Relations among Statistical Security Notions or Why Exponential Adversaries are Unlimited

Dominique Unruh

Institut für Algorithmen und Kognitive Systeme,
Universität Karlsruhe (TH), 76128 Karlsruhe, Germany,
unruh@ira.uka.de

Abstract. In the context of Universal Composability, we introduce the concept of universal environments and simulators. Then, Universal Composability is equivalent to Universal Composability wrt. universal environments and simulators.
We prove the existence of universal environments and simulators and investigate their computational complexity.
From this, we get a number of consequences: First, we see that for polynomial-time protocols, exponential adversarial entities are as powerful as unlimited ones.
Further, for a large class of protocols (those with bounded communication-complexity) we can show that UC and specialised-simulator UC coincide in the case of statistical security, i.e., that it is does not matter whether the simulator is chosen in dependence of the environment or not. This also implies that for the Universal Composition Theorem for polynomial-time protocols specialised-simulator UC is sufficient.
This result is the last piece needed to find all implications and non-implications between the notions of UC, specialised-simulator UC, $O(1)$-bounded and polynomially-bounded general composability for polynomial-time protocols in the cases of perfect, statistical and polynomial security.
Finally, we introduce the notion of bounded-risk UC, which allows to give explicit security guarantees for concrete security parameters and show that in the above case also this variant coincides with UC.

1 Introduction

Independently, in [Can01] and [PW01] general frameworks for defining the security of reactive multiparty protocols (as opposed to multiparty function evaluations) were proposed that had the following important property: Given a protocol π for a primitive (or ideal functionality) $\mathcal{F}$, one could replace an occurrence of $\mathcal{F}$ in any larger context by the protocol π *without loosing security* (Composition Theorem). The general idea of both frameworks was the same: For π to securely implement $\mathcal{F}$, we require that any attack on π by an adversary $\mathcal{A}$ can be mimicked by a simulator $\mathcal{S}$ attacking the (by definition secure) ideal functionality $\mathcal{F}$. Since an attack that can be performed on the ideal functionality is by definition harmless, so is every attack on π. However, to decide whether the simulator successfully mimics the attack, a further entity was introduced, the environment that can interact with the protocol π and the adversary $\mathcal{A}$ (or $\mathcal{F}$ and $\mathcal{S}$) and has to guess whether it talks to π or $\mathcal{F}$.

Now in the model of [Can01] (the *UC framework*), the environment is chosen last, i.e., the simulator is not allowed to depend on the environment, while in the model of [PW01] (the *RS framework*) the simulator is allowed to depend on the environment. Interestingly, only with the order of quantifiers that was the default in the UC framework a stronger variant of the

Composition Theorem could be shown, where not only one occurrence of $\mathcal{F}$ could be replaced by π, but polynomially many simultaneously.[1]

Then, [Lin03] started a systematic comparison between these two orders of quantifiers in the UC framework. They showed, that the weaker order of quantifiers where the simulator is chosen last (called *specialised-simulator UC* there) is not only sufficient for the weaker form of the Composition Theorem where only one or a constant number of replacements is allowed (called *$O(1)$-bounded general composition*), but also necessary. However, they gave as an open question whether the stronger order of quantifiers (called simply *UC*) was *necessary* for the stronger variant of composability (called *polynomially-bounded general composability*). Moreover, it was even unclear, whether UC and specialised-simulator UC were different notions.

The first separating example was given in [HU05a]. They showed that while specialised-simulator UC and UC coincided in the case of perfect security, in the case of statistical and computational security, these two notions differed. In the computational case, the separating example consisted of a polynomial-time protocol, but for the separation in the statistical case, a protocol without a priori runtime-bound was necessary. It remained open whether such a separation could be achieved with a polynomial-time protocol.

Later, in [HU05b], it was shown that not only did UC and specialised-simulator UC differ in the case of computational security, but that computational specialised-simulator UC is not even sufficient for polynomially-bounded general composability. In contrast, they also showed that specialised-simulator UC guaranteed polynomially-bounded general composability in the case of statistical and perfect security.

1.1 Our work

We further examine the case of statistical security abd the computational complexity of the adversarial entities in that setting. We show that for protocols that have a bounded communication complexity (no matter how large), the security notions mentioned above coincide. In particular this implies, that for these protocols statistical specialised-simulator UC is sufficient for polynomially-bounded general composability.[2]

Further, we introduce another variant of the UC definition, bounded-risk UC, which can give concrete security guarantees for concrete security parameters (in contrast to the other definitions, where concrete security is only guaranteed given a concrete adversary). We show that for protocols of bounded-size, this variant also coincides with the notions above.

Our result negatively answers the open question from [HU05a] whether there is a separating example for UC and specialised-simulator UC in the statistical case that has bounded running time. Further, together with the works mentioned above, it gives us a complete picture of the relations between the different variants of security (UC, specialised-simulator UC, $O(1)$-bounded and polynomially-bounded general composability) for perfect, statistical and computational security *in the case of polynomially bounded protocols* (see Figure 1).

Finally, we are able to give tighter complexity bounds on the adversarial entities needed for these implications. In particular, we could show that security with respect to exponential-time adversarial entities is equivalent to statistical security (i.e., with respect to unlimited adversarial entities) in the case of polynomial time protocols. Similar complexity-results are given for other classes of protocols.

