The security of public-key cryptosystems and the complexity of promise problems

Diplomarbeit

von

Lennart Suhr

May 9, 2014

Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt habe.

Lennart Suhr

Abstract

Over the course of four years, Even, Selman and Yacobi [EY80, SY82, ESY84] came up with a conjecture about the existence of certain NP-hard promise problems and their relation to the security of public-key cryptography. The conjecture, if it were true, has several interesting consequences such as $\mathsf{P} \neq \mathsf{NP}$ and that no non-probabilistic public-key cryptosystem is able to comprise an NP-hard cracking problem. Moreover, it is also related to the theory of propositional proof systems, boolean formulas and some other complexity issues, strengthening the impression that it could be really hard to prove. Though the conjecture still remains open, there has been some exciting progress and evidence recently [HMPRS12].

Contents

1 Introduction

More than thirty years have passed since Even, Selman and Yacobi [EY80, SY82, ESY84] stipulated their conjecture on promise problems, which was originally raised to concern the security of public-key cryptosystems. In more detail, they asserted that no underlying cracking problem could be NP-hard at all. Beyond these main intention, the so-called ESY conjecture comes up with a wide range of different consequences, as we will see in one of the following sections, and also can be reformulated to deal with the more famous disjoint NP-pairs instead of promise problems. Up until now, there has been a lot of progress regarding the conjecture itself or the concepts involved. Before investigating these results, we will take a closer look at what it is all about.

1.1 Disjoint NP-pairs and promise problems

There is a fundamental relation between promise problems and disjoint NP-pairs and that is why the latter will be considered first.

Disjoint NP-pairs play an important role in the theory of proof complexity and particularly when considering propositional proof systems [Raz94, Pud01, GSZ07] such as resolution or natural deduction. A pair (A, B) belongs to the class DisNP if A and B are two nonempty, disjoint sets in NP. Razborov, for example, showed that the existence of an optimal propositional proof system (which is a proof system that is able to simulate all other proof systems) implies the existence of a complete pair for DisNP and raised the question whether we can find one. The belief is that there do not exist optimal propositional proof systems, which in turn would imply that a variant of the previously mentioned conjecture is true [GSSZ04]. As a further example, several questions have come up as to whether an p-inseparable disjoint pair (A, B) exists within DisNP, meaning that there is no separator belonging to P (a separator is any superset of A that is disjoint from B). Interestingly enough, this was connected to the existence of NP-hard promise problems and moreover led to a different appearance of the conjecture [GS88].

Promise problems (or at least complexity issues for promise problems) were first investigated by S. Even and Y. Yacobi in 1980 [EY80] originally modeling cracking problems for public-key cryptosystems (short: PKCSs) and, among others, have quite recently come into force in quantum computation. Though decision problems tend to be the standard approach for perceiving problems, it is way more natural to talk about the relating promise problem instead (see [Gol06] for a proper discussion). Often one does not want to decide a property for all strings out of $\{0,1\}^*$, but only for the ones that already fulfill a certain promise.

Definition 1.1. A *promise problem* is a disjoint pair of sets (Π_{yes}, Π_{no}) with $\Pi_{yes}, \Pi_{no} \subseteq \{0,1\}^*$. The set $\Pi_{yes} \cup \Pi_{no}$ is called the *promise.*

Π_{yes} represents the set of strings that belong to the promise and fulfill a specific property, whereas a string belongs to Π_{no} if it is promised but does not have the property. By taking a look at a common decision problem such as SAT the approach of considering promise problems becomes more convincing. For example, one may be interested in the satisfiability of Boolean formulas that have at most one satisfying assignment. This problem is called $\mathsf{Unambiguous - SAT}$ and is shown to stay computationally hard, even if the mentioned promise is given, by the Valiant–Vazirani theorem [VV86] (if there is a polynomial-time algorithm for $\mathsf{Unambiguous - SAT}$, then $\mathsf{NP} = \mathsf{RP}$). Hence, $\Pi_{yes} \cup \Pi_{no}$ would be the set of all (binary representations of) Boolean formulas that have at most one satisfying assignment and Π_{yes} the subset of the ones being satisfiable. Classically it would be $\Pi_{yes} \cup \Pi_{no} = \{0,1\}^*$ and that is when we say that the promise is *trivial.*

On the other hand, there are some reasons why promise problems did not prevail (every decision problem with inputs over $\{0,1\}^*$ could be handled as promise problem with trivial promise). For example, when it comes to some well-known structural relations which need not hold consistently: the existence of an NP-hard promise problem within $\mathsf{NPP} \cap \mathsf{coNPP}$ (as promise complexity class extensions of NP and coNP) does not seem to imply $\mathsf{NP} = \mathsf{coNP}$, which means that there truly are some relevant differences between promise and decision problems apart from being different ways of looking at problems.

The standard notion of complexity classes for languages extends naturally towards promise problems because every promise $\Pi_{yes} \cup \Pi_{no}$ can be seen as a decision problem and every decision problem may be considered to be a promise problem (every language $L \subseteq \Sigma^*$ extends to the promise problem $(L, \Sigma^* \backslash L)$. To distinguish between complexity classes for decision problems and those for promise problems, Selman and Yacobi formerly introduced the classes NPP and coNPP as extensions of NP and coNP [SY82]. Before coming to these, we need to define solutions for promise problems.

Definition 1.2. A deterministic Turing machine M that decides whether an already promised input x belongs to Π_{yes} or Π_{no} is said to *solve* the promise problem (Π_{yes}, Π_{no}):

$$\forall x \, [x \in \Pi_{yes} \cup \Pi_{no} \Rightarrow [M(x) \downarrow \wedge (M(x) = 1 \Leftrightarrow x \in \Pi_{yes})]].$$

Note that for inputs outside the promise, it does not matter whether the Turing machine says "yes" or "no". At least until the following definition.

Definition 1.3. A *solution* of a promise problem is the set of inputs being accepted by a Turing machine that solves the promise problem, therefore any set S that includes Π_{yes} and is disjoint from Π_{no}.

In this manner, solutions to promise problems can be handled set-theoretically, though not every solution needs to be recursive by definition. However, since we are focusing on polynomial-time complexity issues, we will assume in the following that if a Turing machine M solves a promise problem then it halts on every input, leading to recursive solutions only. Now we are able to state the aforementioned complexity classes for promise problems.

Definition 1.4. NPP is the class of promise problems (Π_{yes}, Π_{no}) that have a solution in NP and coNPP is the class of promise problems, so that (Π_{no}, Π_{yes}) is in NPP.

Note that a language $L \in \mathsf{NP}$ is a solution to (Π_{yes}, Π_{no}) if and only if $\overline{L} \in \mathsf{coNP}$ is a solution to (Π_{no}, Π_{yes}).

Definition 1.5. A promise problem is NP-*hard* if it is solvable and every solution is NP-hard.

The property for NP-hard promise problems of being solvable is necessary because otherwise, every unsolvable promise problems would be NP-hard since the set of solutions then is the empty set.

There is also another way to treat the complexity classification, such as that a promise problem, for example, belongs to P in the sense of decision problems, if there is a deterministic polynomial-time algorithm A satisfying

$$x \in \Pi_{yes} \Rightarrow A(x) = 1 \text{ and}$$

$$x \in \Pi_{no} \Rightarrow A(x) = 0$$

(compare [Gol06]). Since we are talking about a polynomial-time algorithm, the responding Turing machine halts on every input and therefore every solution of the promise problem belongs to P, which coincides with the suiting former definition. It depends on personal preferences which definition to work with. Since we partly need to focus on the original papers, we are best going with the former one.

Alternatively to the pair (Π_{yes}, Π_{no}) a promise problem can be written (and formerly was written) as a pair of predicates (Q, R) with promise Q and property R (being a superset of Π_{yes}). The transformation from one to another is quite simple:

$$\begin{aligned}
(\Pi_{yes}, \Pi_{no}) &\longmapsto (\Pi_{yes} \cup \Pi_{no}, R), \text{ while } R \supseteq \Pi_{yes} \text{ and} \\
(Q, R) &\longmapsto (Q \cap R, Q \setminus R)
\end{aligned}$$

(compare with the picture in Section 1.2). This denotation makes sense since a promise problem then is directly denoted by its relating objects (for example could Q be the set of Hamiltonian graphs and R be the set of 3-colorable ones). Both variants describe the same promise problem and will come into force in two apparently different conjectures.

1.2 ESY conjecture and implications

We now state the ESY conjecture in its current form which is in the manner of disjoint NP-pairs (recall the relation between promise problems and disjoint pairs).

ESY conjecture. For every two disjoint languages $A, B \in \mathsf{NP}$, there is a separator S, meaning $A \subseteq S$ and $B \cap S = \emptyset$, that is not Turing-hard for NP.

Or in other words, there do not exist disjoint NP-pairs all of whose separators are NP-hard (via Turing reductions). The following equivalent (see [GS88]) conjecture is the original one as introduced by Even, Selman and Yacobi in 1984 [ESY84]:

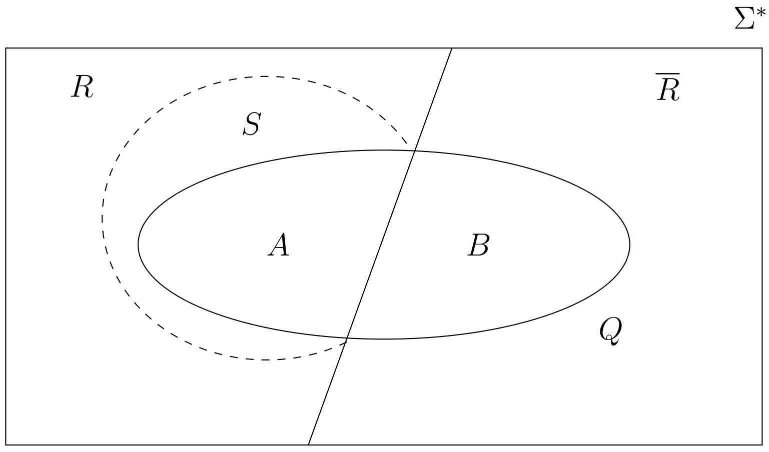
ESY conjecture (previously). There exists no promise problem (Q, R) such that

(i) $(Q, R) \in \mathsf{NPP} \cap \mathsf{coNPP}$,
(ii) (Q, R) is NP-hard, and
(iii) Q is in NP.

Again, we are talking about NP-hardness via Turing reductions (we will generalize the reduction types in Chapter 3).

Two years before, in the year 1982 where the conjecture first came up, there had not yet been a point (iii). However, with the promise problem $(Q, R) := (\mathsf{EX}, \mathsf{SAT1})$, where $\mathsf{EX}(x, y) = 1 \Leftrightarrow \mathsf{SAT}(x) \oplus \mathsf{SAT}(y)$ and $\mathsf{SAT1} := \lambda x \lambda y \mathsf{SAT}(x)$ so that $\mathsf{SAT1}(x, y) \Leftrightarrow \mathsf{SAT}(x)$ ($\mathsf{SAT}(x)$ asserts that x is satisfiable), there was an example which is both in $\mathsf{NPP} \cap \mathsf{coNPP}$ and NP-hard (see [SY82]) forcing them to reformulate it. Since EX has been shown to be $\mathsf{D^P}$-complete [ESY84], where $\mathsf{D^P} := \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP} \text{ and } L_2 \in \mathsf{coNP}\}$, they subsequently conjectured that Q cannot be in NP (note that $\mathsf{NP} \subseteq \mathsf{D^P}$).

To see the equivalence of the newer and former conjecture simply observe that if (Q, R) actually is a promise problem that satisfies the conditions of the former conjecture, it naturally extends to the disjoint NP-pair $(Q \cap R, Q \setminus R)$. Because every separator S then is a solution to (Q, R), it is also NP-hard by condition (ii). Vice versa, every disjoint NP-pair (A, B) having only NP-hard separators provides a suiting (former) promise problem $(A \cup B, A)$.



Note that the separator or solution S does not only need to contain inputs satisfying property R. If $\mathsf{NP} = \mathsf{coNP}$ then the conjecture is false, because by taking A to be SAT and B to be its complement $\overline{\mathsf{SAT}}$, then the only possible separator would again be SAT, which is NP-hard for sure (we will come to this later).

This commonly unknown and by now more than thirty-years-old conjecture, named after the initials of its originators, has some wide-ranging and fascinating consequences apart from $\mathsf{NP} \neq \mathsf{coNP}$ (and therefore $\mathsf{P} \neq \mathsf{NP}$) concerning complexity theory, Boolean formulas and public-key cryptography, making it obviously hard to prove.

The conjecture, for example, does imply that satisfying assignments of Boolean formulas cannot be computed by single-valued NP-machines, thus $\mathsf{SAT} \notin \mathsf{NPSV}$ [ESY84, GS88], and also has an impact on the task of computing functions and finding solutions when they are unique. If one knows in advance given any x, that at most one y will satisfy a feasibly recognizable property $R(x, y)$, then define

$$A := \{\langle x, w\rangle \mid \exists y \text{ such that } R(x, y) = 1 \text{ and } w \text{ is a prefix of } y\},$$

$$B := \{\langle x, w\rangle \mid \exists y \text{ such that } R(x, y) = 1 \text{ and } w \text{ is } \textbf{not} \text{ a prefix of } y\}.$$

An important case for this is factoring, where x is a number and y is a unique representation of its prime factorization. Then (A, B) is a disjoint NP-pair and any separator C can be used to find y by building up w character-by-character using Turing queries to C [Reg12]. Remember that UP stands for *Unambiguous non-deterministic Polynomial-time* and is the class of all decision problems solvable by a non-deterministic Turing machines such that

(1) if the answer is "yes", exactly one computational path accepts,

(2) if the answer is "no", all computational paths reject.

Observe that $\mathsf{P} \subseteq \mathsf{UP} \subseteq \mathsf{NP}$ and that an example for a language in UP is the previous mentioned $\mathsf{Unambiguous - SAT}$. Thus if the conjecture is true, then $\mathsf{NP} \neq \mathsf{UP}$ and there, for example, would be no way of reducing the effort for the determination of satisfying assignments to SAT down to a unique choice. On the contrary, if the task of finding y (when it exists) is always NP-hard then the conjecture is false.

