some applications.

The Complexity of L': Another requirement of Definition 1.2 is that the language L' be in $\mathcal{NP}(poly(m))$. In general, this requirement may also be relaxed and the result still be meaningful for some applications. In particular, we do not need to put a bound on the complexity of L', but only require that there is enough information in $Z(x)$ to determine whether $x \in L$ or not. One case where we use a definition with unbounded extraction is the compression of search problems.It should be noted however that in some settings the requirement for L' to be in $\mathcal{NP}(poly(m))$ is essential, such as when defining the witness retrievability property (Definition 3.2). Moreover, in some cases it is natural to further restrict L' to actually be in $\mathcal{NP}$ (that is in $\mathcal{NP}(poly(n, \log m))$. For instance, this is the case in the definition of compression of SAT (Example 1.1). Finally, note that if the compression is *errorless*, then L' must be in $\mathcal{NP}(poly(m))$ simply by the definition of compression.[6]

Paper organization: Due to space limitations most of the formal treatment is deferred to the full version of this paper [21]. Instead we survey all the results of this paper including the complexity study in Section 2 and the cryptographic applications in Section 3. We give one full cryptographic application (to collision resistant hash) in Section 4. In Section 5 we discuss related and subsequent works. and conclude with a discussion and some open problems.

2 Classifying $\mathcal{NP}$ Problems with Respect to Compression.

Examples of compression: We are interested in figuring out which $\mathcal{NP}$ languages are compressible and, in particular, whether important languages such as SAT and Clique are compressible. For starters, we demonstrate some non-trivial languages that do admit compression: we show compression for the well known $\mathcal{NP}$-complete problem of vertex-cover (see Section 2.1) and for another $\mathcal{NP}$-complete language known as minimum-fill-in. We show a generic compression of sparse languages (languages containing relatively few words from all possible instances). As a specific example we mention the language consisting of strings that are the output of a cryptographic pseudorandom generator and also consider the sparse subset sum problem. In addition we show compression for the promise problem GapSAT.[7] However, these examples are limited and do not shed light on the general compression of other $\mathcal{NP}$ problems. Moreover, it becomes clear that the traditional notions of reductions and completeness in $\mathcal{NP}$ do not apply for the case of compression (i.e., the compression of an $\mathcal{NP}$-complete language does not immediately imply compression for all of $\mathcal{NP}$). This is not surprising since this is also the case with other approaches for dealing with $\mathcal{NP}$-hardness such as approximation algorithms or subexponential algorithms (see for example [38]) and parameterized complexity (see [12] and further discussion in Section 5 on related work). For each of these approaches, appropriate new reductions where developed, none of which is directly relevant to our notion of compression.

W-reductions and completeness: We introduce W-reductions in order to study the possibility of compressing various problems in $\mathcal{NP}$. These are reductions that address the length of the witness in addition to membership in an $\mathcal{NP}$ language. Formally:

Definition 2.1 (W-Reduction) *For two $\mathcal{NP}$ languages L and L' we say that L* W-reduces *to L' if there exist polynomials p_1 and p_2 and a polynomial time computable function f that takes an instance x for L and outputs an instance $f(x)$ for L' such that: (i) $f(x) \in L'$ if an only if $x \in L$, and (ii) If x is of length m with witness length n, then $f(x)$ is of length $p_1(n, m)$ with witness length only $p_2(n, \log m)$.*

W-reductions have the desired property that if L W-reduces to L', then any compression algorithm for L' yields a compression algorithm for L. Following the definition of W-reductions we define also the matching notion of compression-complete and compression-hard languages for a class.

The $\mathcal{VC}$ classification: We introduce a classification of $\mathcal{NP}$ problems with respect to compression. The classification presents a structured hierarchy of $\mathcal{NP}$ problems, that is surprisingly different from the traditional view and closer in nature to the W hierarchy of parameterized complexity

[5] For clarity we choose a polynomial in $\log m$, although this may be replaced by any sub-polynomial function $m'(.)$ (a function such that for large enough m for any polynomial $q(\cdot)$ we have $m'(m) < q(m)$).

[6] Suppose that there exists a compression algorithm Z for L then define L' to be the language of all $Z(x)$ such that $x \in L$. Then, for every $y \in L'$ a verification algorithm takes as a nondeterministic witness a value x, a witness to $x \in L$ along with randomness for the compression algorithm and verifies that indeed $y = Z(x)$. Thus if Z never introduces an error then L' is in $\mathcal{NP}(poly(m))$.

[7] I.e. a promise problem were either the formula is satisfiable or every assignment does not satisfy a relatively large number of clauses.