[1] This is indeed due to the order of quantifiers and not to other details in the modelling, since in [BPW04a] it was shown that this stronger variant does hold in the RS framework, if the stricter order of quantifiers was chosen.

[2] The latter was already known from [HU05b], but our approach is completely different and gives much tighter complexity bounds on the needed simulators (see below).

$$\begin{tikzcd}
\text{perf. UC} \arrow[r, leftrightarrow] \arrow[d, shift left] & \text{perf. poly.} \arrow[r, leftrightarrow] \arrow[d, shift left] & \text{perf. spec.} \arrow[r, leftrightarrow] \arrow[d, shift left] & \text{perf. } O(1) \arrow[d, shift left] \\
\text{stat. UC} \arrow[r, leftrightarrow] \arrow[u, shift left, "/" marking] \arrow[d, shift left, "/" marking] & \text{stat. poly.} \arrow[r, leftrightarrow] \arrow[u, shift left, "/" marking] \arrow[d, shift left, "/" marking] & \text{stat. spec.} \arrow[r, leftrightarrow] \arrow[u, shift left, "/" marking] \arrow[d, shift left, "/" marking] & \text{stat. } O(1) \arrow[u, shift left, "/" marking] \arrow[d, shift left, "/" marking] \\
\text{comp. UC} \arrow[r, bend left] \arrow[u, shift left, "/" marking] & \text{comp. poly.} \arrow[l, bend left, "/" marking] \arrow[r, bend left] \arrow[u, shift left, "/" marking] & \text{comp. spec.} \arrow[l, bend left, "/" marking] \arrow[r, leftrightarrow] \arrow[u, shift left, "/" marking] & \text{comp. } O(1) \arrow[u, shift left, "/" marking]
\end{tikzcd}$$

Fig. 1. Implications and separations between various security notion in the case of polynomial-time protocols. Perf. stands for perfect security, stat. for statistical security, comp. for computational security, UC for UC security, poly. for polynomially-bounded general composability, spec. for specialised-simulator UC, and $O(1)$ for $O(1)$-bounded general composability. The relation between any pair of two notions can be deduced from the arrows depicted here. The results and computational assumptions used in this diagram are listed in Appendix A.1.

2 The Universal Composability Framework

In this section, we give a sketch of the Universal Composability (UC) framework from [Can05]. Due to the complexity of the topic, we can only give a very rough introduction. For a full understanding of this paper, we strongly encourage the reader to familiarise himself with either the UC framework as described in [Can05] or the Reactive Simulatability framework as in [BPW04b].

Since we are concerned with statistical security in this paper (i.e., security with respect to unbounded adversarial entities), we directly give the definitions adapted to the statistical case (the definitions in [Can05] are given for the computational case only).

The basic idea of the UC framework is to define security by comparison: We say a protocol π implements another protocol ρ (or: is as secure as ρ), if for any adversary $\mathcal{A}$ that attacks the protocol π, there is a simulator $\mathcal{S}$ that mimics that attack on protocol ρ, s.t. no environment $\mathcal{Z}$ can distinguish whether it is communicating with instances of π and $\mathcal{A}$ or with instances of ρ and $\mathcal{S}$.

To define this more formally, let the random variable $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k)$ denotes the output that $\mathcal{Z}$ gives when it terminates after running in a network with π and $\mathcal{A}$ with security parameter k (for the exact behaviour of the network and a definition of EXEC see [Can05]).

Then we can define security as follows:

Definition 1 (Universal composability (statistical case)). *Let π and ρ be protocols. We say π* implements *ρ* with respect to adversaries in *$C_\mathcal{A}$*, environments in *$C_\mathcal{Z}$* and simulators in *$C_\mathcal{S}$ if for any adversary $\mathcal{A} \in C_\mathcal{A}$ there is a simulator $\mathcal{S} \in C_\mathcal{S}$ s.t. for every environment $\mathcal{Z} \in C_\mathcal{Z}$ with one-bit output*[3] *$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k), \mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k))$ is negligible in k. (Here Δ denotes the statistical distance, cf. Def. 5.)*

By specifying the classes $C_\mathcal{A}$, $C_\mathcal{Z}$ and $C_\mathcal{S}$, we can tune this definition to get many different flavours of UC. When we do not specify $C_\mathcal{A}$, $C_\mathcal{Z}$ and $C_\mathcal{S}$, we assume security with respect to unlimited adversaries/environments/simulators, i.e., $C_\mathcal{A}$, $C_\mathcal{Z}$ and $C_\mathcal{S}$ are the set of all machines. For ease of language, we say *adversarial entities* instead of adversaries/environments/simulators.

The reader familiar with [Can05] may have noticed that we have omitted the explicit statement of the auxiliary input z both from the definition of EXEC and from that of UC. However, the auxiliary input can be included by letting $C_\mathcal{Z}$ contain machines with auxiliary input. E.g.,

[3] See [Can05] for a discussion on the restriction to environments with one-bit output.

the definition of computational UC from [Can05] is then captured by our definition as UC with respect to polynomial-time adversaries, polynomial-time environments with auxiliary input, and polynomial-time simulators. So, using our formulation we get the additional flexibility of being able to handle both uniform and non-uniform security in one definition.