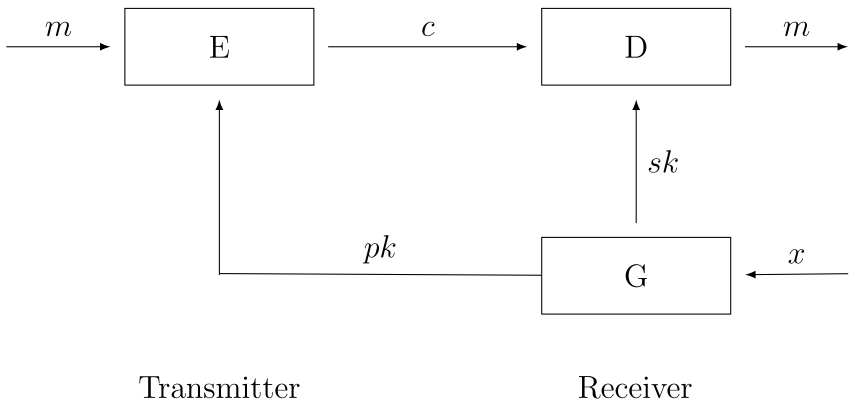
One of the most interesting implications of the conjecture and the main motivation of this paper is that if the conjecture holds, then no cracking problem of a PKCS that can be formulated as disjoint NP-pair or promise problem respectively could be NP-hard to crack. RSA for example is one of these systems fitting the model and that may be one of the reasons why there had been no real competitor yet based on an NP-hard cracking problem. On the other hand, *probabilistic* encryption is said to be unaffected by the conjecture and thus being a disadvantage of it. In Section 2.2 we will falsify this assumption and show that if the conjecture is true and the considered cryptosystem additionally fulfills a certain condition, then it also does not comprise an NP-hard cracking problem.

To date there is no hypothesis known that implies the ESY conjecture or all of its consequences, fueling its power. Even so there has been some exciting progress on variants of the conjecture recently. We will take a detailed look at this in Chapter 3.

1.3 Public-key cryptosystems

In the year 1976, Diffie and Hellman [DH76] introduced a new variant of encryption which became widely known as public-key cryptography or, in distinction to previous symmetric approaches, as asymmetric encryption. The large benefit of public-key encryption consists of not relying on any hidden key exchanges in contrast to classical symmetric cryptosystems, where the same key is used both for encryption and decryption (and therefore must be known to all participants). While public-key approaches often require more runtime than symmetric ones (at least for large messages), it is not unusual to combine both in hybrid systems, meaning public-key encryption for the transmission of keys for symmetric systems. However, all of these three possibilities are currently applied in many different algorithms. We will now come to a more formal description of the public-key approach:

A *public-key cryptosystem* (PKCS) is a triple (E, D, G) with *encryption function* E, *decryption function* D and *key-generator* G. All of these functions are publicly-known and can be computed in polynomial-time. The key-generator delivers a pair of keys (pk, sk) containing pk as *public key* and sk as *secret key* by processing $G(x)$ for a randomly generated string x. The secret key remains concealed and is only known to the receiver. For a (binary encoded) plain-text message m, one may now compute $c = E(m, pk)$ and afterward send this *cryptogram* c to the receiver who then determines $m = D(c, sk)$.



(based on [ESY84], p.161.)

One of the most commonly known PKCSs is RSA, named after the initials of its originators Rivest, Shamir and Adleman [RSA78]. For the sake of completeness and to provide an impression on how a PKCS may work in detail, we now presume to shortly describe the general RSA-cryptosystem (based on [Bey11]). RSA works as followed:

(1) Generation of two large prime numbers p and q (commonly 1024, 2048 or 4096 bit) and multiplication to receive $n = p \cdot q$.

(2) Determination of two exponents $e, d \in \mathbb{N}$ such that $e \cdot d \equiv_{\text{mod } \varphi(n)} 1$, where φ denotes Euler's φ-function $\varphi(n) := |\mathbb{Z}_n^*|$. Then $pk = (n, e)$ and $sk = (n, d)$.

(3) Splitting of the plaintext message m into parts $x_1, ..., x_k$ such that $x_i < n$ for $1 \leq i \leq k$.

(4) Encryption of each of those parts with $E(x_i, (n, e)) = x_i^e \bmod n$ and transmission to the receiver.

(5) Decryption of each of the cryptograms c_i with $D(c_i, (n, d)) = c_i^d \bmod n$.

The computation of a pair (e, d) that satisfies the properties out of step two is quite easy and can be done with the help of the extended Eucledian algorithm. Function E and D provide efficient computation with multiple squaring and multiplication. If an eavesdropper manages to compute one of the prime numbers p, q or the exponent d with the help of the information (n, e) and c (the knowledge of d enables easy factorization of n), he is able to decrypt any ciphertext within that session. Thus the security of RSA is based on the complexity of factorizing large integer numbers (the problem is known as FACT), for which it is neither known to be NP-complete nor if it can be done in polynomial-time. It is assumed that $\mathsf{FACT} \in \mathsf{NPI}$ where NPI stands for NP-Intermediate and is the class of all problems that are are contained in $\mathsf{NP} \setminus \mathsf{P}$ not being NP-complete. If $\mathsf{P} \neq \mathsf{NP}$ then $\mathsf{NPI} \neq \emptyset$ by Ladner's theorem (therefore $\mathsf{P} \neq \mathsf{NP} \Leftrightarrow \mathsf{NPI} \neq \emptyset$) and this may provide some evidence.

Finally, there also exist probabilistic public-key cryptosystems, as mentioned before, which additionally contain the factor of randomness. We will investigate those and deliver a recent example of a probabilistic PKCS with Gentry's *homomorphic encryption* [Gen09a] in Section 2.2. The existence of such a fully homomorphic encryption system is quite fascinating because at first, no one was ever able to construct one before and second, this provides several possibilities for the encryption and decryption process, as we will see in the relating section. Nevertheless, we will show that if the ESY conjecture holds, it cannot be NP-hard to crack either.

2 Security of public-key cryptosystems

We said that if the ESY conjecture holds, then no cracking problem of a PKCS that can be formulated as promise problem or disjoint NP-pair can be NP-hard. In this chapter we will take a closer look at these transformation and why this is implied. Furthermore, we will investigate probabilistic public-key encryption which is said to be unaffected by the conjecture and then provide a proof that under a certain assumption, some of the well-known probabilistic cryptosystems cannot be NP-hard to crack too.

2.1 Classical public-key cryptosystems

For the purpose of a comprehensible derivation and to illustrate the former chain of thoughts, in this section, the construction of the responding promise problem and the connection to the ESY conjecture will be done in the classical form for promise problems as originally considered by Even, Selman and Yacobi, meaning (Q, R) instead of (Π_{yes}, Π_{no}) (keep in mind that they are convertible into each other). In Section 2.2 we will again work with the disjoint notation for promise problems and therewith deliver two possible approaches. This intention may be a bit confusing, but since we want to gain a more encompassing impression on promise problems it may be quite reasonable.

As stated in the introduction, recall that a PKCS is a triple (E, D, G) with encryption function E, decryption function D and key generator G such that for a pair of keys (pk, sk) generated by G and a plain-text message m the transmitter may compute $E(m, pk) = c$ and afterward send this cryptogram to the receiver (through an open channel) who then determines $D(c, sk) = m$. The connection between a classical PKCS and promise problems can be derived by considering its relating *cracking problem* (short: CP) which is the problem of computing m (under knowledge of E, D, G, pk and c) such that $E(m, pk) = c$. A PKCS can be deemed secure if this computational task leads to enormous and impractical long runtime. Since function E is not forced to be onto, there may be no m satisfying the conditions. However, E is one-one, which means that if there is such an m, then it is unique. This problem can, at first, be formulated as decision problem in the following way:

$$CP_D := \left\{ \langle m', pk, c \rangle \;\middle|\; \begin{array}{l} m' \geq m, \text{ where } m \text{ is the message that} \\ \text{satisfies } E(m, pk) = c \end{array} \right\}$$

The inequality $m' \geq m$ is reasonable because if we can say whether the numerical value of m' is greater or equal than the numerical value of m (encoded as binary numbers) then we are also able to determine the plain-text message via binary search.

To receive the related promise problem, we only need to add the promise condition to the decision version of the cracking problem, which is the property of the existence of such x and m.

$$\begin{aligned} \text{INPUT}: &\quad \langle m', pk, c \rangle, \\ \text{PROMISE } Q: &\quad \exists x, m \text{ such that } G(x) = (pk, sk) \text{ and } E(m, pk) = c, \\ \text{PROPERTY } R: &\quad m' \geq m, \text{ where } m \text{ is the message which satisfies} \\ &\quad E(m, pk) = c. \end{aligned}$$

This generates the promise problem CP_P, where $CP_P = (Q, R)$. Thus we have easily seen that the cracking problem of any classical PKCS can be formulated as a promise problem. We will now show that every cracking promise problem is in $\mathsf{NPP} \cap \mathsf{coNPP}$. A promise problem belongs to $\mathsf{NPP} \cap \mathsf{coNPP}$ if and only if it has a solution in NP and a solution in coNPP. The first property is given by concerning a non-deterministic polynomial-time Turing machine M that on inputs of form $\langle m', pk, c \rangle$, guesses m and x and if the promise holds, simply checks whether $m' \geq m$. Since M solves the promise problem and halts on every input, the language accepted by M is recursive and belongs to NP, thus $CP \in \mathsf{NPP}$. To see the second part, simply take M again but now check whether $m < m'$. Call this Turing machine M'. Observe that for every language L accepted by M, there is a language L' accepted by a Turing machine M' described as above so that $\overline{L} = L'$. Therefore there also exists a solution in coNP which leads to $CP \in \mathsf{coNPP}$ and finally to $CP \in \mathsf{NPP} \cap \mathsf{coNPP}$.

To imply the aforementioned consequences for PKCSs, it is necessary to show that Q is in NP. To do so, we need to concern Q as decision problem (recall that every promise can be seen as decision problem):

$$Q = \{ \langle m', pk, c \rangle \mid \exists x, m \text{ such that } G(x) = (pk, sk) \text{ and } E(m, pk) = c \}$$

Clearly, Q is polynomial verifiable. A solution or proof for an input would be the pair

$\langle x, m \rangle$ satisfying the conditions of the language. Since G and E are publicly known and can be computed in polynomial time, these can easily be verified. So $Q \in \mathsf{NP}$.

Because $CP_P \in \mathsf{NP} \cap \mathsf{coNP}$ and $Q \in \mathsf{NP}$, the cracking promise problem fulfills (i) and (iii) of the ESY conjecture and thus cannot be NP-hard (property (ii)) if the conjecture holds.

Obviously, this result would not cause a heavy impact on practical encryption. Clearly, the conjecture additionally implies $\mathsf{P} \neq \mathsf{NP}$ and therefore may provide some evidence for a problem like FACT to be in NPI (which means not in P), but we neither can confirm that cracking problems of PKCSs provide polynomial-time algorithms nor that they do not (even if a cracking problem is in P, this does not automatically lead to insecurity of the relating PKCS). The only thing to be sure about is that no PKCS can be NP-hard to crack, which is truly interestingly enough for theoretical aspects by itself.

One central statement out of that recent paper on the ESY conjecture [HMPRS12] is that the conjecture, in contrast to certain assumptions, does even affect probabilistic approaches, as we will now see in the next section.

2.2 Probabilistic public-key cryptosystems

A *probabilistic public-key cryptosystem* (E, D, G) (probabilistic PKCS) works the same way a classical PKCS does, despite the fact that the encryption function E (in general) additionally depends on a random generated string r. Thus we have $E(m, r, pk) = c$ and $D(c, sk) = m$ if c is a valid cipher text for m. Note that one normally receives different cipher texts for the same message because of the randomness yielded by r.

Due to its (more or less) recent relevance and specific properties, we will now consider an interesting example of probabilistic encryption with *Gentry's homomorphic cryptosystem* [Gen09a]. The outstanding feature of Gentry's encryption scheme is the property of being fully homomorphic, prior also known as private homomorphic, which is something that no other cryptosystem was able to provide before. However, there came up plenty of partially homomorphic cryptosystems (we will see the difference between those properties later) like Elgamal [Elg85], Goldwasser-Micali [GM84] or even RSA [RSA78] and thus the question arose whether there even exists a fully homomorphic one. We will now investigate the corresponding advantages and what this means in particular.

The following part contains some definitions and notations out of Gentry's paper [Gen09a]. For any cryptosystem one is interested in the question whether it is possible to perform some processing on the encrypted plaintext (querying, writing into it or simply anything that can be efficiently expressed as a circuit) only with the use of the public key and without the necessity of decrypting or of knowing the content, as if it would have been done before the encryption. If a cryptosystem is fully homomorphic, then it provides this possibility and it would not matter whether something is applied to the message before the encryption or afterward - the resulting decrypted ciphertext would be the same. This question was originally raised by R. Rivest in 1978 and led to some serious considerations. As an application there had been several ideas given such as private data banks (or cloud-computing in general) where users can store encrypted data on an untrusted server and allow the server to process on and respond to user's data queries without the need of additional expensive client-depending interactions for decryption. Since then, plenty more possible ideas were listed and for sure, the reader may imagine some by himself.

Gentry describes his cryptosystem as follows. A fully homomorphic encryption scheme $\mathcal{E}$ basicly consists of three algorithms:

$$\begin{aligned}
\mathsf{KeyGen}_{\mathcal{E}}(\lambda) = (pk, sk) :&\ \text{The key generator using a security parameter } \lambda.\\
\mathsf{Encrypt}_{\mathcal{E}}(\pi_i, pk) = \psi_i :&\ \text{Encryption function having the plaintext and}\\
&\ \text{the public key as input.}\\
\mathsf{Decrypt}_{\mathcal{E}}(\psi_i, sk) = \pi_i :&\ \text{Decryption function for ciphertexts and suiting}\\
&\ \text{secret keys.}
\end{aligned}$$

Moreover, the cryptosystem has an additional fourth algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ for processing encrypted data. For *any* circuit C of a permitted set of circuits $C_{\mathcal{E}}$, any valid public key pk and any ciphertexts π_i with $\mathsf{Encrypt}_{\mathcal{E}}(\pi_i, pk) = \psi_i$, it needs to hold that:

$$\mathsf{Evaluate}_{\mathcal{E}}(\psi_1, ..., \psi_t, C, pk) = \psi,$$

where ψ is a valid encryption of $C(\pi_1, ..., \pi_t)$ under pk. The computational complexity of those algorithms should not exceed $\lambda^{O(1)}$. We concern the following relating definition.

Definition 2.1. $\mathcal{E}$ is *correct* for circuits in $C_{\mathcal{E}}$ if for any key-pair (pk, sk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in C_{\mathcal{E}}$, any plaintexts $\pi_1, ..., \pi_t$, and any ciphertexts $\Psi = \langle \psi_1, ..., \psi_t \rangle$ with $\mathsf{Encrypt}_{\mathcal{E}}(pk, \pi_i) = \psi_i$, it is the case that

$$\mathsf{Evaluate}_{\mathcal{E}}(\Psi, C, pk) = \psi \Rightarrow C(\pi_1, ..., \pi_t) = \mathsf{Decrypt}_{\mathcal{E}}(sk, \psi).$$

Observe that the property of being correct is quite exactly the possibility that we have with a fully homomorphic cryptosystem as we described in the introductory part of Section 2.2. Now we are able to state the property of being homomorphic.

Definition 2.2. $\mathcal{E}$ is *homomorphic* for circuits in $C_{\mathcal{E}}$ if $\mathcal{E}$ is correct for $C_{\mathcal{E}}$ and $\mathsf{Decrypt}_{\mathcal{E}}$ can be expressed as a circuit $D_{\mathcal{E}}$ of size polynomial in λ.

Definition 2.3. $\mathcal{E}$ is *fully homomorphic* if it is homomorphic for all circuits.