(see [12]). We call our hierarchy $\mathcal{VC}$, short for "verification classes", since the classification is closely related to the verification algorithm of $\mathcal{NP}$ languages when allowed a preprocessing stage. We give here a very loose description of the classes, just in order to convey the flavor of the classification. Formal definitions appear in the full version [21]. In the following definition, when we use the term "verification" we actually mean "verification with preprocessing":

- For $k \geq 2$, the class $\mathcal{VC}_k$ is the class of languages that have verification which can be presented as a depth k polynomial size circuit (polynomial in n and m). For example, the language SAT is compression-complete for the class $\mathcal{VC}_2$. Other examples include Integer-Programming that resides in $\mathcal{VC}_{\log n}$ and Dominating-Set that is in $\mathcal{VC}_3$. Both of which are shown to be compression-hard for $\mathcal{VC}_2$.
- $\mathcal{VC}_1$ is the class of languages that have *local* verification. That is, languages which can be verified by testing only a small part (of size $poly(n, \log m)$) of the instance. This class contains many natural examples such as the Clique language or Long-path.
- $\mathcal{VC}_{OR}$ is the class of languages that have verification which can be presented as the OR of m small instances of SAT (each of size n). This class contains the languages that are relevant for the cryptographic applications. The Clique language is compression-hard for this class.
- $\mathcal{VC}_0$ is the class of compressible languages. In particular it includes vertex cover, sparse languages and GapSAT.

We show that the classes described form a hierarchy, that is:

$$\mathcal{VC}_0 \subseteq \mathcal{VC}_{OR} \subseteq \mathcal{VC}_1 \subseteq \mathcal{VC}_2 \subseteq \mathcal{VC}_3 \ldots$$

We discuss some of the more interesting classes in the $\mathcal{VC}$ hierarchy, classify some central $\mathcal{NP}$ problems and mention compression-complete problems for the classes. Note that the existence of a compression algorithm for a complete problem for some class entails the collapse of the hierarchy up to that class into $\mathcal{VC}_0$.

In addition, we study the compression of $\mathcal{NP}$ *search* problems. That is, compressing an instance in a way that maintains all the information about a witness for the problem. We show that the compression of a class of decision problems also implies compression for the corresponding search problems. Formally:

Theorem 2.2 *If a class $\mathcal{VC}_k$ has a compression algorithm, then there is a compression algorithm for the search problem of a relation R_L of $L \in \mathcal{VC}_k$.*

This theorem turns out to be useful for the cryptanalysis result regarding the bounded storage model (see Section 3).

2.1 Compression for Vertex Cover

Attempting to compress $\mathcal{NP}$ instances may require a different approach than actually solving $\mathcal{NP}$ problems. The compressed version may actually be harder to solve (computational time-wise) than the original one (and may require a somewhat longer witness altogether). However, in some cases, a solution for compression might arise while trying to solve the problem. While a full solution of an $\mathcal{NP}$ problem may take exponential time, it is plausible that the first polynomial number of steps leaves us without an explicit solution but with a smaller instance. One such example is the well studied $\mathcal{NP}$-complete problem of Vertex-Cover. The problem receives as input a graph $G = (V, E)$ and asks whether there exists a subset of vertices $S \subseteq V$ of size at most k such that for every edge $(u, v) \in E$ either u or v are in S. The parameters are the instance length m, which is at most $O(|E| \log |V|)$, and the witness length $n = k \log |V|$

Claim 2.3 *There exists a witness retrievable compression algorithm for Vertex-Cover.*

Proof: We are following the parameterized complexity algorithm for vertex-cover (presented in [12] and attributed to Buss). If a vertex-cover S of size k exists, then any vertex of degree greater than k must be inside the set S. The compression algorithm simply identifies all such vertices and lists them in the cover, while removing all their outgoing edges (that do not need to be covered by other vertices). The graph left after this process has maximal degree k, and furthermore all edges have at least one end in the cover. Thus, if the original graph has a k vertex cover then the total number of edges left is at most k^2 (at most k vertices in the cover with at most k edges each). If there are more then k^2 edges then the answer to the problem is NO, otherwise, such a graph can be represented by the list of all edges, which takes $k^2 \log k$ bits. The compression can be made witness retrievable since if we use the original labels of vertices to store the new graph, then the original cover is also a cover for the new compressed graph. □

It is interesting to note that we do not know of a compression algorithm for the Clique problem or the Dominating Set problem, which are strongly linked to the vertex-cover problem in the traditional study of $\mathcal{NP}$, and in fact, the results in Section 3 show strong cryptographic implications of a compression algorithm for these languages.