Definitions based on simulatability have the great advantage of allowing for Composition Theorems. Composition theorems state, roughly, that when a protocol ρ implements a functionality $\mathcal{F}$, and a protocol σ using $\mathcal{F}$ implements τ, then we can replace $\mathcal{F}$ by ρ in σ, resulting in a protocol $\sigma^{\mathcal{F}/\rho}$. The resulting protocol $\sigma^{\mathcal{F}/\rho}$ then still implements τ. This property is very important for the modular design of secure protocols. Different flavour of Composition Theorems exist, the most important difference being whether σ is allowed to call polynomially many or only a single copy of $\mathcal{F}$. If polynomially many copies are allowed, we speak of *polynomially-bounded general composability,* and if only a single copy (or a constant number) is allowed, we speak of $O(1)$*-bounded general composability* [Lin03].

A fine point in the definition of UC is the order of the quantifiers, i.e., whether the simulator may depend on the environment or not. In the above definition, the same simulator had to be used for all environments. In comparison, in [Lin03] the following notion was introduced for the UC framework (in the RS framework, this order of quantifiers was introduced earlier under the name of *standard security* [PW01]):

Definition 2 (Specialised-Simulator UC (statistical case)). *Let π and ρ be protocols. We say π* implements ρ with respect to specialised-simulator UC, adversaries in $C_\mathcal{A}$, environments in $C_\mathcal{Z}$ and simulators in $C_\mathcal{S}$ *if for any adversary $\mathcal{A} \in C_\mathcal{A}$ and any environment $\mathcal{Z} \in C_\mathcal{Z}$ there is a simulator $\mathcal{S} \in C_\mathcal{S}$, s.t. $\varDelta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k), \mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k))$ is negligible in k.*

When we quantify only over environments $\mathcal{Z}$ with one-bit output, we speak of one-bit specialised-simulator UC.

We have again omitted the auxiliary input for the reasons mentioned above. Note that we have added the definition of *one-bit* specialised-simulator UC, which in itself is a somewhat unnatural notion (cf. [Lin03]). We have only included it here, because of the interesting fact that it sometimes coincides with specialised-simulator UC (Corollary 4). That again allows to see that intermediate notions (e.g., computational indistinguishability of $\mathcal{Z}$'s output) also coincide.

Note that in contrast to UC, specialised-simulator UC does not allow for a Universal Composition Theorem [HU05b]. However, it is sufficient to allow for a weaker variant of the Composition Theorem where only a constant number of copies of the functionality $\mathcal{F}$ are replaced [Lin03].

A further definitional variant is motivated by the following reasoning: When choosing which value to use for the security parameter when using a protocol, it would be very helpful to have an upper bound on the probability with which the adversary is successful. Of course, with most (probably all) protocols based on computational assumptions such bounds cannot be given, since for a given security parameter k, there is a polynomial adversary that is just powerful enough to break the protocol for parameter k. In contrast, if a protocol is statistically secure, in many cases we can provide such a bound that is independent of the chosen adversary. Therefore when studying statistical security, the following variant of the UC definition makes sense:

Definition 3 (Bounded-risk UC (statistical case)). *Let π and ρ be protocols. We say π* implements ρ with respect to adversaries in $C_\mathcal{A}$, environments in $C_\mathcal{Z}$ and simulators in $C_\mathcal{S}$ *if there is a negligible function μ, s.t. for any adversary $\mathcal{A} \in C_\mathcal{A}$ there is a simulator $\mathcal{S} \in C_\mathcal{S}$ s.t. for every environment $\mathcal{Z} \in C_\mathcal{Z}$ with one-bit output and every $k \in \mathbb{N}$, $\varDelta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k), \mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k)) < \mu(k)$.*

In the case of statistical security, it is hard to imagine protocols that are secure with respect to UC but not to bounded-risk UC. For the case of protocols with a bound on the amount of their communication (bounded-size protocols), we indeed show in Corollary 4 that these notions coincide.

An important tool for proofs about simulatability is the dummy-adversary [Can05]. Roughly spoken, the dummy adversary is a machine that simply forwards everything the environment tells him to. More to the point, upon a message from the protocol, the dummy-adversary forwards that message to the environment (together with the id of the original sender), and upon a message (id, m) from the environment, the dummy-adversary forwards m to id. Now, in most cases, it turns out that if a protocol is secure with respect to the dummy-adversary, it is secure with respect to all adversaries. Unfortunately however, the dummy-adversary as described above is unbounded (since there is no limit on the number or length of messages). In most variants of the definition of UC such an unbounded adversary is not a valid one. In order to meet that problem, one often has to use dummy-adversaries with a fixed upper bound on their communication-complexity:

Definition 4 (Dummy-adversary). *Let p be a function. Then the p-dummy-adversary $\tilde{\mathcal{A}}_p$ is an adversary with the following behaviour:*

Whenever $\mathcal{A}$ receives a message m from the protocol, it sends (m, id) to the environment, where id is the sender of m. This also applies to special messsages like corruption-responses, etc. Whenever $\mathcal{A}$ receives a message (m, id) from the environment, it sends m to id. This also applies to special messages like corruption-requests, etc. However, each message m is truncated to the length $p(k)$, where k is the security parameter. And at most $p(k)$ messages from the protocol and at most $p(k)$ messages from $\mathcal{Z}$ are accepted (further messages lead to the termination of $\tilde{\mathcal{A}}_p$).