Thus an encryption scheme is only *partially* homomorphic if it is not homomorphic for all circuits (for example only for addition **or** multiplication).

Gentry separates the development of his cryptosystem into three steps. First, the construction of an encryption scheme that permits evaluation of arbitrary circuits. Second, the construction of an encryption scheme that uses ideal lattices and is almost bootstrappable and third, the explanation of a way to decrease the depth of the decryption circuit, without reducing the depth that the scheme is able to evaluate, to obtain a bootstrappable encryption scheme (which means that it can actually evaluate its own decryption circuit and therefore has a self-referential property). If one is interested in receiving more details, we refer to his paper [Gen09a] or his complete PhD thesis [Gen09b]. Unfortunately, more work needs to be done to obtain a satisfying and efficient implementation of his cryptosystem (if even possible), though we additionally note that there also exist an extended approach of his ideas without the use of ideal lattices. We will now show that even if the difficulties for an efficient implementation were overcome, the relating cracking problem cannot be NP-hard to crack.

To achieve this observation we need to consider a certain characteristic for encryption functions of probabilistic cryptosystems.

Definition 2.4. A probabilistic cryptosystem is called *error-free* if whenever m and m' are two distinct messages, then for every r and public key pk, it holds that

$$E(m, r, pk) \neq E(m', r, pk).$$

The following theorem is one of the main results out of [HMPRS12] and obviously increases the value of the ESY conjecture.

Theorem 2.5 ([HMPRS12]). If the ESY conjecture holds, then any error-free probabilistic PKCS is not NP-hard to crack.

Proof. Given an error-free probabilistic PKCS (E, D, G), let

$$\Pi_{yes} := \left\{ \langle m', pk, c\rangle \;\middle|\; \begin{array}{l} \exists m, r, x, sk \text{ such that } E(m, r, pk) = c, \\ G(x) = \langle pk, sk\rangle \text{ and } m \geq m' \end{array} \right\},$$

$$\Pi_{no} := \left\{ \langle m', pk, c\rangle \;\middle|\; \begin{array}{l} \exists m, r, x, sk \text{ such that } E(m, r, pk) = c, \\ G(x) = \langle pk, sk\rangle \text{ and } m < m' \end{array} \right\}.$$

Since the cryptosystem is error-free we have that $\Pi_{yes} \cap \Pi_{no} = \emptyset$ and both Π_{yes} and Π_{no} are in NP (compare Section 2.1). Thus (Π_{yes}, Π_{no}) is a disjoint NP-pair. Clearly, a separator for this pair can be used to crack the cryptosystem (use oracle queries to the separator to generate the plain-text message m via binary search). Thus if the ESY conjecture holds, then this problem has a separator that is not NP-hard and hence is not NP-hard to crack. $\square$

Because the Goldwasser-Micali cryptosystem [GM84] as well as Gentry's homomorphic cryptosystem [Gen09a] are shown to be error-free, we receive the following corollary.

Corollary 2.6. If the ESY conjecture holds, then the Goldwasser-Micali cryptosystem and Gentry's homomorphic cryptosystem cannot be NP-hard to crack.

Note again that, for now, we are only talking about NP-hardness via Turing reductions. Since the proof of Theorem 2.5 depends on adaptive oracle queries, some weaker reduction types and their corresponding ESY conjectures (see Section 3.2) do not immediately imply the same consequences.

This result significantly increases the value of the ESY conjecture since probabilistic public-key encryption was said to be unaffected by its consequences. Clearly, there also exist many probabilistic cryptosystems which are not error-free such as the Atjai-Dwork cryptosystem [AD97] (on the other hand, if the polynomial-time hierarchy is infinite, this one is also known to not have a NP-hard cracking problem [NS98]). However, in [HMPRS12] it is shown that if the ESY-conjecture holds, then the Atjai-Dwork cryptosystem could be modified to efficiently check for collisions and hence would also become error-free. So if the ESY-conjecture holds, then the Atjai-Dwork cryptosystem is not NP-hard to crack either.

We have now observed that some well-known probabilistic PKCSs do not comprise NP-hard cracking problems if the conjecture holds and hence that any other error-free PKCS is among that compilation. There also exist certain construction patterns for the design of probabilistic cryptosystems based on NP-hard combinatorial problems (see [FK93]). Those are provably NP-hard to crack (but often without practical relevance). Thus if one of these can be shown to be error-free, then the ESY conjecture is wrong.

In the next chapter we will investigate the ESY conjecture for reduction types other than Turing. We remark that the implication for non-probabilistic PKCSs of not being NP-hard for the considered type of reductions remains the same. Consequently, those weaker hardness assumptions may lead to weaker implications for complexity because if $\mathsf{P} \neq \mathsf{NP}$ and even if we have a problem that is, for example, not NP-hard for one-one reductions, it still may be NP-hard for truth-table reductions. Unfortunately, only the ESY conjecture for Turing reductions provides significant implications for the security of PKCSs.

3 ESY-$\mathcal{R}$ conjectures

The ESY conjecture has not yet come up with many possible accesses. Hence one may think of whether the conjecture can be weakened in such a way that some of its significant consequences remain unaffected and maybe some more interconnecting approaches appear. Unsurprisingly, the ESY conjecture can be generalized to allow different kinds of hardness apart from Turing hardness. This separation is useful since some weaker reduction types actually have nearly the same set of consequences as the original conjecture. We note that most of the definitions and results out of the following sections are based on [HMPRS12].

3.1 Preliminaries

For a language L and a string x let $x-1$ define the immediate predecessor of x in standard lexicographic order and $L|x$ define the characteristic sequence of all predecessors of x in standard lexicographic order: $L(\varepsilon)L(0)L(1)...L(x-1)$.

Moreover, let NQP (here it refers to non-deterministic quasi-polynomial and no quantum complexity class) be the class of all languages that can be accepted by non-deterministic Turing machines in quasi-polynomial-time, meaning run times slower than polynomial but not as slow as exponential. Therefore

$$\mathsf{NQP} = \bigcup_{c>0} \mathsf{NTIME}(2^{log(n)^c}).$$

3.1.1 Truth-table reductions and strong non-determinism

Truth-table reductions are an interesting type of reduction between the classical ones by Karp and Cook. In contrast to Turing reductions, where the Turing machine may adaptively ask questions one by one and with respect to previous answers, all oracle queries must be presented at the same time.

Furthermore, we will investigate strong non-determinism which is a restriction of classical non-determinism, where none of the paths of the responding Turing machine is allowed to output a wrong computation. Instead it may output $\perp$ which can be seen as something like "I don't know". The restriction even allows us to define strong non-deterministic computable functions which cannot be done so simply when considering classical non-determinism.

Definition 3.1. A function f is called SNP-computable (SNQP-) if there is a non-deterministic polynomial-time (quasi-polynomial-time) bounded Turing machine M, such that for every x, at least one path of $M(x)$ outputs $f(x)$ and no path outputs a wrong answer. Some paths may output $\perp$.

Clearly, every (deterministic) polynomial-time computable function is SNP-computable and every SNP-computable function is SNQP-computable. Since no path outputs a wrong answer, one benefit of strong non-determinism is that we have the possibility to invert the output of a Turing machine, for example, when considering characteristic functions of languages.

Definition 3.2. A function f is called *polynomially-bounded* if there is a polynomial p such that $|f(x)| \leq |p(x)|$ for all inputs x.

Now we come to the different types of truth-table reductions we will need in the following.

Definition 3.3. A language A is

(1) *polynomial k-truth-table reducible* ($\leq_{ktt}^{\mathsf{P}}$),

(2) *strong, non-deterministic polynomial k-truth-table reducible* ($\leq_{ktt}^{\mathsf{SNP}}$),

(3) *strong, non-deterministic quasi-polynomial k-truth-table reducible* ($\leq_{ktt}^{\mathsf{SNQP}}$),

to a language B, if there is a

(1)+(2) polynomial-time computable function f,

(3) quasi-polynomial-time, polynomially-bounded, computable function f,

and a/an

(1) polynomial-time computable function t,

(2) SNP-computable function t,

(3) SNQP-computable function t,

such that for every x

$$f(x) = \langle q_1, ..., q_k \rangle \text{ and}$$
$$t(x, B(q_1), ..., B(q_k)) = A(x).$$

Function f is called the *truth-table generator* which computes the k queries that represent the oracle questions, whereas function t is called the *truth-table evaluator* that determines the output. The respective result can be summarized in a so-called *truth-table* (example with $k = 2$):

$t(x)$	$B(q_1)$	$B(q_2)$
0	0	0
1	0	1
1	1	0
0	1	1

Note that strong, non-deterministic reductions only use strong non-determinism for the truth-table evaluator and not for the generation of the queries. Moreover, observe that k-truth-table reductions are a special case of classic truth-table reductions ($\leq_{tt}$), where the number of queries may depend on the size of the input.

Definition 3.4. Let D denote a complexity class out of $\{\mathsf{P}, \mathsf{SNP}, \mathsf{SNQP}\}$ and let A and B be two languages such that $A \leq_{tt}^{\mathsf{D}} B$ via $\langle f, t\rangle$. The reduction is called

(1) *bounded* ($\leq_{btt}^{D}$) if there is a constant $k > 0$ such that $A \leq_{ktt}^{\mathsf{D}} B$,

(2) *length-increasing* ($\leq_{ktt,li}^{D}$) if the length of every query is bigger than the length of the input,

(3) *weakly-length-increasing* ($\leq_{ktt,wli}^{D}$) if there is at least one query that is bigger than the length of the input.

Moreover, we say that a query q_i is *relevant* if it holds that

$$t(x, B(q_1), ..., B(q_i), ..., B(q_{k(x)})) \neq t(x, B(q_1), ..., \overline{B(q_i)}, ...B(q_{k(x)})).$$

If a query q_i is relevant then knowing answers to all other queries still does not help us to determine $A(x)$.

3.1.2 Unpredictability

The notion of unpredictability tries to capture the complexity of languages given additional auxiliary information of all predecessors of the input. It turns out that, for example, SAT becomes considerably easy when having information about the satisfiability of every predecessor. In contrary, it is known that EXP contains some languages that are unpredictable for some upper bounds on the runtime of the predictors and it is supposed that NP contains such languages too (see Section 3.4).

Unpredictability is very similar to the notion of genericity [ASFH87, ASNT96] and is equivalent for deterministic computations [BM95].

The conditions in the following definition are in much the same manner as for the SNP-computability of functions.

Definition 3.5. We say that a non-deterministic Turing machine M is *strong* if for every input x, exactly one of the following conditions hold:

(1) at least one path of M accepts x and no path rejects,

(2) at least one path of M rejects x and no path accepts.

Some paths of the machine may output $\perp$.

Definition 3.6. Let M be a strong non-deterministic Turing machine and L be a language. M is a *predictor* for L if for every $x \in L$, M accepts $\langle x, L|x\rangle$ and for every $x \notin L$, M rejects $\langle x, L|x\rangle$.

Given complete information about the characteristic sequence of L regarding x, the Turing machine M is able to determine whether $x \in L$ or $x \notin L$.

Definition 3.7. Let $t(n)$ be any time bound and L and L' be two languages such that $L \subseteq L'$. We say that L is

(1) SNTIME$(t(n))$-*unpredictable*,

(2) SNTIME$(t(n))$-*unpredictable within* L',

if for every strong non-deterministic Turing machine M that predicts L, M runs for more than $t(n)$ time on inputs of the form $\langle x, L|x\rangle$ for

(1) all but finitely many x.

(2) all but finitely many $x \in L'$.

Note that the running time $t(n)$ of the predictor is in terms of the length of the input which is $\langle x, L|x\rangle$ rather than x. Measured in terms of the length of x, it is roughly $t(|x| + 2^{|x|})$. This observation is very important for some of the upcoming proofs.

3.1.3 Secure one-way functions and pseudorandom generators

Intuitively, one-way functions are functions that are easy to compute but hard to invert (which is the task of finding an x on given y such that $f(x) = y$). Though, the question is still open as to whether such ones even exist. None of the functions that are currently applied in certain algorithms for cryptography, personal identification, authentication and so forth have been proven to completely satisfy the conditions for being a secure one-way function (which besides would imply that $\mathsf{P} \neq \mathsf{NP}$).

Furthermore, we need to state some definitions about O-oracle circuits. An *O-oracle circuit C*, denoted by C^O, is nothing more than a Boolean circuit with access to oracle O (implemented as specific oracle-gate).

The whole theory of (secure) one-way functions, pseudorandom generators or actually hard-core functions is quite complex and requires much background knowledge. The following definitions are only the specific parts out of these concepts which we need to state to obtain some results for the ESY conjecture. Hence if one is interested in receiving more context we refer to [GL89, HILL99] or similar papers.

Definition 3.8. A family of functions $\{f_n\} : \Sigma^n \to \Sigma^{l(n)}$ is *one-way, $s(n)$-secure against oracle O* if f_n is uniformly computable in polynomial-time and for every non-uniform circuit C^O of size at most $s(n)$ and for sufficiently large n, it holds that

$$\Pr_{x \in \Sigma^n} \left[C^O(f_n(x)) \in f_n^{-1}(f_n(x)) \right] \leq \tfrac{1}{s(n)}.$$

This means that for sufficiently large n and any x of that size, even for any non-uniform oracle-circuit of size at most $s(n)$ given $f_n(x)$, the probability that the circuit computes an element of the preimage of $f_n(x)$ must not exceed $\frac{1}{s(n)}$. Thus f_n^{-1} is a rather hard function. The property for f_n of being uniformly computable in polynomial time relates to the whole time for determining the function f_n and computing $f_n(x)$ ($x \in \Sigma^n$).

For any function $f : \{0,1\}^n \to \{0,1\}$ and oracle O, the *circuit complexity of f relative to O-oracle*, denoted by C_f^O, is the size of the smallest O-oracle circuit that computes f on every input of size n. If a family of functions $\{f_n\}$ is one-way, $s(n)$-secure against O-oracle circuits, then f_n^{-1} has a circuit complexity relative to O-oracle of at least $s(n)$.

The next definition concerns pseudorandom generators. These are some kind of deterministic procedures computing pseudorandom strings such that no statistical test is able to distinguish between those and the uniform distribution. Thus pseudorandomness is no "real" randomness but provides the benefit of being easy to compute and always mapping the same inputs (often called "seeds") to the exact same output, which makes it useful for, for example, testing or different security protocols.

Definition 3.9. A *pseudorandom generator* is a function G_n out of a family of functions $\{G_n\} : \Sigma^{m(n)} \to \Sigma^n$ such that G_n is uniformly computable in time $2^{\mathcal{O}(m(n))}$ and for every circuit C of size at most $\mathcal{O}(n)$, it holds that

$$|\Pr_{x \in \Sigma^n}[C(x) = 1] - \Pr_{y \in \Sigma^{m(n)}}[C(G_n(y)) = 1]| \leq 1/8.$$

The probability that a circuit evaluates to true on a random input x must be about the same probability (tolerance 12%) that the same circuit evaluates to true on the pseudorandom string computed by the generator G_n.