3 Implications to Cryptography

As the main motivation for the study of compression, we provide some strong implications of compressibility to cryptography. The implications described are of contrasting flavors. On the one hand we show constructions of cryptographic primitives using compression algorithms, while on

the other hand we show a cryptanalysis using compression algorithms (or alternatively, this can be considered as an application of incompressibility of languages). For simplicity we provide the implication with respect to the compression of SAT. We note however, that the same statements can actually be made with compression of languages that are compression hard for the class $\mathcal{VC}_{OR}$ (such as the Clique language). This class is the lowest class in our $\mathcal{VC}$ hierarchy, and potentially easier to compress than SAT. Moreover, the instances that we need to compress for our applications are further limited in the sense that (i) the instances are in $\mathcal{NP} \cap$ Co-$\mathcal{NP}$ and (ii) the (positive and negative) instances have a unique witness.

Basing Collision Resistant Hash Functions on Any One-Way Function: Collision Resistant Hash functions (CRH) are important cryptographic primitives with a wide range of applications, e.g. [36, 7, 28, 8, 32, 3]. Loosely speaking, a CRH is a family $\mathcal{H}$ of length reducing functions, such that no efficient algorithm can find collisions induced by a random hash from the family. Currently there is no known construction of CRH from general one-way functions or one-way permutations, and moreover, Simon [41] showed that basing CRH on one-way permutations cannot be achieved using black-box reductions. We show how a general compression algorithm may be used to bridge this gap.

Theorem 3.1 *If there exists an errorless*[8] *compression algorithm for SAT then there exists a construction of collision resistant hash functions based on any one-way function.*

The construction of the CRH in Theorem 3.1 is inherently non-black-box and uses the program of the one-way function via Cook's Theorem [6]. This is essential to the validity of this approach, in light of the black-box impossibility result [41].

An interesting corollary of this result is a construction of statistically hiding bit commitment from any one-way function, which is currently an open problem. Moreover, the construction would require only a single round of interaction ([35, 19] show constructions of statistically hiding bit commitment based on one-way functions with a specific structure and also require a large number of rounds of interaction).

On Everlasting Security and the Hybrid Bounded Storage Model: The *bounded storage model* (BSM) of Maurer [31] provides the setting for the appealing notion of *everlasting security* [2, 10]. Loosely speaking, two parties, Alice and Bob, that share a secret key in advance, may use the BSM to encrypt messages in a way that the messages remain secure against a computationally unbounded adversary, even if the shared secret key is eventually revealed.

However, if the parties do not meet in advance to agree on a secret key then everlasting security requires high storage requirements from Alice and Bob [16], rendering encryption in this model less appealing. Hoping to overcome this, it was suggested to combine the BSM with computational assumptions (what is called here the hybrid BSM). In particular, to run a computational key agreement protocol in order to agree on a shared secret key, and then run one of the existing BSM schemes. Dziembowski and Maurer [16] showed that this idea does not necessarily work in all cases, by showing an attack on a protocol consisting of the combination of a specific (artificial) computational key agreement protocol with a specific BSM encryption scheme.

We show that compression of $\mathcal{NP}$ instances can be used to attack *all* hybrid BSM schemes. Or in other words, if a compression of SAT exists, then the hybrid BSM is no more powerful than the standard BSM. One interpretation of this result is that in order to prove everlasting security for a hybrid BSM scheme, without further conditions, one must prove that there exists no compression algorithm for SAT. Alternatively, as a relaxation, one should come up with a reasonable incompressibility assumption regarding the resulting formulae. Note however that a straightforward assumption of the form "this distribution on SAT formulae is incompressible" is not efficiently falsifiable, in the sense of Naor [34], that is, it is not clear how to set up a challenge that can be solved in case the assumption is false.

ON RANDOM ORACLES: The authors of this paper show in [20] that if all parties are given access to a random oracle, then there actually exists everlasting security in the hybrid BSM without an initial key and with low storage requirements from Alice and Bob[9]. Therefore, finding a compression algorithm for SAT would show an example of a task that is simple with random oracles but altogether impossible without them. This would constitute an argument against relying (blindly) on random oracles to determine whether a task is feasible at all. This is different than previous results such as [5, 18, 4] that show a specific protocol that becomes insecure if the random oracle is replaced by a function with a small representation. Model separation results (such as the implication of compression) were discussed by Nielsen [37](for non-interactive non-committing encryption) and Wee [44] (for obfuscating point functions), but the separation there is between the programmable and non-programmable random oracle models (in contrast, the hybrid BSM result in [20] holds also if the oracle is non-programmable).

[8] The construction of CRH requires that the error probability of compression algorithm will be zero. This can be slightly relaxed to an error that is exponentially small in m (rather than n).

[9] This does not contradict the compressibility of SAT, since the cryptanalytic result is not black-box and assumes access to the full description of the programs of Alice and Bob. Thus this result is not preserved in the presence of a random oracle.

The actual model and results: The *bounded storage model* bounds the storage space of an adversary rather than its running time. It utilizes the public transmission of a long random string $\mathcal{R}$ of length m (sometimes referred to as the broadcast string), and relies on the assumption that an eavesdropper cannot possibly store all of this string. The everlasting security achieved by encryption schemes in this model means that an encrypted message remains secure even if the adversary eventually gains more storage or gains knowledge of (original) secret keys that may have been used. However, if the honest parties do not share any private information in advance, then achieving everlasting security requires storage capacity of $\Omega(\sqrt{m})$ from the honest parties (as shown in [16]).