Note that the p-dummy-adversary is obviously $O(p)$-sized and $\mathrm{poly}(p)$-time (see Def. 7 and 6). We will make use of this construction in the proof of Corollary 3.

3 Notation and tools

In this section we present our notation and introduce some definitions and tools used througout this paper.

Definition 5 (Miscellaneous). *$\mathbb{N}$ denotes the set of all positive integers. When we talk of a* function *without specifying domain and range, we always assume real-valued functions on $\mathbb{N}$. The argument to a function we usually name k if not specified otherwise. A function μ is* negligible *if for every polynomial p, $\mu(k) < 1/p(k)$ for all sufficiently large k. For a function or set of functions p, $\mathrm{poly}(p)$ is the set of all functions bounded polynomially in p, i.e., $\mathrm{poly}(p) := \bigcup_{c>0} O(p^c)$, and $\mathrm{EXP}(p)$ is the set of all functions bounded exponentially in p, i.e., $\mathrm{EXP}(p) := 2^{poly(p)}$. For $\mathrm{EXP}(k)$ and $\mathrm{poly}(k)$ we simply write EXP and poly (the sets of exponentially and polynomially bounded functions). If X and Y are two discrete random variables, the statistical distance $\Delta(X, Y)$ between X and Y is defined as $\Delta(X, Y) := \max_T |P(X \in T) - P(Y \in T)| = \frac{1}{2}\sum_a |P(X = a) - P(Y = a)|$.*

3.1 Machine classes and complexity

Definition 6 (Time complexity). *Let f be a function, and A be a probabilistic algorithm.*

A runs in f-time *in k if for any $\varepsilon > 0$ and all inputs x the probability that the running time of A exceeds $f(k)$ is zero.*

A runs in almost-f-time in k *if for any $\varepsilon > 0$ and all inputs x the probability that the running time of A exceeds $f(k - \log \varepsilon)$ is bounded by ε.*

If F is a set of functions, we say A runs in F-time/almost-F-time in k *if it runs in f-time/almost-f-time in k for some $f \in F$.*

A runs in bounded-time *if it runs in f-time for some function f.*

We write polynomial-time *for* poly*-time and* exponential-time *of* EXP*-time. Analogously* almost-polynomial-time *and* almost-exponential-time.

If A is an ITM (interactive Turing machine, in contrast to a probabilistic algorithm), we require the runtime bounds to hold for all possible interactions (instead of all inputs x). Furthermore, k always denotes the security parameter in this case.

A machine without any restrictions (than can evaluate everything, even, e.g., the halting problem) we call unlimited*, a machine that can be implemented by a probabilistic Turing machine (without any runtime-bound) we call* Turing-unlimited.

Note the fact that we have used formulations like polynomial-time in k without specifying what k is. This allows to capture different notions within one definition, like e.g., being polynomial-time in the security parameter, or being polynomial-time in the length of one's input.

Note further the notion of almost-f-time. The intuition behind this class is of being in f-time with very high probability. Note that one can easily see that almost-polynomial-time is strictly contained in expected polynomial-time as defined in e.g., [Lev86]. Lemma 1 in Appendix A gives an example for a natural algorithm which is almost polynomial time, but not polynomial time.

Besides the above running-time-based classes of machines, we will need classes that put restrictions on the communication made by the machines, without placing any restrictions on the complexity of their internal computations:

Definition 7 (Communication complexity). *Let A be an ITM (interactive Turing machine) and f a function. We call A* f-sized *if for any possible interaction, the following holds: The maximum number of activations of A is bounded by $f(k)$, and the length of the output within each activation is bounded by $f(k)$, and the machine reads at most the first $f(k)$ symbols of each incoming message.*

If F is a set of functions, we call A F-sized *if it is f-sized for some $f \in F$.*

We say A is bounded-size *if it is f-sized for some function f.*

3.2 Game-theoric notions

In this section we give an overview of the game-theoretic notions and results used in this paper. These are only required to understand the proof of Theorem 2. The statements of all results in this paper are formulated without any reference to game-theoretic notions. For a deeper understanding of these notions, we refer the reader to standard textbooks on game-theory (e.g., [Ras89]).

Definition 8 (Game). *An n-*player-game *G (in* extensive form*) consists of a game-tree G, a set of information sets $\mathcal{U}$, and for each player $i = 1, \dots, n$ a payoff-function H_i.*

The game-tree *G is a finite tree,[4] where each node has one of the following types: (i)* player-i-node *for some $i = 1, \dots, n$, (ii)* chance-node*, or (iii)* leaf*. Each player-node or chance-node has at least one outgoing edge (called* moves*), the outgoing edges of player-nodes are labelled with names (so that no player-node has two outgoing edges with the same name), and the outgoing*

[4] It is possible to allow infinite trees, too. However, we only need finite ones in this paper.

edges of chance-nodes are labelled with probabilities (so that the sum of the probabilities of the edges of any given chance-node is one).

The set of information sets $\mathcal{U}$ is a partition of the player-nodes, s.t. within one information set all nodes appertain to the same player and have the same number and names of outgoing edges.

The payoff-function H_i is a function from the leafs of the game-tree into the real numbers.