Finally, given an oracle O, G is said to be *secure against O-oracle* if the above inequality holds for all O-oracle circuits C^O of size at most $\mathcal{O}(n)$, for almost all n.

3.2 Generalization, ESY-m and ESY-tt

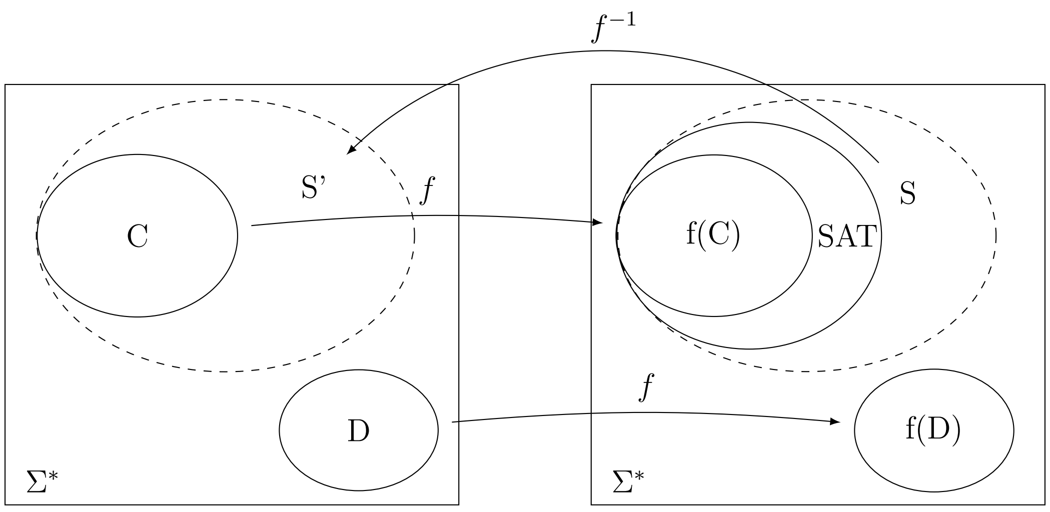
In this section we will concern generalized forms of the ESY conjecture along with that one for Turing reductions. By now, there have already been several investigations on certain other reduction types such as for (polynomial-time) many-one reductions.

ESY-$\mathcal{R}$ conjecture. For every two disjoint languages $A, B \in \mathsf{NP}$, there is a separator C, meaning $A \subseteq C$ and $B \cap C = \emptyset$, that is not *$\mathcal{R}$-hard* for NP.

Although the ESY conjecture deals with arbitrary languages A and B, one of the languages can always be taken to be SAT which eases some of the upcoming proofs.

Observation 3.10. The ESY-$\mathcal{R}$ conjecture is equivalent to the following statement: For every set B in NP that is disjoint from SAT, there is a separator that is not $\mathcal{R}$-hard for NP.

Proof. Let (C, D) be a disjoint NP-pair. Let f be a one-one, length-increasing reduction from C to SAT (every reduction to SAT can be modified to be one-one and length-increasing by simply adding an encoded form of the input to the end of the formula which does not effect the satisfiability). Consider the disjoint pair $(\mathsf{SAT}, f(D))$. Since f is length-increasing, $f(D) \in \mathsf{NP}$ (note that the image of f consists of formulas regardless of whether the inputs are out of C or D). Assume there is a separator S for $(\mathsf{SAT}, f(D))$ that is not $\mathcal{R}$-hard for NP. Then $\mathsf{SAT} \subseteq S$ and $S \cap f(D) = \emptyset$. Let $S' := \{y \mid f(y) \in S\}$.



Observe that $C \subseteq S'$ and $D \cap S' = \emptyset$. Therefore S' is a separator for (C, D). If S' is $\mathcal{R}$-hard for NP, then it implies that S is also $\mathcal{R}$-hard for NP. This is a contradiction. □

If for any two types of reductions, one reduction is stronger than the other, then there is a simple relation between those reductions and the ESY conjectures for those reductions.

Observation 3.11. Let $\mathcal{R}$ and $\mathcal{R}'$ be two types of reductions, such that $\mathcal{R}'$-hardness for NP implies $\mathcal{R}$-hardness for NP. If the ESY-$\mathcal{R}$ conjecture holds then the ESY-$\mathcal{R}'$ conjecture holds.

Proof. Suppose $\mathcal{R}'$-hardness for NP implies $\mathcal{R}$-hardness for NP and that the ESY-$\mathcal{R}'$ conjecture does not hold. Then there exists a disjoint NP-pair (A, B) such that all separators are $\mathcal{R}'$-hard for NP. Because $\mathcal{R}'$-hardness for NP implies $\mathcal{R}$-hardness for NP, every separator of (A, B) must also be $\mathcal{R}$-hard for NP. Thus the ESY-$\mathcal{R}$ conjecture does not hold. □

This observation allows us to collate different types of the conjecture. Let A and B be two languages and $\leq_r^{\mathsf{D}}$ denote the D-time r-reduction between two languages, where D is a complexity class out of $\{\mathsf{P}, \mathsf{SNP}, \mathsf{SNQP}\}$. Since the following implications hold:

$$\begin{tikzcd}
A \leq^{\mathsf{D}}_{m} B \arrow[r, Rightarrow] & A \leq^{\mathsf{D}}_{1tt} B \arrow[r, Rightarrow] & \ldots \arrow[r, Rightarrow] & A \leq^{\mathsf{D}}_{btt} B \arrow[r, Rightarrow] & A \leq^{\mathsf{D}}_{tt} B \arrow[r, Rightarrow] & A \leq^{\mathsf{D}}_{T} B \\
& A \leq^{\mathsf{D}}_{1tt,li} B \arrow[r, Rightarrow] \arrow[u, Rightarrow] & \ldots \arrow[r, Rightarrow] & A \leq^{\mathsf{D}}_{btt,li} B \arrow[r, Rightarrow] \arrow[u, Rightarrow] & A \leq^{\mathsf{D}}_{tt,li} B \arrow[u, Rightarrow] &
\end{tikzcd}$$

, we receive these connections for the ESY-$\leq^{\mathsf{D}}_{r}$ conjectures:

$$\begin{tikzcd}
\text{ESY-}\leq^{\mathsf{D}}_{T} \arrow[r, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{tt} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{btt} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \ldots \arrow[r, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{1tt} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{m} \\
& \text{ESY-}\leq^{\mathsf{D}}_{tt,li} \arrow[r, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{btt,li} \arrow[r, Rightarrow] & \ldots \arrow[r, Rightarrow] & \text{ESY-}\leq^{\mathsf{D}}_{1tt,li} &
\end{tikzcd}$$

Note that for getting from implications within reduction types to the corresponding implications within ESY-$\mathcal{R}$ conjectures (or vice versa), it is always necessary to think the other way around.

Additionally, with

$$A \leq^{\mathsf{P}}_{r} B \;\Rightarrow\; A \leq^{\mathsf{SNP}}_{r} B \;\Rightarrow\; A \leq^{\mathsf{SNQP}}_{r} B,$$

it follows that

$$\text{ESY-}\leq^{\mathsf{SNQP}}_{r} \;\Rightarrow\; \text{ESY-}\leq^{\mathsf{SNP}}_{r} \;\Rightarrow\; \text{ESY-}\leq^{\mathsf{P}}_{r}.$$

These implications are worth mentioning since we need to work with some more complicated forms of the conjecture as, for example, the ESY-$\leq^{\mathsf{SNQP}}_{ktt,wli}$ conjecture which will be considered in Section 3.4.

The next observation is a generalization of the connection between the ESY conjecture, NP and coNP. Recall that $\leq^{\mathsf{P}}_{1}$ stands for polynomial-time one-one reductions.

Observation 3.12. Let $\mathcal{R}$ be a reduction type such that if $A \leq^{\mathsf{P}}_{1,li} B$ for languages A and B, it holds that $A \leq_{\mathcal{R}} B$. Then the ESY-$\mathcal{R}$ conjecture implies $\mathsf{NP} \neq \mathsf{coNP}$.

Proof. Suppose $\mathsf{NP} = \mathsf{coNP}$ and consider the disjoint NP-pair $(\mathsf{SAT}, \overline{\mathsf{SAT}})$. The only separator is SAT which is $\leq^{\mathsf{P}}_{m}$-hard for NP by Cook's Theorem. Since the generic reduction f out of the proof of Cook's Theorem can be modified to simply add an unique encoded form of the input to the resulting formula which does not effect the satisfiability, SAT is also $\leq^{\mathsf{P}}_{1}$-hard for NP. Observe that with this modification the reduction also becomes length-increasing because $|f(x)|$ is at least of size $|x|$. Hence SAT is $\leq^{\mathsf{P}}_{1,li}$-hard for NP and moreover also $\mathcal{R}$-hard for NP for every $\mathcal{R}$-reduction that is implied by the $\leq^{\mathsf{P}}_{1,li}$-reduction. Thus the ESY-$\mathcal{R}$ conjecture does not hold. $\square$

We will now have a look at two results for the ESY-m and the ESY-tt conjecture (if not otherwise mentioned, we are always talking about polynomial-time reductions).

Proposition 3.13 ([GSSZ04]). $\mathsf{NP} \neq \mathsf{coNP}$ if and only if the ESY-m conjecture holds (there is no disjoint NP-pair all of whose separators are $\leq^{\mathsf{P}}_{m}$-hard for NP).

Proof. We will prove the negation and moreover only need to take care of the second direction because with Observation 3.12 it already has been shown that $\mathsf{NP} = \mathsf{coNP}$ implies that the conjecture is wrong for those types of reductions for which SAT is NP-hard. Let $(A, B) \in \mathsf{DisNP}$ and suppose that all of its separators are $\leq^{\mathsf{P}}_{m}$-hard for NP. Because $\overline{B}$ is a separator for (A, B), it holds that $\mathsf{SAT} \leq^{\mathsf{P}}_{m} \overline{B}$ and hence $\overline{\mathsf{SAT}} \leq^{\mathsf{P}}_{m} B$. Therefore $\overline{\mathsf{SAT}} \in \mathsf{NP}$ and finally $\mathsf{NP} = \mathsf{coNP}$. $\square$

Observation 3.14 ([HMPRS12]). The ESY-tt conjecture implies that $\mathsf{NP} \neq \mathsf{UP}$, $\mathsf{NP} \neq \mathsf{coNP}$ and that satisfying assignments of Boolean formulas cannot be computed by single-valued NP-machines.

Proof. Assume $\mathsf{NP} = \mathsf{UP}$. Thus $\mathsf{SAT} \in \mathsf{UP}$ and hence let R be a verifier that witnesses that SAT is in UP (this means that for any x, there is at most one w with $R(x, w) = 1$). Consider the following two disjoint languages in NP (compare with Section 1.2):

$$A := \{\langle x, i\rangle \mid \exists w \text{ such that } R(x, w) = 1 \text{ and the i-th bit of } w \text{ is } 1\},$$

$$B := \{\langle x, i\rangle \mid \exists w \text{ such that } R(x, w) = 1 \text{ and the i-th bit of } w \text{ is } 0\}.$$

Though this observation does not matter here, note that w cannot directly be seen as assignment for the variables since Boolean formulas can have more than one satisfying assignment but w must be unique. Now consider any separator S for (A, B). Let m be the number of Boolean variables in the formula x (observe that $\mathsf{SAT}(x) \Leftrightarrow A(\langle x, i\rangle) \oplus B(\langle x, i\rangle)$ for $1 \leq i \leq m$). Below is a truth-table reduction from SAT to S.

1. Produce the queries $\langle x, 1\rangle$,...,$\langle x, m\rangle$.
2. If $\langle x, i\rangle \in S$ ($\Rightarrow \langle x, i\rangle \notin B$), then $a_i := 1$, else, $a_i := 0$.
3. Accept x if and only if $R(x, a_1a_2...a_m) = 1$.

As described for Turing reductions in the introductory part 1.2, we generate each of the m bits of w by querying (non-adaptively) to S and therewith receive the unique certificate if x is satisfiable. Hence $\mathsf{SAT} \leq_{tt}^{\mathsf{P}} S$ and that results in S being NP-hard. So the ESY-*tt* conjecture does not hold.

A similar argument shows that if the ESY-*tt* conjecture holds, then satisfying assignments of Boolean values cannot be computed by single-valued NP-machines (see [GS88] for more information on NPSV and the affiliation).

Lastly, consider Observation 3.12 for the implication of $\mathsf{NP} \neq \mathsf{coNP}$ (every $\leq_{1,li}^{\mathsf{P}}$-reduction is a $\leq_{tt}^{\mathsf{P}}$-reduction). □

These implications indicate that the ESY-*tt* conjecture could be as hard to prove as the original conjecture. We will take a step further and consider the ESY-*btt* conjecture which turned out to provide better approaches.

3.3 ESY-$\leq_{btt,li}^{\mathsf{D}}$

In this section we will investigate several results for the ESY-$\leq_{btt,li}^{\mathsf{SNP}}$ conjecture (hence for the ESY-$\leq_{btt,li}^{\mathsf{P}}$ conjecture) and afterward relax the length-increasing restriction in Section 3.4 to obtain some stronger statements.

At first we need a result in regard to the existence of unpredictable sets. Recall that a language L is $\mathsf{SNTIME(t(n))}$-unpredictable *within* another language L', if $L \subseteq L'$ and every strong non-deterministic Turing machine that predicts L, runs for more than $t(n)$ time for all but finitely many inputs of form $\langle x, L|x \rangle$ and with $x \in L'$.

Theorem 3.15. For every $r > 0$ there is a set R that is $\mathsf{SNTIME}(2^{log(n)^r})$-unpredictable within $\overline{\mathsf{SAT}}$.

The proof mainly consists of two steps. First, that known results about generic languages imply the existence of $\mathsf{DTIME}(2^{2^{log(n)^r}})$-unpredictable languages and the observation that every $\mathsf{DTIME}(2^{t(n)})$-unpredictable language is $\mathsf{SNTIME}(t(n))$-unpredictable and second, the construction of an unpredictable language within $\overline{\mathsf{SAT}}$ that is $\mathsf{SNTIME}(2^{log(n)^r})$-unpredictable out of an arbitrary language. For a more detailed version we refer to [HMPRS12]

We now state two observations for the queries produced by truth-table reductions under the following precondition.

Precondition. There exists a disjoint NP-pair (A, B) with separator S (then $A \subseteq S$ and $S \cap B = \emptyset$) and there is a language C that $\leq_{ltt}^{\mathsf{SNP}}$-reduces to S via $\langle f, t\rangle$ for an $l \geq 1$ but does not $\leq_{(l-1)tt}^{\mathsf{SNP}}$-reduce to S.

For an input x and the above truth-table generator f, let $f(x) = \langle q_1, ..., q_l\rangle$. We suppose that q_l is the largest query (in standard lexicographic order) and denote it with b_x. Recall that a query is relevant if it is indispensable for the correct computation of $t(x, f(x))$.

Observation 3.16. Under the above precondition there exist infinitely many x such that b_x is relevant.