The *hybrid bounded storage model* (see [20] for formal definitions and notions of security) assumes computational limitations on the eavesdropper up until the time that the transmission of $\mathcal{R}$ has ended. Computational assumptions with such a strict time limit are generally very reasonable. For instance, in the key agreement example, all that we require is that the computational protocol is not broken in the short time period between its execution and the transmission of $\mathcal{R}$. An assumption such as the Diffie Hellman key agreement [9] cannot be broken within half an hour, can be made with far greater degree of trust than actually assuming the long term security of a computational key agreement protocol. We consider two models, and give a cryptanalysis result for each of them:

- **The Basic BSM Scheme:** The honest parties may only interact before the broadcast of $\mathcal{R}$ (except for actually sending the encrypted message). Thus the encryption key is fully determined at the end of the broadcast of $\mathcal{R}$. Such a scheme is fully breakable in the standard BSM (without initial keys). We show that compression of SAT allows to break any basic hybrid scheme.[10]
- **The General BSM Scheme:** Alice and Bob can interact both before *and* after the broadcast of $\mathcal{R}$. In the standard BSM (without initial keys) such a scheme is breakable unless Alice and Bob use storage of size $\Omega(\sqrt{m})$. In the hybrid BSM, we show that if a compression of SAT exists then such a scheme is breakable unless Alice and Bob use storage of size $\Omega(\sqrt{m/p(n,\log m)})$, where n is the security parameter of the computational protocol and p is a polynomial (related to the polynomial of the compression algorithm and the running time of the protocol that Alice and Bob use).

Witness retrievable compression and the existence of Minicrypt: The next application is an attempt to use compression in order to prove, in the terminology of [24], that `Minicrypt=Cryptomania`. Impagliazzo [24] summarizes five possibilities for how the world may look like based on different computational assumptions. The two top worlds are `Minicrypt`, where one-way functions exist but oblivious transfer protocols do not exist (in this world some interesting cryptographic applications are possible, and in particular *shared* key cryptography exists) and `Cryptomania` where Oblivious Transfer protocols do exist (and hence also a wide range of cryptographic applications like secure computation and *public* key cryptography). Whether OT can be constructed from any one-way function is a major open problem in cryptography. Impagliazzo and Rudich [26] addressed this problem and proved that key agreement protocols (and hence also OT) cannot be constructed from any one-way function using black-box reductions.

We explore an approach of using compression in order to bridge the gap between the two worlds. In order to do so we introduce an additional requirement of a compression algorithm.

Definition 3.2 (Witness Retrievable Compression) *Let Z, L and L' define a compression algorithm as in Definition 1.2 and let R_L be an $\mathcal{NP}$ relation for L. The compression is said to be* witness retrievable *with respect to R_L if there exists a probabilistic polynomial time machine W such that if input $x \in L$ then for every witness w_x for R_L it holds that $w_y = W(w_x, Z(x))$ is a witness for $Z(x) \in L'$. We allow a negligible error in the success of W (where probability is over the internal randomness of Z and W).*

Theorem 3.3 *If there exists a witness retrievable compression algorithm for a certain type of SAT formulas, then there exists an Oblivious Transfer protocol based on* any *one-way function.*

As in the CRH construction (Theorem 3.1), the construction of OT in Theorem 3.3 is inherently non-black-box. Unfortunately we show that this approach cannot work with a compression algorithm for the *general* SAT problem, due to the following theorem:[11]

Theorem 3.4 *If one-way functions exist then there is no witness retrievable compression of SAT.*

Furthermore, we also rule out the possibility of other types of witness retrievable compression that may be sufficient for Theorem 3.3. More precisely, the inability of witness retrievable compression does not change when allowing an error in the retrieval, or when dealing with a case where there is a unique witness. These developments rule out basing the approach of Theorem 3.3 on the compression of

[10] Basic schemes are very relevant to the hybrid BSM as they include a combination of a key agreement protocol with a private key scheme (such as the scheme described by [16]).

[11] The first version of this paper [21] (dated Feb 17, 2006) did not contain this theorem and was hence more optimistic on the possibility of finding a witness preserving compression algorithm for SAT.

a general and standard language. The approach may still work out with a witness retrievable compression algorithm for the specific CNF formulas as stated in Theorem 3.3.

Finally, we point out that almost all of the examples of compression algorithms in this paper (vertex cover, PRG-output, sparse subset sum, minimum fill-in and GapSAT) are in fact witness retrievable. This demonstrates that these examples fall short of the general compression that we are seeking. In fact a major obstacle in achieving compression for problems such as SAT seems to be that most ideas are witness retrievable.