The intuition behind this is that each node in the game-tree represents a given state of the game. In this state either a given player i may make a move (player-i-node), or a random move happens (chance-node), or the game ends (leaf). At the end of the game, the payoff-function of player i tells us, how much player i gains from this particular outcome of the game. Since in many games hidden moves may occur, a player does not necessarily know, in which node he finds himself, even if it is his turn to move. This is modelled by the information sets. At any given point in time, the current player only knows in what information set he finds himself, and therefore he can perform decisions only based on this information.[5]

In many cases, there are just two players with opposite interests. Then we talk of a zero-sum two-player game:

Definition 9 (Zero-sum two-player games). *A two-player game G is zero-sum, if $H_1 = -H_2$. In such cases, we only give the payoff-function $H = H_1$ for player-I.*

A player participating in a game can be modelled by specifying his strategy, i.e., in what way he reacts to what situation. There are several types of strategies:

Definition 10 (Strategy). *A* pure strategy *s_i for player-i assigns a move to each information set of payer-i (where that move must be an available move from the nodes within that information set).*

The expected payoff $H_i(s_1, \dots, s_n)$ for player i when the players adopt strategies $s_1, \dots, s_n$ is obtained as follows: The game starts at the root of the game tree. On a player-i-node the strategy s_i is queried which move to perform. On a chance-node the next node is chosen according to the probabilities on the outgoing edges. This process finally reaches a leaf, to which the payoff-function of player-i associates a payoff h_i. The expected value of h_i in this random process is the expected payoff *$H_i(s_1, \dots, s_n)$.*

A mixed strategy *μ_i for player-i is a probability distribution on the pure strategies of player-i.*

The expected payoff $H_i(\mu_1, \dots, \mu_n)$ for player i when the players adopt strategies $s_1, \dots, s_n$ is the expected value of $H_i(s_1, \dots, s_n)$ when $s_1, \dots, s_n$ are chosen independently according to the distributions $\mu_1, \dots, \mu_n$.

A behaviour strategy *β_i for player-i assigns a probability distribution on the available moves to each information set of payer-i.*

The expected payoff $H_i(\beta_1, \dots, \beta_n)$ is defined like the expected payoff for pure strategies except that the moves of a player are chosen randomly at each node according to the probability distribution given by the strategy.

A pure strategy represents a strategy of a deterministic entity. A mixed strategy introduces the possibility of randomisation which is a very important tool both in game-theory and cryptography. A behaviour strategy is similar to a mixed strategy, except that the choice of the moves at

[5] One even assumes that the player does not even remember prior events unless this is explicitly modelled by the information sets. This reflects e.g., the fact that a player may consist of several disjoint but cooperating entities which do not know the moves of their peers. A popular example for such a game is Bridge.

different node cannot be correlated.[6] Behaviour strategies have the advantage over mixed strategies that their representation is much smaller. While mixed strategies are vectors of exponential dimension in the size of the game tree, the dimension of behaviour strategies is linear in the size of the game tree. And it turns out that for an important class of games, the games with *perfect recall*, behaviour strategies are as powerful as mixed strategies (see [Kuh56] for more details).

Definition 11 (Perfect recall (informal definition)). *A game G has* perfect recall *if every player remembers all his past information sets and all his moves.*

In other words, if two player-i-nodes a and b are in the same information set, then all the information sets player-i visited and the moves player-i chose on the path from the root to a are the same as on the path to b.

A formal definition can be found e.g., in [Kuh56]. Games with perfect recall are quite natural, indeed, if we assume that a player is a single agent capable of remembering past observations and actions, games with perfect recall are the natural modelling.

It is easily seen that pure and behaviour strategies can be seen as special cases of mixed strategies. In the first case, the induced mixed strategy simply assigns probability 1 to the given pure strategy. In the second case all the choices performed by the behaviour strategy are simply performed beforehand, yielding a mixed strategy (a formal treatment is found in [Kuh56]).

In game theory, many different definitions of "good" strategies exist. The most common one is the nash-equilibrium. The idea is that the strategies of the different players form a nash-equilibrium if no player would gain an advantage by changing his strategy. For our purposes only the nash-equilibrium for two-player zero-sum games is important:

Definition 12 (Nash-equilibrium). *A nash-equilibrium (in mixed strategies) for a two-player zero-sum game G consists of mixed strategies μ_1^*, μ_2^* for player-I and player-II, s.t. for any player-I- and player-II-strategies μ_1' and μ_2', it holds that $H(\mu_1', \mu_2^*) \le H(\mu_1^*, \mu_2^*)$ and $H(\mu_1^*, \mu_2') \ge H(\mu_1^*, \mu_2^*)$.*

It turns out that nash-equilibria for two-player zero-sum games with perfect recall can be efficiently found:

Theorem 1 (Complexity of nash-equilibria). *There exists a deterministic polynomial-time algorithm performing the following task: It takes the description of a game as input and outputs two behaviour strategies that form a nash-equilibrium (when seen as mixed strategies).*

The game is assumed to be given by an explicit description of the game-tree, the set of information sets and the payoff-function. All probabilities and payoffs are given as rational numbers (as pairs of nominator and denominator).

A behaviour strategy is given by explicitly specifying a distribution on the possible moves for each information set. A distribution is given by listing all probabilities as rational numbers.