Proof. Suppose not. Then for all but a finite number of x, we could remove b_x from the list of queries and therewith obtain an $\leq_{(l-1)tt}^{\mathsf{SNP}}$-reduction from C to S. This contradicts that C does not $\leq_{(l-1)tt}^{\mathsf{SNP}}$-reduce to S. □

Let

$$T := \{x \mid b_x \text{ is relevant }\}.$$

Observation 3.17. Under the above precondition there exist infinitely many $x \in T$ such that $b_x \notin A \cup B$.

Proof. Suppose not. Then for all but finitely many $x \in T$, the query b_x is relevant and belongs to $A \cup B$. Hence consider the following reduction $\langle f', t'\rangle$ from C to S: on input x, f' first computes $f(x) = \langle q_1, q_2, ..., q_{l-1}, b_x\rangle$ and then output the queries $\langle q_1, ..., q_{l-1}\rangle$. Let Q_1 and Q_2 be two polynomial-time computable verifier for A and B respectively such that the length of witnesses for positive instances in A and B is bounded by n^r, $r > 0$. We now describe t':

1. Let $b_1 = S(q_1), ..., b_{l-1} = S(q_{l-1})$.
2. Compare $t(x, b_1, ..., b_{l-1}, 0)$ with $t(x, b_1, ..., b_{l-1}, 1)$ to check whether b_x is relevant or not. If b_x is not relevant, then output $t(x, b_1, ...b_{l-1}, 0)$ (or $t(x, b_1, ...b_{l-1}, 1)$).
3. Guess a witness $w \in \Sigma^{n^r}$. If $Q_2(b_x, w)$ holds ($b_x \in B$), then output $t(x, b_1, ..., b_{l-1}, 0)$.
4. If $Q_2(b_x, w)$ does not hold, then guess a witness $u \in \Sigma^{n^r}$. If $Q_1(b_x, u)$ holds ($b_x \in A$) then output $t(x, b_1, ..., b_{l-1}, 1)$, else output $\perp$.

We claim that the above reduction is an $\leq^{\mathsf{SNP}}_{(l-1)tt}$-reduction from C to S, which would contradict that C does not $\leq^{\mathsf{SNP}}_{(l-1)tt}$-reduce to S. Clearly, f' produces only $l-1$ queries. If b_x is not relevant, then $t'(x, b_1, ..., b_{l-1}) = t(x, b_1, ..., b_{l-1}, 0) = t(x, b_1, ..., b_{l-1}, 1) = C(x)$ by our initial assumption that $C \leq^{\mathsf{SNP}}_{ltt}$-reduces to S via $\langle f, t\rangle$. Hence suppose that b_x is relevant. Then for all but finitely many x, b_x is in $A \cup B$ and moreover, we are able to simply ignore the finitely many x for which b_x is not in $A \cup B$ (only finitely many exceptions). If $b_x \in B$, then $b_x \notin S$ (recall that B and S are disjoint) and therefore $t'(x, b_1, ..., b_{l-1}) = t(x, b_1, ..., b_{l-1}, 0) = C(x)$. If $b_x \in A$, then $b_x \in S$. Thus $t'(x, b_1, ..., b_{l-1}) = t(x, b_1, ..., b_{l-1}, 1) = C(x)$. If $b_x \notin A \cup B$ then the output is $\perp$. Thus the reduction is always correct. It remains to show that this is an SNP-reduction. Hence f' must be computable in deterministic polynomial-time and t' has to be SNP-computable. The first can be seen quite easily since f' mainly computes f from the SNP-reduction $\langle f, t\rangle$. To see the second, we take a look at the steps 2-4 and again observe that t is SNP-computable, so we only need to focus on the remainder. If $b_x \in B$, then there is a $w \in \Sigma^{n^r}$ such that $Q_2(b_x, w)$ holds (step 3) and hence there is at least one path which outputs the correct answer. Because B is disjoint from A, for every $u \in \Sigma^{n^r}$, $Q_1(b_x, u)$ does not hold (step 4) and thus no path outputs the wrong answer. A similar argumentation shows that if $b_x \in A$, then $b_x \notin B$ and there is no $w \in \Sigma^{n^r}$ such that $Q_2(b_x, w)$ holds, but there is an $u \in \Sigma^{n^r}$ such that $Q_1(b_x, u)$ holds. Thus C $\leq^{\mathsf{SNP}}_{(l-1)tt}$-reduces to S and this ends the proof. $\square$

The following Theorem is one of the main results out of [HMPRS12].

Theorem 3.18 ([HMPRS12]). If $\mathsf{NP} \neq \mathsf{coNP}$ then the ESY-$\leq^{\mathsf{SNP}}_{btt,li}$ conjecture is true.

Proof. Suppose $\mathsf{NP} \neq \mathsf{coNP}$. Let (A, SAT) be a disjoint NP-pair and Q_1 and Q_2 be two polynomial-time computable verifier for A and SAT respectively. Assume that the length of witnesses for positive instances in A and SAT is bounded by n^r, $r > 0$. Let R be a set that is $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictable within $\overline{\mathsf{SAT}}$ (exists by Theorem 3.15). This provides the separator $S := A \cup R$ for (A, SAT). We will show that there is no $k \geq 0$ such that S is $\leq^{\mathsf{SNP}}_{ktt,li}$-hard for NP and achieve this by induction over k. For the base case consider $k = 0$. This means that the number of queries for the truth-table reduction is zero and that there is an SNP-computable function t such that $t(x) = \mathsf{SAT}(x)$. This implies $\mathsf{NP} = \mathsf{coNP}$ since strong non-determinism allows to complement outputs and therewith contradicts the hypothesis $\mathsf{NP} \neq \mathsf{coNP}$. Thus S cannot be $\leq^{\mathsf{SNP}}_{0tt,li}$-hard.

As the inductive hypothesis, assume that S is not $\leq^{\mathsf{SNP}}_{(l-1)tt,li}$-hard (we use l to distinguish between this induction and the definition of a $\leq_{ktt}$-reduction). Now suppose that there is a length-increasing l-truth-table reduction $\langle f, t\rangle$ such that $\mathsf{SAT} \leq^{\mathsf{SNP}}_{ltt,li} S$ for contradiction. For an input x, let $f(x) = \langle q_1, ..., q_{l-1}, b_x\rangle$, where b_x is assumed to be the largest query. Observe that the proofs of Observation 3.16 and 3.17 still go through if we modify the assumptions for the reductions from C to S out of the precondition to include the property of being length-increasing. This results to the following claim (consider the pair (A, SAT) and $C = \mathsf{SAT}$ to suit the languages in the precondition).

Claim 1. There exist infinitely many $y \notin A \cup \mathsf{SAT}$ with the following property: There exists an x such that $|x| < |y|$, $y = b_x$ and y is relevant.

Note that the property $|x| < |y|$ results out of f being length-increasing. This allows us to build the following predictor for R (to recall: a predictor is a strong non-deterministic Turing machine that for every $y \in R$, accepts $\langle y, R|y\rangle$ and for every $y \notin R$, rejects $\langle y, R|y\rangle$). Let M be a strong non-deterministic algorithm that decides R (no further requirements apart from the existence) and I be the set of all y for which the conditions of Claim 1 hold (we want to come to a result about the runtime on those infinitely many inputs $y \in I$). The predictor works as follows.

1. Input $\langle y, R|y\rangle$.
2. If $y \in A \cup \mathsf{SAT}$, then run $M(y)$ and stop.
3. Search for an x such that $|x| < |y|$ and $b_x = y$. If no such x is found then run $M(y)$ and stop.
4. Let $f(x) = \langle q_1, ..., q_{l-1}, y\rangle$ ($q_l = b_x = y$ by step 3). Compute $b_i = S(q_i)$, where $1 \leq i \leq l-1$ and $S = A \cup R$ ($\Rightarrow (q_i \in S \Leftrightarrow (q_i \in A \vee q_i \in R))$), by
   a) Decide the membership of $q_i \in A$ by running a brute force algorithm for A (with the help of the verifier Q_1).
   b) Decide the membership of $q_i \in R$ by looking at $R|y$.
5. Compare $t(x, b_1, ..., b_{l-1}, 0)$ with $t(x, b_1, ..., b_{l-1}, 1)$ to check whether y is relevant or not. If y is not relevant, then run $M(y)$ and stop.
6. Now it is ensured that $y \in I$. Compute $\mathsf{SAT}(x)$. Find the unique bit b such that $\mathsf{SAT}(x) = t(x, b_1, ..., b_{l-1}, b)$.

7. Accept if and only if b equals 1.

Note that no path of the predictor is allowed to output wrong answers (restriction to be a strong Turing machine) and hence it is not allowed to simply, non-deterministically guess witnesses for A or SAT (see step 2 and 4a).

Claim 2. The above predictor correctly decides R and for infinitely many strings from $\overline{\mathsf{SAT}}$ runs in time $2^{log(n)^{2r}}$.

If $y \notin I$, then the predictor runs $M(y)$ and thus is correct on all such y. Now let $y \in I$. We know that $\mathsf{SAT}(x) = t(x, b_1, ..., b_{l-1}, S(y))$. Since y is relevant, $\mathsf{SAT}(x) \neq t(x, b_1, ..., b_{l-1}, \overline{S(y)})$. Therefore the b out of step 6 equals $S(y)$. Because $y \notin A \cup \mathsf{SAT}$, it holds that $y \in S$ ($b = 1$, step 7) if and only if $y \in R$. Thus the above predictor correctly decides every y in I and hence correctly decides R.

Now we will show that for every $y \in I$ (infinitely many!), the above predictor halts in quasi-polynomial time. Let $|y| = m$ and note that the length of x found in step 3 is at most m (more precise: $m - 1$). Deterministic checking for membership of y in $A \cup \mathsf{SAT}$ takes $\mathcal{O}(2^{m^r})$ time. Since $y = b_x$ is the largest query produced, it holds that $|q_i| \leq m$ for $1 \leq i \leq l - 1$, and since A can be decided in time 2^{n^r}, step 4a takes $\mathcal{O}(2^{m^r})$ time. Because $y > q_i$ for $1 \leq i \leq l - 1$ (again in standard lexicographic order), step 4b takes polynomial time. Computing $\mathsf{SAT}(x)$ takes $\mathcal{O}(2^m)$ time. Lastly, the predictor computes the function t. However, t is SNP-computable which means that its computation time is sub-exponential and negligible. Thus the total time taken is $\mathcal{O}(2^{m^r})$ and hence also $\mathcal{O}(2^{m^{r+1}})$. Since the runtime of the predictor is measured in terms of the length of $\langle y, R|y\rangle$, which is at least 2^m, the total time taken becomes $\mathcal{O}(2^{log(n)^{r+1}})$ for input length n. Hence for every $y \in I$, the predictor runs in time $2^{log(n)^{2r}}$. Because I is an infinite set and by definition is a subset of $\overline{\mathsf{SAT}}$, the claim follows.

The existence of a predictor for R with runtime $2^{log(n)^{2r}}$ contradicts the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictability within $\overline{\mathsf{SAT}}$ and thus the definition of R. Because Claim 1 and 2 result out of the assumption that SAT $\leq^{\mathsf{SNP}}_{ltt,li}$-reduces to S, this cannot hold. Therefore we have shown that S is not $\leq^{\mathsf{SNP}}_{ltt,li}$-hard for NP. This completes the induction step. Thus there is no $k \geq 0$ such that S is $\leq^{\mathsf{SNP}}_{ktt,li}$-hard for NP and hence S cannot be $\leq^{\mathsf{SNP}}_{btt,li}$-hard for NP. This completes the proof of the theorem. □

The main result of this section is a corollary of Theorem 3.18.

Theorem 3.19 ([HMPRS12]). $\mathsf{NP} \neq \mathsf{coNP}$ if and only if the ESY-$\leq^{\mathsf{P}}_{btt,li}$ conjecture holds.

Proof. Suppose $\mathsf{NP} \neq \mathsf{coNP}$. Then the ESY-$\leq^{\mathsf{SNP}}_{btt,li}$ conjecture is true by Theorem 3.18. Since ESY-$\leq^{\mathsf{SNP}}_{btt,li}$ implies ESY-$\leq^{\mathsf{P}}_{btt,li}$ (see the part behind Observation 3.11), the first direction holds. The second direction directly follows by Observation 3.12 (every $\leq^{\mathsf{P}}_{1,li}$-reduction is a $\leq^{\mathsf{P}}_{btt,li}$-reduction). □

Thus the ESY-$\leq^{\mathsf{P}}_{btt,li}$ conjecture is equivalent to $\mathsf{NP} \neq \mathsf{coNP}$, as it holds for the ESY-$\leq^{\mathsf{P}}_{m}$ conjecture. Unfortunately, we have no hypotheses yet that imply one of those. Though, we actually have two for the ESY-$\leq^{\mathsf{P}}_{btt}$ conjecture, as we will see in the next section.

3.4 ESY-$\leq^{\mathsf{P}}_{btt}$

In this section we relax the length-increasing requirement and consider general bounded truth-table reductions. We show two interesting relations between the existence of certain unpredictable sets, secure one-way functions against O-oracle circuits and the ESY-$\leq^{\mathsf{P}}_{btt}$ conjecture.

Theorem 3.20 ([HMPRS12]). If NP has an $\mathsf{SNTIME}(n^2)$-unpredictable set, then the ESY-$\leq^{\mathsf{P}}_{btt}$ conjecture holds.

Recall that the runtime of the predictor refers to the length of the input which is $\langle x, L|x\rangle$ instead of x (approximately $|x| + 2^{|x|}$ in terms of the length of x).

Some parts of the proof are very similar to the proof of Theorem 3.18. Hence we will focus on the differences and shorten some redundant parts.

Proof. Let U be an $\mathsf{SNTIME}(n^2)$-unpredictable set in NP and (A, SAT) be a disjoint NP-pair. As before, let Q_1 and Q_2 be polynomial-time computable verifier for A and SAT and again assume that the lengths of witnesses is bounded by n^r. Since NP has an $\mathsf{SNTIME}(n^2)$-unpredictable set it follows that NP has an $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictable set which we call G (this observation results out of a similar technique as used in [ASTZ97] called "reverse padding trick" and will not further be described here). Now let $G_0 := \{x \mid x0 \in G\}$ and $G_1 := \{x \mid x1 \in G\}$ $(x \in \{0,1\}^*)$ and observe that $G_0, G_1 \in \mathsf{NP}$.

Claim 1. Since G is $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictable, neither G_0 nor G_1 is in $\mathsf{NP}\cap\mathsf{coNP}$.

Suppose G_0 is in $\mathsf{NP} \cap \mathsf{coNP}$. Then we are able to build the following predictor for G which then contradicts the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictability. Let M be a strong non-deterministic algorithm that decides G (no further requirements apart from the existence) and R_1 and R_2 be two polynomial-time computable verifier for G_0 and $\overline{G_0}$ respectively. Assume that the runtime of R_1 and R_2 is bounded by n^k for $k > 0$ and that the lengths of witnesses are bounded by n^s for $s > 0$.