4 Basing Collision Resistant Hash Functions on Any One-Way Function

Loosely speaking, a family of length reducing functions $\mathcal{H}$ is called collision resistant hash functions (CRH) if no efficient algorithm can find collisions induced by a random member of the family. That is, no PPTM can find for a randomly chosen $h \in_R \mathcal{H}$, a pair of input strings x and x' such that $x \neq x'$ but $h(x) = h(x')$. In addition we want (i) An efficient algorithm for *sampling* from $\mathcal{H}$ using (possibly secret) randomness (the secret coins approach is potentially more powerful then when only public coins are used [23]) and (ii) An efficient evaluation algorithm that given the description of $h \in \mathcal{H}$ and x produces $h(x)$. As mentioned in the introduction, CRHs have wide cryptographic applications, see discussion and formal definitions in, for example, [27]. We are interested in basing CRH on as general assumption as possible. There is no known construction of CRH from general one-way functions or one-way permutations. Moreover, Simon [41] showed that basing CRH on one-way permutations cannot be achieved using black-box reductions. We show that compression can be used to bridge this gap.

Theorem 4.1 *If there exists an errorless compression algorithm for SAT, or for any problem that is compression-hard for $\mathcal{VC}_{OR}$, then there exists a family of Collision Resistant Hash functions (CRH) based on any one-way function.*

Proof: Let $(Commit, Verify)$ be a statistically binding computationally hiding commitment scheme based on the one-way function f. Recall that the protocol $Commit$ takes from the sender a string S and randomness r and after an interaction the receiver gets a commitment σ. The polynomial time algorithm $Verify$ takes the commitment σ and a possible opening to value S' with randomness r' and verifies that S', r' are consistent with σ. One could take for example the commitment scheme of Naor [33] based on the one-way function f.[12] In our setting we can work under the assumption that the sender (in the commitment) is honest, and in such a case, the commitment may be achieved without interaction at all.[13]

The CRH construction is inspired by the approach of Ishai, Kushilevitz and Ostrovsky [27] for constructing collision resistant hash from Private Information Retrieval (PIR). A high level description is: choose a hash function from a naive hash family with no computational hardness guarantees; in the construction below we use the selection function, i.e. a random position i. The new hash function is defined by a computationally hiding commitment to the naive hash function, and the output of the new hash function is a compression maintaining the information of the committed naive hash function when applied to the input (i.e. compression of the formula that checks that the value is what it claimed to be). Intuitively, finding a collision would require guessing with non-negligible advantage the naive hash function (the position i). The actual construction is given in Figure 1.

By the compressing properties of Z we get that h_{σ,r_Z} indeed shrinks its input (note that shrinkage by a single bit allows further shrinking by composition). We also have that sampling h_{σ,r_Z} from $\mathcal{H}$ can be done efficiently (with secret coins).

As for collisions, let $x \neq x'$ be two strings in $\{0,1\}^m$ that form a collision, i.e., $h_{\sigma,r_Z}(x) = h_{\sigma,r_Z}(x')$. This equality implies, by the property of the compression, that $\Phi_{\sigma,x}$ is satisfiable iff $\Phi_{\sigma,x'}$ is satisfiable (here we use the fact that the compression is errorless). Due to the binding property of the commitment we have that any assignment satisfying Φ_σ must have $y = i$ (recall that i is the index that σ is a commitment to). Thus the first part of $\Phi_{\sigma,x}$ is only satisfied when $y = i$. But the second part is only satisfied if $x_y = 1$, thus $\Phi_{\sigma,x}$ is satisfied if and only if $x_i = 1$. We get that $\Phi_{\sigma,x}$ is satisfiable if and only if $x_i = 1$ and $\Phi_{\sigma,x'}$ is satisfiable if and only if $x'_i = 1$. Therefore it must be the case that $x_i = x'_i$, since otherwise one of them is 0 and the other one is 1 and $\Phi_{\sigma,x}$ satisfiability is not that of $\Phi_{\sigma,x'}$. necessarily the strings x and x' are such that $x_i = x'_i$. But for some j we have $x_j \neq x'_j$ and for that j we deduce that σ is not a commitment to j.

Suppose now that we have an efficient method of finding a collision x and x' for a given (σ, r_Z). Pick any j such that $x_j \neq x'_j$. Then we know that σ is *not* a commitment to j. This procedure can be used to break the hiding properties of the commitment scheme, since it yields an efficient method that distinguishes the commitment value from random with advantage $1/m$: given (the real) i and a random one $i' \in [m]$ in a random order, run the above procedure to

[12] To be more exact, the commitment of [33] can be based on the pseudorandom generator of Håstad et al. [22] which in turn can be based on the function f.