This theorem is proven in [KM92]. They reduce the task to a linear programming problem which then can be solved using the algorithm given by [GLS88, Th. 6.4.9].

[6] This is indeed less powerful than the notion of a mixed-strategy: Imagine a game where a player may choose two bits, but when choosing the second bit is not able to remember his first decision. A mixed strategy might be to choose with probability $\frac{1}{2}$ the bits 11, and with probability $\frac{1}{2}$ the bits 00. A behaviour strategy would be unable to mimic this, since both bits would have to be chosen independently.

4 Universal environment and simulator

In this section we will introduce the notion of universal environments and simulators, and show several results concerning their existence and complexity. In Section 5 we will see some applications of universal environments and simulators.

Definition 13 (Universal environment and simulator). *Let π and ρ be protocols and $\mathcal{A}$ an adversary. Let further ε denote a function. We call an environment $\mathcal{Z}^*$ and a simulator $\mathcal{S}^*$* ε-universal for π, ρ and $\mathcal{A}$, *if for any (unlimited) simulator $\mathcal{S}'$ and any (unlimited) environment $\mathcal{Z}'$ it holds that*

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k)) \geq \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k)) - \varepsilon(k) \qquad (1)$$

(i.e., the environment $\mathcal{Z}^*$ distinguishes at least as good as any other environment when running with $\mathcal{S}^*$ up to an exactitude of $\varepsilon(k)$), and

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k)) \leq \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}',\mathcal{Z}^*}(k)) + \varepsilon(k) \qquad (2)$$

(i.e., the simulator $\mathcal{S}^*$ simulates as good as any other simulator when running with $\mathcal{Z}^*$ up to an exactitude of $\varepsilon(k)$).

If $\varepsilon = 0$, we say $\mathcal{Z}^*$ and $\mathcal{S}^*$ are perfectly universal*, and if ε is negligible, we call $\mathcal{Z}^*$ and $\mathcal{S}^*$* statistically universal.

The reader familiar with game-theoretic notions will recognise that a pair of perfectly universal environment and simulator are nothing else than a nash-equilibrium. We have chosen to call environments and simulators according to Definition 13 *universal* since in many situations we can w.l.o.g. restrict the security definition to such environments/simulators without making it stronger or weaker (cf. e.g., the proof of Corollary 3).

The following theorem (and the following two corollaries) now tells us, that universal environments and simulators do indeed exist for the large class of bounded-size protocols and adversaries, and gives us upper bounds on their complexity.

Theorem 2 (Universal environment and simulator). *Let p be a positive function in the security parameter k. Let π and ρ be p-time protocols, and let $\mathcal{A}$ be a p-time adversary.*

Then there exist perfectly universal environment $\mathcal{Z}^*$ and simulator $\mathcal{S}^*$ for π, ρ and $\mathcal{A}$.

*Both $\mathcal{Z}^*$ and $\mathcal{S}^*$ run in almost-*EXP(p)*-time, are $O(p)$-sized, and do not maintain any state between invocations with exception of their respective view. $\mathcal{Z}^*$ has one-bit output.*

The rough idea of the proof is as follows: We design a game G for environment $\mathcal{Z}$ and simulator $\mathcal{S}$ to play that has the following rules: It is randomly chosen whether the real protocol (with adversary $\mathcal{A}$) or the ideal protocol is executed. The simulator is only allowed to to play if the ideal protocol has been chosen. The environment then wins if it guesses correctly whether the real of the ideal protocol has been played. The expected payoff in this game is directly related to the statistical distance between $\mathcal{Z}$'s views in runs of the real and the ideal protocol (with simulator $\mathcal{S}$). We then use Theorem 1 to construct a environment- and simulator-strategies that form a nash-equilibrium for this game. These directly correspond to a universal pair of environment and simulator. Care has however to be taken since Theorem 1 only applies for finite game-trees, while $\mathcal{Z}$ and $\mathcal{S}$ have no bound on their communication complexity, resulting in principle in an infinite game. This is solved by deriving a communication bound from the communication complexity of the protocols π, ρ and the adversary $\mathcal{A}$, and then showing that environment and simulator can w.l.o.g. be assumed not be communicate more than this bound.

Proof. Before we begin, we need an auxiliary definition similar to that of p-sized machines: We call an environment (p,p)*-sized*, if it sends and reads at most p messages of length at most p to/from the protocol, it sends and reads at most p messages of length at most p to/from the adversary/simulator. In a way, a (p,p)-sized environment is "*separately* p-sized for the protocol and for the adversary/simulator". Analogously we call a simulator (p,p)*-sized*, if it sends and reads at most p messages of length at most p to/from the protocol, it sends and reads at most p messages of length at most p to/from the environment. Obviously, a (p,p)-sized environment/simulator is $O(p)$-sized.

To show the theorem, we will first transform the problem into a game theoretic setting.

First, for a given security parameter k, we transform π and $\mathcal{A}$ into a game R_k (the *real game*). Roughly, this game is defined so that any actions performed by π and $\mathcal{A}$ are performed with the probabilities given by the programs of π and $\mathcal{A}$, while the actions of the (p,p)-sized environment $\mathcal{Z}$ are choosen by player-I. The payoff of that game is the value of the bit that $\mathcal{Z}$ outputs. Any (p,p)-sized $\mathcal{Z}$ can then be considered as a strategy $\mathcal{Z}_k$ for that game.