1. Input $\langle y, G|y\rangle$.
2. If $y = x1$ or $y = \varepsilon$, then run $M(y)$ and stop.
3. Otherwise it holds that $y = x0$. Guess a witness $w \in \Sigma^{n^s}$. If $R_1(x, w)$ holds ($\Rightarrow x \in G_0 \Rightarrow y \in G$) then accept $\{y, G|y\}$.
4. If $R_1(x, w)$ does not hold, then guess a witness $u \in \Sigma^{n^s}$. If $R_2(x, u)$ holds ($\Rightarrow x \in \overline{G_0} \Rightarrow y \notin G$) then reject $\{y, G|y\}$, else output $\perp$.

Clearly, if $y = x1$ or $y = \varepsilon$, then the predictor correctly decides G by computing $M(y)$. If $y = x0$, then it holds that $y \in G \Leftrightarrow x \in G_0$. Either there exists a witness w such that $R_1(x, w)$ holds or there exists a witness u such that $R_2(x, u)$ holds. Hence at least one path outputs the correct answer and no path outputs a wrong answer (some paths outputs $\perp$). Since the runtime of R_1 and R_2 is bounded by n^k, the runtime of the predictor is $\mathcal{O}(n^k)$ for infinitely many inputs (there exist infinitely many y of form $y = x0$). Because the runtime is clearly within $2^{log(n)^{2r}}$ for infinitely many inputs and the predictor is correct on all inputs, this contradicts the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictability of G. The same argument, but the other way around, shows that G_1 cannot be in $\mathsf{NP} \cap \mathsf{coNP}$.

We will now show that with the above result and the existence of the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictable set G, we are able to construct a separator for (A, SAT) that is not $\leq^{\mathsf{P}}_{btt}$-hard for NP.

Let $G' := G_1 \setminus \mathsf{SAT}$ and S be the separator $A \cup G'$. Then $A \subseteq S$ and $S \cap \mathsf{SAT} = \emptyset$. We claim that S is not $\leq^{\mathsf{P}}_{btt}$-hard for NP and again show this by induction. Moreover, we proof a stronger statement. Since G_0 is in NP, the claim follows if for every $k \geq 0$, G_0 does not $\leq^{\mathsf{SNP}}_{ktt}$-reduce to S.

The base case is when $k = 0$ and the number of queries is zero. This means that there is an SNP-computable function t such that $t(x) = G_0(x)$. Hence $G_0 \in \mathsf{NP} \cap \mathsf{coNP}$ since strong non-determinism allows to complement outputs. This is a contradiction because G_0 was proven to be not in $\mathsf{NP} \cap \mathsf{coNP}$.

As the inductive hypothesis, assume that G_0 does not $\leq^{\mathsf{SNP}}_{(l-1)tt}$-reduce to S. We will now prove that G_0 does not $\leq^{\mathsf{SNP}}_{ltt}$-reduce to S. Hence assume that G_0 actually does $\leq^{\mathsf{SNP}}_{ltt}$-reduce to S and let $\langle f, t\rangle$ be one such reduction from G_0 to S. Given x, let Q_x denote the set of queries produced by $f(x)$ and let b_x be the largest query, as it was in the proof of Theorem 3.18.

Claim 2. For all but finitely many x, it holds that $x0 < b_x1$ in standard lexicographic order.

Suppose the opposite and that there exist infinitely many strings x such that $x0 > b_x1$. Since b_x is the largest query, it holds for infinitely many x that $x0 > q_i1$ and $x > q_i$, for every $q_i \in Q_x$. Again, this enables us to build the following predictor for G. As before, let M be a strong non-deterministic algorithm that decides G.

1. Input $\langle y, G|y\rangle$.
2. If $y = x1$ or $y = \varepsilon$, run $M(y)$ and stop.
3. Otherwise it holds that $y = x0$. Use f to compute $Q_x = \{q_1, ..., q_l\}$, where $b_x = q_l$. If $x \leq b_x$, run $M(y)$ and stop.
4. Now it holds that $y > b_x$. Determine $S(q_i)$, where $1 \leq i \leq l$ and $S = A \cup G'$, by
    a) Decide the membership of $q_i \in A$ by running a brute force algorithm for A (with the help of the verifier Q_1).
    b) Decide the membership of $q_i \in G'$ (recall: $G' = G_1 \setminus \mathsf{SAT}$) by deciding the membership in G_1 and SAT via looking at $\langle y, G|y\rangle$ ($q_i1 < y$ and $q_i \in G_1 \Leftrightarrow q_i1 \in G$) and computing $\mathsf{SAT}(x)$, since it holds that $q_i \in G' \Leftrightarrow (q_i \in G_1 \wedge q_i \notin \mathsf{SAT})$.
5. Determine $G_0(x)$ by computing $t(S(q_1), ..., S(q_l))$. This will tell whether $y \in G$ or not because $y = x0$ and $x \in G_0 \Leftrightarrow x0 \in G$.

It is easy to see that the above predictor correctly decides G. Suppose $|y| = m$. Since t is SNP-computable, the total time taken for step 3 to 5 is at most $2^{m^{2r}}$. Because there exist infinitely many y of the form $y = x0$ and the runtime is measured in terms of the length of $\langle y, G|y\rangle := n$, the predictor runs in time $2^{log(n)^{2r}}$ for infinitely many inputs. This contradicts the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictability of G.

As before, we consider whether a query is relevant or not. Observation 3.16 and 3.17 and Claim 2 result in the following claim (for the precondition: consider (A, SAT) as disjoint NP-pair and let C be G_0).

Claim 3. There exist infinitely many $y \notin A \cup \mathsf{SAT}$ with the following property: There exists an x with $x < y$ such that $b_x = y$ and y is relevant.

Note that the property $x < y$ follows from the result $x0 < b_x 1$ out of Claim 2 and not because of f being length-increasing (which it is not here) as it was the case in Claim 1 in the proof of Theorem 3.18 (there we received $|x| < |y|$). We will now describe a predictor for G. As before, let M be a strong non-deterministic algorithm that decides G and let I be the set of all y for which the conditions of Claim 3 hold (again, we want to come to a result about the runtime on this infinitely many $y \in I$). The predictor works as follows (compare with the predictor in the proof of Theorem 3.18:

1. Input $\langle z, G|z\rangle$.
2. Let $z = yb$. If $b = 0$ or $y \in A \cup \mathsf{SAT}$, run $M(z)$ and stop.
3. Search for an x such that $x < y$ and $b_x = y$. If there is no such x, then run $M(z)$ and stop.
4. Now it holds that $z = y1$ and $y \notin A \cup \mathsf{SAT}$. Let $f(x) = \langle q_1, ...q_{l-1}, y\rangle$. Compute $b_i = S(q_i)$, where $1 \leq i \leq l-1$ and $S = A \cup G'$, by
   a) Decide the membership of $q_i \in A$ by running a brute force algorithm for A (with the help of the verifier Q_1).
   b) Decide the membership of $q_i \in G'$ (recall: $G' = G_1 \setminus \mathsf{SAT}$) by deciding the membership in G_1 and SAT via looking at $\langle z, G|z\rangle$ ($z > y \Rightarrow z > q_i$ since $y > q_i$, and $q_i \in G_1 \Leftrightarrow q_i 1 \in G$) and computing $\mathsf{SAT}(q_i)$, since it holds that $q_i \in G' \Leftrightarrow (q_i \in G_1 \wedge q_i \notin \mathsf{SAT})$.

5. Compare $t(x, b_1, ..., b_{l-1}, 0)$ with $t(x, b_1, ..., b_{l-1}, 1)$ to check whether y is relevant or not by comparing . If y is not relevant, then run $M(z)$ and stop.
6. Now it is ensured that $y \in I$. Compute $\mathsf{SAT}(x)$. Find the unique bit b such that $\mathsf{SAT}(x) = t(x, b_1, ..., b_{l-1}, b)$.
7. Accept if and only if b equals 1.

Claim 4. The above predictor correctly decides G and for infinitely many strings runs in time $2^{log(n)^{2r}}$.

The proof is very similar to the proof of Claim 2 out of the proof of Theorem 3.18, so we presume to skip the statement of nearly the same explanation. The main difference in the runtime of the predictor is the computation of $\mathsf{SAT}(q_i)$ for $1 \leq i \leq l-1$ in step 4b, which takes $\mathcal{O}(2^m)$ time for $|y| = m$ (note that $|q_i| \leq m$) and therefore does not significantly increase the overall runtime.

The existence of a predictor for G with runtime $2^{log(n)^{2r}}$ contradicts the $\mathsf{SNTIME}(2^{log(n)^{2r}})$-unpredictability and thus the definition of G. Because Claim 1-4 result out of the assumption that $G_0 \leq_{ltt}^{\mathsf{SNP}}$-reduces to S, this cannot hold. Therefore we have shown that S is not $\leq_{ltt}^{\mathsf{SNP}}$-hard for NP. This completes the induction step. Thus there is no $k \geq 0$ such that S is $\leq_{ktt}^{\mathsf{SNP}}$-hard for NP and hence S cannot be $\leq_{btt}^{\mathsf{SNP}}$-hard for NP. Clearly, this implies that S cannot be $\leq_{btt}^{\mathsf{P}}$-hard for NP. This completes the proof of the theorem. □

We will now come to the second set of hypotheses that imply the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture. The following part requires the notions out of Section 3.1.3.

Observation 3.21. Suppose that every set A that is $\leq_{ktt}^{\mathsf{SNP}}$-hard for NP is $\leq_{ktt,wli}^{\mathsf{SNP}}$-hard for NP. Then $\mathsf{NP} \neq \mathsf{coNP}$ if and only if the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture holds.

Proof. The second direction of the bi-implication unconditionally holds by Observation 3.12. Thus we only need to focus on the first. With Theorem 3.18 we have shown that $\mathsf{NP} \neq \mathsf{coNP}$ implies the truth of the ESY-$\leq_{btt,li}^{\mathsf{SNP}}$ conjecture and therefore also of the ESY-$\leq_{btt,li}^{\mathsf{P}}$ conjecture (Observation 3.11). Hence we consider the proof of Theorem 3.18 and try to remove the necessity of the reduction to be length-increasing. To achieve this, we need to make use of the above assumption that every set A that is $\leq_{ktt}^{\mathsf{SNP}}$-hard for NP is $\leq_{ktt,wli}^{\mathsf{SNP}}$-hard for NP. The only parts of the proof of Theorem 3.18 where the length-increasing property was required were in Claim 1 and in the runtime analysis of Claim 2. Thus assume that there is an $\leq_{ltt}^{\mathsf{SNP}}$-reduction and no $\leq_{(l-1)tt}^{\mathsf{SNP}}$-reduction from SAT to S (hence no $\leq_{(l-1)tt,wli}^{\mathsf{SNP}}$-reduction) in place of the length-increasing reductions out of the proof of Theorem 3.18. Then by our assumption, there also is an $\leq_{ltt,wli}^{\mathsf{SNP}}$-reduction $\langle f, t\rangle$ from SAT to S. Observe that the proof of Observation 3.16 still goes through, even if we consider the above weakly length-increasing reduction for the pre-condition (though, if we remove the largest query, then the resulting reduction may not be weakly length-increasing anymore, but we additionally have that there does not exist any $\leq_{(l-1)tt}^{\mathsf{SNP}}$-reduction). Moreover, since $\langle f, t\rangle$ is weakly length-increasing, it holds for every x, that there is at least one query that is larger than x and in particular, the biggest query $y = b_x$ must be larger than x. Thus Claim 1 still remains and enables us to build the respective predictor for R. Observe, that because $y = b_x$ is the largest query produced and because it is also larger than x, it again constrains the runtime of the predictor in the analysis of Claim 2. Hence with these changes, based on the above assumption, the proof of Theorem 3.18 still holds, implies the truth of the ESY-$\leq_{btt}^{\mathsf{SNP}}$ conjecture and furthermore the truth of the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture. □

The next observation is quite similar, though it deals with quasi-polynomial reductions and NQP instead of NP. Observe that the proof of Observation 3.21 still goes through if we replace the SNP-reductions with SNQP-reductions and demand for $\mathsf{NQP} \neq \mathsf{coNP}$ instead of $\mathsf{NP} \neq \mathsf{coNP}$.

Observation 3.22. Suppose that every set A that is $\leq_{ktt}^{\mathsf{SNQP}}$-hard for NP is $\leq_{ktt,wli}^{\mathsf{SNQP}}$-hard for NP and that $\mathsf{NQP} \neq \mathsf{coNP}$. Then the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture holds.

We now state two results where each of them implies one of the hypotheses for the above Observation 3.22. Afterward, we will summarize the hypotheses for those results in Theorem 3.28 and finally obtain a further indication for the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture.

Observation 3.23. Suppose that there exists an $\varepsilon > 0$ and a one-one, one-way function that is $2^{n^{\varepsilon}}$-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits. Then $\mathsf{NQP} \neq \mathsf{coNP}$.

Proof. Note that every one-way function f can be trivially inverted by a polynomial-size circuit with an NP-oracle. To see this, consider the oracle

$$O := \{\langle x, y\rangle \mid \exists x' : f(xx') = y\}$$

to build up x bit by bit. Now suppose $\mathsf{NQP} = \mathsf{coNP}$ (hence $\mathsf{NP} = \mathsf{coNP}$ and also $\mathsf{NQP} = \mathsf{coNQP}$ which finally results to $\mathsf{NQP} = \mathsf{coNP} = \mathsf{NP} = \mathsf{coNQP}$). Then every one-way function can be trivially inverted by a polynomial-size circuit with an $\mathsf{NQP} \cap \mathsf{coNQP}$-oracle ($\mathsf{NQP} \cap \mathsf{coNQP} = \mathsf{NP}$). A simple padding argument shows that one can convert such circuits into quasi polynomial-size circuits with $\mathsf{NP} \cap \mathsf{coNP}$-oracles. This could work as followed:

$$x \longmapsto x\$...\$ \ \text{ s.t. } \ |x\$...\$| = 2^{log(|x|)^k} \text{ (for a } k \geq 1),$$
$$O' := \{x\$...\$ \mid x \in O\}.$$

Observe that if O is an $\mathsf{NQP} \cap \mathsf{coNQP}$-oracle, then it holds that O' is an $\mathsf{NP} \cap \mathsf{coNP}$-oracle because the size of the inputs is quasi-polynomial larger than the size of the inputs to O. The respective circuit grows by the same factor since we need quasi-polynomial many additional gates for the quasi-polynomial many \$ as input for the oracle-gate.

Thus there is no $\varepsilon > 0$ such that there is a one-one, one-way function that is 2^{n^ε}-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits. □

The proof of the upcoming second result heavily relies on the ideas of the papers of Agrawal [Agr02] and Agrawal and Watanabe [AW09] who showed, among several other results, that if one-one, one-way functions exist, then all NP-complete languages are complete via non-uniform, one-one and length-increasing reductions. Before we come to this, we need to set up one more definition.

Henceforth, we will view an $\leq_{ktt}^{\mathsf{SNP}}$-reduction $\langle f, t\rangle$ as a function from Σ^* to $(\Sigma^*)^k \times T_k$, where T_k is the set of all truth-tables for k variables. Then if $A \leq_{ktt}^{\mathsf{SNP}} B$, then $x \overset{f,t}{\mapsto} \langle q_1, ...q_k, t_k\rangle$, where $t_k \in T_k$. Denote this function by $F_{f,t}$.