[13] In the scheme of Naor [33], the receiver is required to provide the sender with a (public) random string. Certainly, an honest sender can generate this string by himself without harming the properties of the commitment. Thus in such a setting, the sender can generate the commitment without interaction.

CRH family $\mathcal{H}_f$:

Description of the hash function: Let Z be a compression algorithm for SAT. A function in the CRH collection is denoted h_{σ,r_Z} and defined by a commitment σ to a value $i \in [m]$, and randomness r_Z for Z. The commitment uses security parameter n (where $n << m$).

Input to h_{σ,r_Z}: a string $x \in \{0,1\}^m$

The CNF formula $\Phi_{\sigma,x}$ is defined as follows:

- Denote by $\mathrm{V}erify_\sigma$ the algorithm $\mathrm{V}erify$ with the input σ fixed. That is, $\mathrm{V}erify_\sigma$ takes as inputs y and r and accepts if and only if they form a legal opening of the commitment σ (and in particular this means that $y = i$).
- Translate $\mathrm{V}erify_\sigma$ into a CNF formula Φ_σ over the variables $y_1, ..., y_\ell$ of y and the bits of r (using Cook's reduction).
- For every $j \in [m]$ define the clause $C_{j,x} = (y_1^{\bar{j_1}} \vee y_2^{\bar{j_2}} \vee .... \vee y_\ell^{\bar{j_\ell}})$ if $x_j = 0$ (where y^0 denotes $\bar{x}$ and y^1 denotes x) and $C_{j,x} = 1$ if $x_j = 1$.
- Set
$$\Phi_{\sigma,x} = \Phi_\sigma \wedge \bigwedge_{j\in[m]} C_{j,x}$$

The hash function:

$$h_{\sigma,r_Z}(x) = Z(\Phi_{\sigma,x}, r_Z)$$

Figure 1. The construction of Collision Resistant Hash from any one-way function.

obtain j. If j equals one of the two values i or i', then guess this one as the random one and otherwise flip a coin. This contradicts our assumptions on building blocks (namely, the one-way function).

To prove the result when using compression for any language that is compression-hard for $\mathcal{VC}_{OR}$, a similar construction is defined based on the OR of small circuits rather than CNF formulas: For every $j \in [m]$ let $C_{\sigma,j}$ be the circuit that outputs one if and only if there exists randomness r such that σ is consistent with (j, r) (that is σ is a possible commitment to the value j using randomness r). Let $C_{\sigma,x}$ be the circuit that takes the OR of all $C_{\sigma,j}$ such that $x_j = 1$ and let Z be a compression algorithm for the language OR(CircuitSAT). We define $h_{\sigma,r_Z}(x) = Z(C_{\sigma,x}, r_Z)$. The proof is identical to the case of SAT. □

Note that instead of an errorless compression we can do away with an error probability slightly smaller than 2^{-m}. That is, for all x we want the probability that $Z(\Phi_{\sigma,x}, r_Z)$ preserves the satisfiability of $\Phi_{\sigma,x}$ to be at least $1 - 2^{-m+u}$ where the probability is over σ and r_Z and $u \approx \log m$. In this case we can argue (using a union bound) that with probability at least $1-2^{-u}$ no x exists violating the preservation of satisfiability.

We also note that the construction is inherently non-black box as it uses the code of the one-way function (via the commitment) in the application of Cook's Theorem. This is essential for the validity of the whole approach in light of the black-box impossibility of Simon [41]. Theorem 4.1 implies the following corollary:

Corollary 4.2 *If there exists an errorless compression algorithm for SAT or for any problem that is compression-hard for $\mathcal{VC}_{OR}$, then there exist* statistically hiding, computationally binding commitment *schemes based on any one-way function.*

The corollary follows since CRH imply statistically hiding bit commitment, see Naor and Yung [36] (and Damgård, Pedereson and Pfitzman [8] for commitment to many bits). As mentioned in the introduction, the currently known minimal assumptions for constructing statistically hiding bit commitments are the existence of one-way permutations [35] and the more general one-way functions with known pre-image size [19]. Furthermore, the commitment schemes of [35, 19] require many rounds of interaction (at least linear in the security parameter), while the commitments based on CRHs are non-interactive, at least after the initial phase where the function $h \in \mathcal{H}$ is chosen.

5 Related Work and Further Issues

Related Work: The relationship between compression and complexity in general is a topic that has been investigated since the early days of Complexity Theory (i.e. Kolmogorov Complexity [30]). However, the feature that we are introducing in this work is compressibility with respect to the *solution* (witness) rather than the instance. The goal of maintaining the solution differs our work from a line of seemingly related works about notions of compression ([14, 42, 43] to name a few), all of which aim at eventually retrieving the input of the compression algorithm.