We now give the formal details of that game. This can be skipped at a first reading. Let $\tilde{\mathcal{Z}}$ be an arbitrary (p,p)-sized environment with one-bit output interacting with π and $\mathcal{A}$. Consider a protocol execution of $\tilde{\mathcal{Z}}$ with π and $\mathcal{A}$ with security parameter k (as defined in [Can05]). For any state that can be reached in the protocol execution, add a node to the game tree annotated with the state of all machines except $\tilde{\mathcal{Z}}$ and with $\tilde{\mathcal{Z}}$'s view[7].

Every node n that represents a state where $\tilde{\mathcal{Z}}$ is activated is a player-I-node, and the nodes reachable from n are the states directly reachable from state n (given any behaviour a (p,p)-sized $\tilde{\mathcal{Z}}$ could show), i.e., the valid moves at that node correspond to the messages $\tilde{\mathcal{Z}}$ can send. The root of the game-tree is the node corresponding to the initial state of the system.

Every node n where another machine than $\tilde{\mathcal{Z}}$ is activated is a chance-node, where the probabilities of reaching another node n' is the probability of the transition from state n to state n' in the protocol execution. Note that the definition of the game tree is independent of the actual code of $\tilde{\mathcal{Z}}$, since all nodes where the code of $\tilde{\mathcal{Z}}$ would be considered have been replaced by player-I-nodes.

The information sets of R_k are as follows: All player-I-nodes annotated with the same view of $\tilde{\mathcal{Z}}$ are in one information set.

The payoff H_{R_k} (for player-I) is defined as follows: Any leaf-node in the game-tree of R_k constitutes a final-state of the protocol-execution, i.e., a state where $\mathcal{Z}$ has written its one-bit output to its output tape. The payoff is 0 if that bit was 0, it is 1 if that bit was 1.

Since the environment $\tilde{\mathcal{Z}}$ is (p,p)-sized, and all other machines are p-time (and therefore p-sized), the game tree has a depth of $O(p)$ and a fan-out of $O(2^p)$ (since the message length is bounded by p), so the size of the game tree is bounded by $O(2^p)^{O(p)} \subseteq \mathrm{EXP}(p)$. Note that R_k does not contain player-II-nodes, i.e., it is a one-player-game.

Any (p,p)-sized environment $\mathcal{Z}$ with one-bit output now induces a mixed strategy $\mathcal{Z}_k$ for R_k in a natural manner: For any content t of the random tape of $\mathcal{Z}$ the behaviour of $\mathcal{Z}$ on security parameter k (and therefore the next state of the protocol execution) is deterministically given by t and $\mathcal{Z}$'s view v at that point. So at any player-I-node, we define the pure strategy $\mathcal{Z}_k^t$ to choose the next state that is given by t and v. $\mathcal{Z}_k^t$ is a legal pure strategy, since its choices depend only on the view of $\mathcal{Z}$, which is the same for all nodes of an information set. Now, we simply let $\mathcal{Z}_k$ be the mixed strategy resulting from choosing t uniformly and then behaving as does $\mathcal{Z}_k^t$.

[7] By view we mean the sequence of all messages sent and received by $\tilde{\mathcal{Z}}$, but not including the internal states of $\tilde{\mathcal{Z}}$.

By construction of R_k and $\mathcal{Z}_k$, the expected payoff of R_k with strategy $\mathcal{Z}_k$ is exactly the probability that $\mathcal{Z}$ outputs 1 in a protocol execution with π and $\mathcal{A}$ on security parameter k, i.e.,

$$H_{R_k}(\mathcal{Z}_k) = P(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k) = 1). \tag{3}$$

Now we construct a game I_k for the ideal model. The definition is similar to R_k, except that now the actions of the simulator are choosen by player-II, and only the actions of the protocol are chance-moves. Again, the formal description can be skipped at the first reading.

Formally, this game is defined analogously to R_k, with the following differences: The nodes correspond to states of a protocol execution of $\tilde{\mathcal{Z}}$ with ρ and $\tilde{\mathcal{S}}$ upon security parameter k (where $\tilde{\mathcal{Z}}$ and $\tilde{\mathcal{S}}$ are arbitrary (p,p)-sized environment and simulator). A node is annotated with the state of system excluding $\tilde{\mathcal{Z}}$ and $\tilde{\mathcal{S}}$ and with the views of $\tilde{\mathcal{Z}}$ and $\tilde{\mathcal{S}}$. All nodes corresponding to $\tilde{\mathcal{Z}}$'s activations are player-I-nodes, as before, all states corresponding to $\tilde{\mathcal{S}}$'s activations are player-II-nodes, and all other nodes are chance-nodes with the transition-probabilities as before. Information sets for player I consist of player-I-nodes with the same views of $\tilde{\mathcal{Z}}$, and information sets for player II consist of player-II-nodes with the same views of $\tilde{\mathcal{S}}$. The payoff H_{I_k} is defined as for R_k.

Analogously to above, we see that the size of the game tree of I_k lies in $\mathrm{EXP}(p)$.