Definition 3.24. A truth-table-reduction $\langle f, t\rangle$ is *α-sparse* on a set $S \subseteq \Sigma^n$, $0 < \alpha < 1$, if for every x_0 in S,

$$|\{x \in \Sigma^n \mid F_{f,t}(x) = F_{f,t}(x_0)\}| \leq \tfrac{2^n}{2^{n^\alpha}}.$$

This means that the number of elements of length n that result in the same queries and truth-table as any element out of S (which also have length n) must not exceed $\frac{2^n}{2^{n^\alpha}}$.

Now we state the theorem that implies the other hypothesis for Observation 3.22.

Theorem 3.25. Suppose that there exists an $\varepsilon > 0$ and a one-one, one-way function that is 2^{n^ε}-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits. Then every language that is $\leq_{ktt}^{\mathsf{SNP}}$-hard for NP is also $\leq_{ktt,wli}^{\mathsf{SNQP}}$-hard for NP.

Proof. Let $\{f_o\}_n : \Sigma^n \to \Sigma^{l(n)}$ be a one-one, one-way function family that is 2^{n^ε}-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits (for an $\varepsilon > 0$) and L be an $\leq_{ktt}^{\mathsf{SNP}}$-complete language for NP (thus $L \in \mathsf{NP}$). We will first come to a result that under those assumptions, it holds that for every language A in NP, there is also a sparse $\leq_{ktt}^{\mathsf{SNP}}$-reduction to L. Afterward, we claim that from every language A in NP, there is also a certain *randomized* $\leq_{ktt,wli}^{\mathsf{SNP}}$-reduction to L and finally remove the randomness.

Claim 1. There exists a $\gamma > 0$ such that for every language A in NP, there is an $\leq_{ktt}^{\mathsf{SNP}}$-reduction from A to L that is γ-sparse on $A^n := \{w \in A \mid |w| = n\}$ for every $n \geq 0$.

The proof of this claim requires some more background knowledge of so called *hard-core functions* [GL89, HILL99]. Hence we presume to simply state a few essential but already known properties and results within.

Goldreich and Levin showed that every one-way function f, padded to the form $f'(x, y) = (f(x), y)$ with $|x| = |y|$ has by itself a so called *hard-core predicate* of the same security, often denoted by $b(x)$, that can be easily computed on input x but cannot be efficiently guessed only with the information of $f(x)$. To cite from their paper: "Intuitively, the hard-core predicate “concentrates” the onewayness of the function in a strong sense“.

If we suppose, for the sake of simplicity, that the language A is only defined on even length strings (otherwise we could work with $A' = A \cdot A$), then define the Goldreich-Levin hard-core function [GL89] using the one-way function f_o out of the hypothesis:

$$f_{gl}(xy) := \{f_o(x), y, x \oplus y\},$$

where xy is a $2n$ bit string and $x \oplus y$ the above mentioned hard-core predicate. Note that $x \oplus y$ refers to the inner product modulo 2 for Boolean vectors and results in one single bit.

The following observation on hard-core functions is a well-known result[GL89, HILL99].

Claim 2. There exists a γ $(< \varepsilon)$ such that for every sufficiently large n, for every oracle O in $\mathsf{NP} \cap \mathsf{coNP}$, and for every O-oracle circuit D of size at most 2^{n^γ},

$$\left| \Pr_{x,y \in \Sigma^n} [D(f_o(x), y, x \oplus y) = 1] - \Pr_{x,y \in \Sigma^n, b \in \{0,1\}} [D(f_o(x), y, b) = 1] \right| \leq \tfrac{1}{2^{(3n)^\gamma}},$$

where b is called the *hard-core bit*.

Now define

$$B := \{f_{gl}(w) \mid w \in A\}.$$

Observe that since f_o is one-one, it holds that f_{gl} is one-one and thus $B \in \mathsf{NP}$ (recall that $A \in \mathsf{NP}$). Because L is an $\leq_{ktt}^{\mathsf{SNP}}$-complete language for NP, there is an $\leq_{ktt}^{\mathsf{SNP}}$-reduction $\langle g, t\rangle$ from B to L. Hence $\langle f, t\rangle$ is a $\leq_{ktt}^{\mathsf{SNP}}$-reduction from A to L if we set $f = g \circ f_{gl}$. Note that there is an oracle $O \in \mathsf{NP} \cap \mathsf{coNP}$ such that f can be computed in polynomial-time using queries to O.

$$\begin{tikzcd}
\mathrm{A} \arrow[r, "f_{gl}"] \arrow[rr, bend right, "\langle g \circ f_{gl}, t\rangle", "ktt\text{-reduction}"'] & \mathrm{B} \arrow[r, "\langle g, t\rangle", "ktt\text{-reduction}"'] & \mathrm{L}
\end{tikzcd}$$

Now we will show that $\langle f, t\rangle$ is γ-sparse on A^{2n} and therewith complete the proof of Claim 1. Suppose that $\langle f, t\rangle$ is not γ-sparse on A^{2n}. Then by the definition of an α-sparse reduction there exists a string $w_0 \in A^{2n}$, such that the size of the following set S is bounded below as follows:

$$|S := \{w \in \Sigma^{2n} \mid F_{f,t}(w) = F_{f,t}(w_0)\}\,| \geq \tfrac{2^{2n}}{2^{(2n)^\gamma}}.$$

Observe that since f_{gl} is one-one, it holds that $|S| = |f_{gl}(S)|$ and $F_{g,t}(f_{gl}(S)) = \{F_{f,t}(w_0)\}$ (always the same queries and truth-table). Hence define an O-oracle circuit D that on input string z of length $l(n) + n + 1$ (this is exactly the size of every string in the image of f_{gl}) behaves as follows:

If $F_{g,t}(z) = F_{f,t}(w_0)$ then accept, otherwise reject.

This oracle circuit contradicts the upper bound of the allowed difference between the probabilities out of Claim 2 as we will ascertain now:

By the definition of S and the observation we made right before the definition of D, it holds that the circuit D accepts a string $z \in \Sigma^{l(n)+n+1}$ if and only if $z \in f_{gl}(S)$. This results to the following probability:

$$\begin{aligned} p_n := \Pr_{x,y\in\Sigma^n}[D(f_o(x), y, x\oplus y) = 1] &= \Pr_{x,y\in\Sigma^n}[(f_o(x), y, x\oplus y) \in f_{gl}(S)] \\ &= \tfrac{|S|}{2^{2n}} \\ &\geq \tfrac{\frac{2^{2n}}{2^{(2n)^\gamma}}}{2^{2n}} \\ &= \tfrac{1}{2^{(2n)^\gamma}} \end{aligned}$$

Note that 2^{2n} is the total number of different binary strings of length $2n$. Thus $p_n \geq \frac{1}{2^{(2n)^\gamma}}$. Furthermore, we have that:

$$\begin{aligned} \Pr_{x,y\in\Sigma^n, b\in\{0,1\}}[D(f_o(x), y, b) = 1] &= \Pr[b = x\oplus y] \cdot \Pr[D(f_o(x), y, b) = 1 \mid b = x\oplus y] \\ &+ \Pr[b = \overline{x\oplus y}] \cdot \Pr[D(f_o(x), y, b) = 1 \mid b = \overline{x\oplus y}] \\ &= 1/2 \cdot \Pr[D(f_o(x), y, x\oplus y) = 1] \\ &+ 1/2 \cdot \Pr[D(f_o(x), y, \overline{x\oplus y}) = 1] \\ &= 1/2 \cdot p_n \\ &+ 1/2 \cdot 0 \\ &= \tfrac{1}{2}p_n \end{aligned}$$

Observe that for every x and y, the tuple $\langle f_o(x), y, \overline{x\oplus y}\rangle$ does not belong to $f_{gl}(\Sigma^{2n})$ (only $\langle f_o(x), y, x\oplus y\rangle$) and hence does not belong to $f_{gl}(S)$. Furthermore, it holds that $F_{g,t}(\langle f_o(x), y, \overline{x\oplus y}\rangle) \neq F_{f,t}(w_0)$ (same observation as before but with $\langle f_o(x), y, \overline{x\oplus y}\rangle$ instead of $f_{gl}(S)$. So D does not accept $\langle f_o(x), y, \overline{x\oplus y}\rangle$. This results to:

$$\begin{aligned} \left| \Pr_{x,y\in\Sigma^n}[D(f_o(x), y, x\oplus y) = 1] - \Pr_{x,y\in\Sigma^n, b\in\{0,1\}}[D(f_o(x), y, b) = 1] \right| &= \tfrac{1}{2}p_n \\ &\geq \tfrac{1}{2}\tfrac{1}{2^{(2n)^\gamma}} \end{aligned}$$

Because of $\frac{1}{2}\frac{1}{2^{(2n)^\gamma}} \geq \frac{1}{2^{(3n)^\gamma}}$ and since D is a polynomial-size circuit with access to an $\mathsf{NP} \cap \mathsf{coNP}$-oracle O, this contradicts the hard-core bit of Claim 2. Thus $\langle f, t\rangle$ is a $\leq_{ktt}^{\mathsf{SNP}}$-reduction that actually is γ-sparse on A^{2n}. This completes the proof of Claim 1.

We will now show that for any set A in NP there is also a certain randomized, weakly length-increasing $\leq_{ktt}^{\mathsf{SNP}}$-reduction from A to L, which is an $\leq_{ktt}^{\mathsf{SNP}}$-reduction $\langle h, t\rangle$ that additionally depends on a random string $r \in \Sigma^*$ and only for a certain probability needs to generate a query that is larger then the input:

(1) For every x and r, if $F_{h,t}(x,r) = \langle q_1, ..., q_k, t_k\rangle$, then $x \in A \Leftrightarrow t_k(L(q_1)...L(q_k)) = 1$,

(2) For every $x \in A$ of length n it holds that

$$\Pr_{r\in\Sigma^*}[\exists q_i \in Q_{x,r} \text{ such that } |q_i| > n] \geq 3/4,$$

where $Q_{x,r}$ is the set of all queries produced by $h(x,r)$.

Claim 3. Let A be any language in NP. Then there is a randomized, weakly length-increasing $\leq_{ktt}^{\mathsf{SNP}}$-reduction from A to L if the length of the random string r is set to be $\left\lfloor \frac{20kn}{\gamma} \right\rfloor$ (γ of out Claim 1).

By Claim 1 there is an $\leq_{ktt}^{\mathsf{SNP}}$-reduction $\langle h', t\rangle$ from A to L that is γ-sparse on A^n for every $n \geq 0$. We construct a $\leq_{ktt}^{\mathsf{SNP}}$-reduction $\langle h, t\rangle$ from $A \times \Sigma^*$ to L by simply adding a random string $r \in \Sigma^*$ to the input. Observe that this reduction is γ-sparse on $S := A^n \times \Sigma^m$ for every $m, n \geq 0$. We will now show that because of this, the reduction is also randomized, weakly length-increasing if we set $m = \left\lfloor \frac{20kn}{\gamma} \right\rfloor$ and $r \in \Sigma^m$. Let R be the set of all tuples $\langle q_1, ...q_k, t_k\rangle$, where each q_i is of length at most n and t_k is a truth-table over k variables. Thus

$$R := \{\langle q_1, ...q_k, t_k\rangle \mid |q_i| \leq n \text{ for } 1 \leq i \leq k, t_k \in T_k\} \text{ and}$$

$$\overline{R} := \{\langle q_1, ...q_k, t_k\rangle \mid \exists q_i \text{ such that } |q_i| > n, t_k \in T_k\}.$$

We actually want to come to a result about the number of $r \in \Sigma^m$ that are mapped to a tuple that belongs to $\overline{R}$. The number of different combinations of queries where the size of each query is smaller than n is $2^{(n+1)k}$ (including empty and identical queries) and the number of different truth-tables is 2^{2^k} (we have 2^k different combinations for $S(q_1), ..., S(q_k)$ and either 0 or 1 as output of a truth-table evaluator on each of those combinations). So it holds that

$$|R| = 2^{(n+1)k} \cdot 2^{2^k}.$$

For a string x of size n, define

$$C_x := \{r \in \Sigma^m \mid F_{h,t}(x,r) \in R\} \text{ and}$$

$$\overline{C_x} := \{r \in \Sigma^m \mid F_{h,t}(x,r) \in R\}.$$

Note that every tuple $(x,r) \in \Sigma^n \times \Sigma^m$ can be trivially encoded as a string of length $2(n+m)$ and that there exist 2^m different $r \in \Sigma^m$ (thus $|C_x| + |\overline{C_x}| = 2^m$). Since $\langle h, t\rangle$ is γ-sparse on S, it holds for every $(x_0, r_0) \in A^n \times \Sigma^m$, that

$$|G(x_0, r_0) := \{(x, r) \in \Sigma^n \times \Sigma^m \mid F_{h,t}(x, r) = F_{h,t}(x_0, r_0)\}\,| \leq \tfrac{2^{2n+2m}}{2^{(2n+2m)^\gamma}}.$$

Now observe that the size of C_x is at most as large as the number of different tuples out of the image of $F_{h,t}$, for which the size of each query is smaller than n (which is the size of R), times the number of elements (x, r) that could be mapped to the same tuple of $F_{h,t}$ (which is the largest possible size of G). We obtain

$$|C_x| \leq |R| \cdot \max\left(\{|G(x_0, r_0)| : (x_0, r_0) \in S\}\right) \leq 2^{(n+1)k} \cdot 2^{2^k} \cdot \tfrac{2^{2n+2m}}{2^{(2n+2m)^\gamma}}.$$

Lemma 3.26 ([HMPRS12]). If m is set to be $\left\lfloor \frac{20kn}{\gamma} \right\rfloor$, then the upper bound for the size of C_x is at most $\frac{1}{4}2^m$.

We will skip the proof of this lemma since we do not need more than the result for further progression. Thus for every x, the size of C_x is at most $\frac{1}{4}2^m$ and the size of $\overline{C_x}$ is at least $\frac{3}{4}2^m$, which is at least three times the size of C_x. This means that for our above $\leq_{ktt}^{\mathsf{SNP}}$-reduction, every $x \in A$ of length n, $m = \left\lfloor \frac{20kn}{\gamma} \right\rfloor$ and $r \in \Sigma^m$, the probability that at least one query is larger than the length of the input is at least 3/4. This completes the proof of the claim.

We will now observe that we can derandomize the above reduction. For this last step we need a result about pseudorandom generators. Most of the known constructions of pseudorandom generators are based on certain hardness assumptions of the circuit complexity of a function. We will use the following pseudorandom generator out of a paper of Klovans and van Melkebeek.

Theorem 3.27 ([KvM02]). If there is a exponential-time computable function f with circuit complexity at least 2^{n^ε} $(\varepsilon > 0)$ relative to O-oracle, then there is a constant $a > 0$ and a pseudorandom generator $G : \Sigma^{\lfloor log(n)^a \rfloor} \rightarrow \Sigma^n$ that is secure against O-oracle.