There are several examples of other relaxations to solving $\mathcal{NP}$ problems in polynomial time. Each of these relaxations yields a corresponding classifications of $\mathcal{NP}$. These classifications, like ours, are subtle and usually turn out to be different than the traditional $\mathcal{NP}$ classification. For example, Papadimitriou and Yannakakis [39] introduce L-reductions and the classes MAX NP and MAX SNP, with respect to approximation algorithms. Impagliazzo, Paturi and Zane [25] define reductions with respect to solution in sub-exponential time.

Perhaps the most relevant classification to ours is that of parameterized complexity (see the monograph on this subject by Downey and Fellows [12]). Parameterized complexity studies the tractability of problems when one of the parameters is considered to be fixed or very small. This is relevant to compression since typically this parameter is related

to the length of the witness. On the one hand, some (but not all) parameterized complexity algorithms yield natural compression algorithms (see examples and discussion in Section 2.1). In addition, some (but certainly not all) compression algorithms may imply a parameterized complexity algorithm. Also the W-hierarchy of parameterized complexity is reminiscent of the $\mathcal{VC}$-hierarchy (they are both defined by reduction to circuits of bounded depth). However, our study of compression yields quite a different classification. This is mainly because in parameterized complexity the witness length is taken to be very small and as such, there is no restriction on running in time that is exponential (or higher) in this parameter. In compression, on the other hand, the parameter (witness length) is usually of substantial size (even if much smaller than the instance length).

A related notion to parameterized complexity that is reminiscent of our work is *limited non-determinism*, which started with the work of Kintala and Fischer [29], see survey by Goldsmith, Levy and Mundheck [17]. This was further studied by Papadimitriou and Yannakakis [40] who defined a few syntactic classes within the class of polylog non-determinism ($LOGNP$ and $LOGSNP$). The interesting point is that several natural problems are complete for these classes. The notion of reduction used is the usual polynomial reduction and hence the classifications arising from this study are very different from our $\mathcal{VC}$ hierarchy.

Subsequent Works: Dubrov and Ishai [13] discussed the compression of problems and showed that a certain incompressibility assumption has an application to derandomization. Specifically they construct a generator that fools procedures that use more randomness than their output length. Their work was mostly conducted independently of ours, following conversations regarding an early phase of our work. In addition, inspired by our CRH construction, they prove that any one-way permutation can either be used for the above mentioned derandomization, or else can be used to construct a weak version of CRH.[14]

In a recent paper, Dziembowski [15] shows the relevance of our notion of witness retrievable compression to a method for achieving *forward-secure storage*. He shows a cryptanalytic result of such compression. Furthermore, following our approach for construction of OT from one-way functions, he shows that for every one-way function either a specific storage scheme is forward-secure, or there exists an OT protocol based on this one-way function.

5.1 Discussion and Open Problems

The issue of compressibility and the corresponding classification introduced in this work raise many open problems and directions. The obvious one is to come up with a compression algorithm for a problem like SAT or Clique (or some $\mathcal{VC}_{OR}$ complete or hard problem). Alternatively, show why such tasks are infeasible. We have seen compressibility of some interesting $\mathcal{NP}$ languages and hence the question is where exactly is boundary between compressibility and incompressibility. We tend to conjecture that it is in the low levels of the $\mathcal{VC}$ hierarchy. We view PCP amplification methods such as the recent result of Dinur [11] as potential leads towards achieving compression. This is since these results show a natural amplification of properties on a graph, and could potentially be combined with a simple compression of promise problems (such as the example for GapSAT) The main issue is doing the PCP amplification without introducing many new variables.

In particular, the following task would suffice for achieving non-trivial compression: given CNF formulae ϕ_1 and ϕ_2 (not necessarily with short witnesses) come up with a formula ϕ that is (1) satisfiable if and only if $\phi_1 \vee \phi_2$ is satisfiable and (2) shorter than $|\phi_1| + |\phi_2|$. Moreover, due to the impossibility results for general witness retrievable compression, a witness for either ϕ_1 or ϕ_2 cannot efficiently yield a witness for ϕ.

Short of showing a compression for general complexity classes, it would be interesting to come up with further interesting compression algorithms as well as to obtain more hardness results. For instance, is Clique or any other embedding problem complete for $\mathcal{VC}_1$? Is there a natural and simple complete problem for $\mathcal{VC}_1$? Also, the $\mathcal{VC}$ hierarchy is by no means the ultimate classification with respect to compressibility. One can hope to further refine this classification, especially within the confines of $\mathcal{VC}_1$.

Since we currently do not have a general compressibility result for a significant class of languages, it is important to understand what are the implications of *incompressibility*. The application to the bounded storage model can be viewed as such a statement. Other examples are the previously mentioned works of Dubrov and Ishai [13] regarding derandomization and Dziembowski [15] with respect to forward-secure storage. In order to gain confidence in an incompressibility assumption when used in a cryptographic setting it is important to come up with an *efficiently falsifiable* assumption of this nature (see [34]).

Finally we feel that we have just scratched the surface of an important topic and in the future there will be other implications of compressibility or the impossibility of compression, whether in cryptography or in other areas.

Acknowledgements: We thank Yuval Ishai for many helpful comments and specifically for pointing out that the CRH construction does not require witness retrievability. We are also grateful to Hovav Shacham for conversations related to witness retrievable compression. Finally we thank Alon Rosen, Ronen Shaltiel, Gillat Kol and the anonymous referees for their helpful comments and suggestions.

[14] This weak version of CRH (like the stronger common version) cannot be constructed from any one-way permutation by black-box reductions. (in [41]).

References

[1] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, 36(1):30–52, 2005.

[2] Y. Aumann, Y. Ding, and M. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[3] B. Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115, 2001.

[4] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT ' 2004, LNCS*, volume 3027, pages 171–188. Springer, 2004.

[5] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[6] S. Cook. The complexity of theorem-proving procedures. In *3rd STOC*, pages 151–158, 1971.

[7] I. Damgård. A design principle for hash functions. In *CRYPTO '89, LNCS*, volume 435, pages 416–427. Springer, 1989.

[8] I. Damgård, T. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *CRYPTO '93, LNCS*, volume 773, pages 250–265. Springer, 1993.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22:644–654, 1976.

[10] Y. Ding and M. Rabin. Hyper-encryption and everlasting security. In *STACS, LNCS*, volume 2285, pages 1–26, 2002.

[11] I. Dinur. The PCP theorem by gap amplification. In *38th STOC*, pages 241–250, 2006.

[12] R. Downey and M. Fellows. **Parameterized Complexity**. Springer-Verlag, 1999.

[13] B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *38th STOC*, pages 711–720, 2006.

[14] C. Dwork, J. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information. In *28th STOC*, pages 489–498, 1996.

[15] S. Dziembowski. On the forward-secure storage. To appear in *CRYPTO*, 2006.

[16] S. Dziembowski and U. Maurer. On generating the initial key in the bounded-storage model. In *EUROCRYPT ' 2004, LNCS*, volume 3027, pages 126–137. Springer, 2004.

[17] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20–29, 1996.

[18] S. Goldwasser and Y. Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–111, 2003.

[19] I. Haitner, O. Horvitz, J. Katz, C. Koo, R. Morselli, and R. Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *EUROCRYPT ' 2005, LNCS*, volume 3494, pages 58–77. Springer, 2005.

[20] D. Harnik and M. Naor. On everlasting security in the *hybrid* bounded storage model. In *33rd ICALP 2006, Part II, LNCS*, volume 4052, pages 192–203. Springer, 2006.

[21] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. In *ECCC, TR06-022*, 2006.

[22] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 29(4):1364–1396, 1999.

[23] C. Hsiao and L. Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO '04, LNCS*, volume 3152, pages 92–105. Springer, 2004.

[24] R. Impagliazzo. A personal view of average-case complexity. In *10th Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society Press, 1995.

[25] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *39th FOCS*, pages 653–663, 1998.

[26] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st STOC*, pages 44–61, 1989.

[27] Y. Ishai, E. Kushilevitz, and R. Ostrovsky:. Sufficient conditions for collision-resistant hashing. In *TCC '05*, volume 3378 of *LNCS*, pages 445–456, 2005.

[28] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th STOC*, pages 723–732, 1992.

[29] C. Kintala and P. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal of Computing*, 9(1):46–53, 1980.

[30] M. Li and P. Vitányi. **An Introduction to Kolmogorov Complexity and Its Applications**, *2nd Edition*. Springer, 1997.

[31] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[32] S. Micali. CS proofs. In *35th FOCS*, pages 436–453, 1994.

[33] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[34] M. Naor. On cryptographic assumptions and challenges. In *CRYPTO '03, LNCS*, volume 2729, pages 96–109. Springer, 2003.

[35] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

[36] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st STOC*, pages 33–43, 1989.

[37] J. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO '02, LNCS*, volume 2442, pages 111–126. Springer, 2002.

[38] C. Papadimitriou. **Computational Complexity**. Addison-Wesley, 1994.

[39] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *20th STOC*, pages 229–234, 1988.

[40] C. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *JCSS*, 53(2):161–170, 1996.

[41] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT ' 1998, LNCS*, volume 1403, pages 334–345. Springer, 1998.

[42] L. Trevisan, S. Vadhan, and D. Zuckerman. Compression of samplable sources. In *IEEE Conference on Computational Complexity*, pages 1–14, 2004.

[43] H. Wee. On pseudoentropy versus compressibility. In *IEEE Conference on Computational Complexity*, pages 29–41, 2004.

[44] H. Wee. On obfuscating point functions. In *37th STOC*, pages 523–532, 2005.