Recall that the circuit complexity of a function relative to O-oracle is the size of the smallest O-oracle circuit that computes f on every input of size n, and moreover, note that our hypothesis (there exists a one-one, 2^{n^ε}-secure function against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits) implies the existence of such an exponential-time computable function whose $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuit complexity is 2^{n^δ} for some $\gamma > \delta > 0$ (consider f^{-1}). With the help of this hard function, it is possible to construct a pseudorandom generator $G : \Sigma^{\lfloor log(m)^a \rfloor} \rightarrow \Sigma^m$ that is secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits of size $\mathcal{O}(m)$. Thus for every $x \in A$ of length n, we have the following:

$$\Pr_{r\in\Sigma^{\lfloor log(m)^a\rfloor}}[\exists q_i \in Q_{x,G(r)} \text{ such that } |q_i| > n] \geq 1/2,$$

where $Q_{x,G(r)}$ is the set of all queries produced by $h(x, G(r))$.

Observe that without the pseudorandom generator, we had a probability of more than 3/4 to receive at least one query that is larger than the input, but since we henceforth consider strings $r \in \Sigma^{\lfloor log(m)^a\rfloor}$ and only obtain pseudorandomness, this results to a small decrease (compare with the definition of a pseudorandom generator). We will now describe the query generator of the derandomized reduction from A to L on input x of length n:

1. Set $m = \left\lfloor \frac{20kn}{\gamma} \right\rfloor$.
2. Cycle through all strings r of length $\lfloor log(m)^a\rfloor$ and compute $h(x, G(r)) = \langle q_1, ...q_k\rangle$, until there is at least one query being larger than n or until all r have been checked.
3. If there is such a query that is larger than n, output the tuple $\langle q_1, ...q_k\rangle$ and stop.
4. Otherwise output $\langle 0, ..., 0\rangle$.

If for every r, the length of every query of $h(x, G(r))$ is at most n, then by the previous inequality it must be the case that x is not in A, because if x would be in A, then for at least half of the $r \in \Sigma^{\lfloor log(m)^a\rfloor}$ there would be a query that is larger than the input. Thus if no such r can be found, then the reduction simply outputs $\langle 0, ..., 0, F\rangle$, where F is the truth-table for k variables that always evaluates to false. Observe that the run time of the query generator is deterministic quasi-polynomial since there exist $2^{\lfloor log(m)^a\rfloor}$ different r and the pseudorandom generator G can be computed in $2^{\mathcal{O}(\lfloor log(m)^a)\rfloor}$. Hence this is a correct $\leq^{\mathsf{SNQP}}_{ktt,wli}$-reduction from A to L and this completes the proof of Theorem 3.25. □

As mentioned before, the following theorem summarizes the results of this section and therefore is a corollary out of Observation 3.22 and 3.23 and Theorem 3.25.

Theorem 3.28 ([HMPRS12]). Suppose that there exists an $\varepsilon > 0$ and a one-one, one-way function that is 2^{n^ε}-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits. Then the ESY-$\leq^{\mathsf{P}}_{btt}$ conjecture holds.

If there is an $\varepsilon > 0$ and a one-one, one-way function that is 2^{n^ε}-secure against $\mathsf{NP} \cap \mathsf{coNP}$-oracle circuits, then by Observation 3.23 it holds that $\mathsf{NQP} \neq \mathsf{coNP}$ and by Theorem 3.25 it holds that every language that is $\leq_{ktt}^{\mathsf{SNP}}$-hard for NP is $\leq_{ktt,wli}^{\mathsf{SNQP}}$-hard for NP. Since those two implications fulfill the requirements for Observation 3.22, we finally obtain the truth of the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture.

With this result, we have achieved two hypotheses out of quite different areas and each of them implies the truth of the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture and therefore $\mathsf{NP} \neq \mathsf{coNP}$. Maybe there are some ways of weakening the hypothesis (we could, for example, consider the existence of one-way functions that are only hard against subexponential-size circuits having no oracles), relax the truth-table restriction out of the proofs to be unbounded or even discover other relations to the conjecture that provide easier accesses.

3.5 Comparison and open questions

We will now summarize the results we obtained and those which were previously known about the ESY-$\mathcal{R}$ conjectures. Furthermore, we will describe some open questions that need to be focused on. By investigating these, one perhaps can imagine more and different possibilities to achieve stronger statements or other accessible hypotheses. First of all, the following diagram shows what is known about the ESY-$\mathcal{R}$ conjectures:

$$\begin{tikzcd}
\mathsf{NP} \neq \mathsf{coNP} & \text{ESY-}\leq_1^{\mathsf{P}} \arrow[l, Rightarrow] & & \\
& & \text{ESY-}\leq_{m,li}^{\mathsf{P}} \arrow[r, Rightarrow] & \mathsf{NP} \neq \mathsf{coNP} \\
\mathsf{NP} \neq \mathsf{coNP} \arrow[r, Leftrightarrow, "\text{Prop. 3.13}"] & \text{ESY-}\leq_m^{\mathsf{P}} \arrow[u, Rightarrow] \arrow[ur, Rightarrow] \arrow[dr, Leftrightarrow] & & \\
\text{Unpredictable set} \arrow[dr, Rightarrow, "\text{Th. 3.20}"] & & \text{ESY-}\leq_{btt,li}^{\mathsf{P}} \arrow[uu, Rightarrow] \arrow[r, Leftrightarrow, "\text{Th. 3.19}"] & \mathsf{NP} \neq \mathsf{coNP} \\
\text{Secure one-way function} \arrow[r, Rightarrow, "\text{Th. 3.28}"'] & \text{ESY-}\leq_{btt}^{\mathsf{P}} \arrow[uu, Rightarrow] \arrow[ur, Rightarrow] & & \\
& & \text{ESY-}\leq_{tt,li}^{\mathsf{P}} \arrow[uu, Rightarrow] & \\
??? \arrow[r, Rightarrow] & \text{ESY-}\leq_{tt}^{\mathsf{P}} \arrow[uu, Rightarrow] \arrow[ur, Rightarrow] \arrow[r, Rightarrow] & \mathsf{NP} \neq \mathsf{UP}, \mathsf{SAT} \notin \mathsf{NPSV} & \\
??? \arrow[r, Rightarrow] & \text{ESY-}\leq_{T}^{\mathsf{P}} \arrow[u, Rightarrow] \arrow[r, Rightarrow] & \mathsf{NP} \neq \mathsf{UP}, \mathsf{SAT} \notin \mathsf{NPSV} &
\end{tikzcd}$$

Observe that since the ESY-$\leq_{btt,li}^{\mathsf{P}}$ conjecture and the ESY-$\leq_m^{\mathsf{P}}$ conjecture are both equivalent to $\mathsf{NP} \neq \mathsf{coNP}$ (by Proposition 3.13 and Theorem 3.19), they are also equivalent to each other. The question marks hint that there is no hypothesis known that implies either the ESY-$\leq_T^{\mathsf{P}}$ conjecture or the ESY-$\leq_{tt}^{\mathsf{P}}$ conjecture, which obviously results out of their wide range of consequences and is another evidence for the hardness of the task of finding proofs for them. Unfortunately, the belief is that the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture does not imply $\mathsf{NP} \neq \mathsf{UP}$, but there has been no proof up to now.

Though it is an open question whether every $\leq_{btt}^{\mathsf{P}}$-hard language for NP is hard for $\leq_{btt,li}^{\mathsf{P}}$-reductions, which would result to ESY-$\leq_{btt}^{\mathsf{P}}$ $\Leftrightarrow$ ESY-$\leq_{btt,li}^{\mathsf{P}}$ and ESY-$\leq_{btt}^{\mathsf{P}}$ $\Leftrightarrow$ $\mathsf{NP} \neq \mathsf{coNP}$, there exist several indications that every $\leq_{m}^{\mathsf{P}}$-complete language for NP is complete for $\leq_{m,li}^{\mathsf{P}}$-reductions [Agr02, HP07, BHHT10, GHP10]. Maybe one of the ideas out of those papers can be adapted to solve the respective problem for $\leq_{btt}^{\mathsf{P}}$-reductions and moreover, this would be one of the next steps for further progression on the ESY conjecture. Another approach is the investigation of even more restricted truth-table reductions where the reduction is only allowed to make $\mathcal{O}(log(n))$ queries.

Lastly to mention, it is also an interesting set of questions, whether some of the known or assumed implications and bi-implications hold relative to oracles. We note that there exists an oracle relative to which the ESY-$\leq_{m}^{\mathsf{P}}$ conjecture holds and $\mathsf{NP} = \mathsf{UP}$. Since ESY-$\leq_{m}^{\mathsf{P}}$ $\Leftrightarrow$ ESY-$\leq_{btt,li}^{\mathsf{P}}$, then there is also an oracle relative to which the ESY-$\leq_{btt,li}^{\mathsf{P}}$ conjecture holds and $\mathsf{NP} = \mathsf{UP}$. On the contrary, there has not yet been found an oracle relative to which the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture holds and $\mathsf{NP} = \mathsf{coNP}$.

4 Conclusion

The more than thirty-year-old ESY conjecture on promise problems, or in its equivalent recent form, on disjoint NP-pairs, is only one example out of a number of many unsolved assumptions and problems within complexity theory, yet another one that seems to provide some evidence on the big question whether we finally obtain $\mathsf{P} \neq \mathsf{NP}$ or not. Though the original conjecture for Turing reductions additionally implies the former main motivation of its formulation, which is the non-existence of any non-probabilistic public-key cryptosystems with NP-hard cracking problems and the non-existence of respective error-free probabilistic ones, it does not apparently seem to provide any feasible access. On the contrary, certain other ESY-R conjectures can be handled somehow easier, but unfortunately come up with a weaker set of consequences or even lose a few of the primary implications. Nevertheless, the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture, which we mainly considered in this thesis, is even equivalent to $\mathsf{NP} \neq \mathsf{coNP}$. Moreover, it provides two interesting hypothesis out of quite different areas. On the one hand, the existence of a certain unpredictable language and on the other hand, the existence of some kind of a secure one-way function, which both would imply the truth of the ESY-$\leq_{btt}^{\mathsf{P}}$ conjecture. Maybe we can find some further approaches to even connect certain other subjects to some of the ESY-R conjectures. Perhaps the ESY conjecture finally holds the key to confirm the widely believed assumption $\mathsf{P} \neq \mathsf{NP}$.

Bibliography

[AD97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 284-293, 1997.

[Agr02] M. Agrawal. Pseudo-random generators and structure of complete degrees. In *17th Annual IEEE Conference on Computational Complexity*, pages 139-145, 2002.

[ASFH87] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. In *Theoretical Computer Science*, 51:177-204, 1987.

[ASNT96] K. Ambos-Spies, H. Neis, and A. Terwijn. Genericity and measure for exponential time. In *Theoretical Computer Science*, 168(1):3-19, 1996.

[ASTZ97] K. Ambos-Spies, A. Terwijn, and X. Zheng. Resource bounded randomness and weakly complete problems. In *Theoretical Computer Science*, 172(1):195-207, 1997.

[AW09] M. Agrawal and O. Watanabe. One-way functions and the Berman-Hartmanis conjecture. In *Proceedings of the 24th IEEE Conference on Computational Complexity*, 2009.

[Bey11] O. Beyersdorff. Lecture notes for cryptography, Hannover, 2011.

[BHHT10] H. Buhrman, B. Hescott, S. Homer, and L. Torenvliet. Non-uniform reductions. In *Theory of Computing Systems*, 47(2):317-341, 2010.

[BM95] J. Balcazar and E. Mayordomo. A note on genericity and bi-immunity. In *Proceedings of the Tenth Annual IEEE Conference on Computational Complexity*, pages 193-196, 1995.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. In *IEEE Trans. Information Theory IT-22*, no. 6, pages 644-654, 1976.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions on Information Theory*, 31(4):469-472, 1985.

[ESY84] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. In *Information and Control*, 61(2):159-173, 1984.

[EY80] S. Even and Y. Yacobi. Cryptocomplexity and NP-completeness. In *Proc. 7th Colloq. on Automata, Languages and Programming*, Lecture Notes in Computer Science, volume 85, pages 195-207, 1980.

[FK93] M. Fellows and N. Koblitz. Combinatorial cryptosystems galore! In *Finite fields: theory, applications, and algorithms, Contemp. Math. 168*, pages 51-61, 1993.

[Gen09a] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169-178, 2009.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. http://crypto.stanford.edu/craig/craig-thesis.pdf, 2009.

[GHP10] X. Gu, J. Hitchcock, and A. Pavan. Collapsing and separating completeness notions under average-case and worst-case hypotheses. In *STACS*, volume 5 of *LIPIcs*, pages 429-440, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 25-32, 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. In *J. Comp. System Sci.*, 28:270-299, 1984.

[Gol06] O. Goldreich. On promise problems: A survey. In *Theoretical Computer Science - Essays in Memory of Shimon Even*, volume LNCS: 3895, pages 254-290, 2006.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. In *SIAM Journal on Computing*, 17(2):309-355, 1988.

[GSSZ04] C Glaßer, A. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. In *SIAM J. Comput.*, 33(6):1369-1416, 2004.

[GSZ07] C. Glaßer, A. Selman, and L. Zhang. Canonical disjoint NP-pairs of propositional proof systems. In *Theoretical Computer Science*, 370(1):60-73, 2007.

[HILL99] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way funcion. In *SIAM Journal on Computing*, 28(4):1364-1396, 1999.

[HMPRS12] A. Hughes, D. Mandal, A. Pavan, N. Russel, and A. Selman. A thirty year old conjecture about promise problems. In *ICALP'12, Proceedings of the 39th international colloq. conference on Automata, Languages, and Programming - Volume Part I*, pages 473-484, 2012. Preliminary version, full paper unpublished (May 2014).

[HP07] J. Hitchcock and A. Pavan. Comparing reductions to NP-complete sets. In *Information and Computation*, 205(5):694-706, 2007.

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *SIAM Journal on Computing*, 31:1501-1526, 2002.

[NS98] P. Nguyen and J. Stern. Cryptoanalysis of the Atjai-Dwork cryptosystem. In *CRYPTO*, pages 223-242, 1998.

[Pud01] P. Pudlak. on reducibility and symmetry of disjoint NP-pairs. In *Electronic Colloq. on Computational Complexity, technical reports*, 2001.

[Raz94] A. Razborov. On provably disjoint NP-pairs. In *Technical Report 94-006*, ECCC, 1994.

[Reg12] K. Regan. It don't come easy. In *Gödel's Lost Letter and* P = NP, http://rjlipton.wordpress.com/2012/07/14/it-dont-come-easy/, 2012.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Comm. ACM 21*, pages 120-126, 1978.

[SY82] A. Selman and Y. Yacobi. The complexity of promise problems. In *Proc. 8th Colloq. on Automata, Languages and Programming, Lecture Notes in Computer Science*, volume 140, pages 502-509, 1982.

[VV86] L. Valiant and Vijay Vazirani. NP is as easy as detecting unique solutions. In *Theoretical Computer Science*, volume 47, pages 85–93, 1986.