Again, a (p,p)-sized $\mathcal{Z}$ with one-bit output gives rise to the mixed player-I-strategy $\mathcal{Z}_k$, and analogously a (p,p)-sized simulator $\mathcal{S}$ gives rise to a mixed player-II-strategy $\mathcal{S}_k$. Then by construction of the game I_k and the mixed strategies $\mathcal{Z}_k$ and $\mathcal{S}_k$, it is

$$H_{I_k}(\mathcal{Z}_k, \mathcal{S}_k) = P(\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k) = 1). \tag{4}$$

From R_k and I_k we can construct a third game G_k. The game-tree of G_k consists of the game-trees of R_k and I_k together with a chance-node c as root, the successors of which are the root nodes of R_k and I_k, each with probability $\frac{1}{2}$. Intuitively, the game G_k randomly chooses which game is played, R_k or I_k. The information sets of player I consist of player-I-nodes with the same view of $\mathcal{Z}$ (note that nodes from R_k and I_k may be in the same information set, reflecting the fact that $\mathcal{Z}$ is not told whether it runs in the ideal or real model). The information sets of player II consist of player-II-nodes with the same view of $\mathcal{S}$ (since there are no player-II-nodes in R_k, the player-II-information sets of G_k are the same as those of I_k). Finally, the (player-I-)payoffs H_{G_k} of G_k for leafs of R_k are those of R_k, but the payoffs for leafs of R_k are the negative of those of I_k.

The size of the game-tree of G_k is the sum of the game-trees of R_k and I_k plus one, so it also lies in $\mathrm{EXP}(p)$.

Since G_k randomly chooses which game to play, we directly have for any (p,p)-sized $\mathcal{Z}$ and $\mathcal{S}$:

$$\begin{aligned} H_{G_k}(\mathcal{Z}_k, \mathcal{S}_k) &= \frac{H_{R_k}(\mathcal{Z}_k) - H_{I_k}(\mathcal{Z}_k, \mathcal{S}_k)}{2} \\ &\overset{(3,4)}{=} \frac{P(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k) = 1) - P(\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k) = 1)}{2} \end{aligned} \tag{5}$$

Since the view of a machine at a given point in time contains the view of that machine at all prior times, and further contains all messages send so far by that machine, the games R_k, I_k and therefore also G_k have perfect recall.

By Theorem 1 there is a deterministic algorithm to find a nash-equilibrium consisting of behaviour strategies $(S^*_{1,k}, S^*_{2,k})$ for G_k, that is polynomial in the size of the game tree of G_k. Since the size of G_k is in $\mathrm{EXP}(p)$, the complexity of finding that nash-equilibrium lies in $\mathrm{poly}(\mathrm{EXP}(p)) = \mathrm{EXP}(p)$-time.

A (p,p)-sized environment $\mathcal{Z}^*$ s.t. $\mathcal{Z}^*_k = S^*_{1,k}$ for *all* security parameters k can be constructed as follows: Upon each activation, $\mathcal{Z}^*$ evaluates the abovementioned algorithm and determines $S^*_{1,k}$. Since $S^*_{1,k}$ is a behaviour strategy, the action taken by $S^*_{1,k}$ depends only on the information set of the current node, i.e., of the view of $\mathcal{Z}^*$ in the current state. So the probabilities for the different actions (i.e., messages sent or outputs made) to be taken by $\mathcal{Z}^*$ are given as a list of rational numbers (the length of the list is bounded by $\mathrm{EXP}(p)$). $\mathcal{Z}^*$ now chooses one of the actions with the given probability and performs it. By construction of the game-tree, $\mathcal{Z}^*$ then is indeed (p,p)-sized.

Since π, ρ and $\mathcal{A}$ are p-time, there are at most $\mathrm{EXP}(p)$ different execution paths of π, ρ and $\mathcal{A}$, each no longer than p steps. Therefore the game trees of R_k and I_k (with the associated probabilities) can be constructed in $\mathrm{EXP}(p)$-time. Then also G_k can be constructed in $\mathrm{EXP}(p)$-time.[8] The time-complexity for calculating the nash-equilibrium is $\mathrm{EXP}(p)$, and choosing the action can be performed in almost-$\mathrm{EXP}(p)$-time by Lemma 1 in Appendix A. This has to be performed at most $O(p)$ times (since $\mathcal{Z}^*$ is $O(p)$-sized), so the complexity of $\mathcal{Z}^*$ is almost-$\mathrm{EXP}(p)$-time. Since $\mathcal{Z}^*$ calculates the nash-equilibrium upon every activation, it does not maintain a state other than its view. $\mathcal{Z}^*$'s output is one-bit because of the construction of G_k.

Analogously, we get an almost-$\mathrm{EXP}(p)$-time simulator $\mathcal{S}^*$ with $\mathcal{S}^*_k = S^*_{2,k}$ for *all* k that does not maintain a state with exception of its view.

It is left to show that $\mathcal{Z}^*$ and $\mathcal{S}^*$ are perfectly universal.

Since $\mathcal{Z}^*_k = S^*_{1,k}$ and $\mathcal{S}^*_k = S^*_{2,k}$, the pair $(\mathcal{Z}^*_k, \mathcal{S}^*_k)$ forms a nash-equilibrium of G_k. Because a player-I-strategy S^0_1 that always ends in a leaf corresponding to a 0-output of the environment will achieve $H_{G_k}(S^0_1, S'_2) = 0$ for any strategy S'_2, we can conclude that $H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^*_k) \geq H_{G_k}(S^0_1, \mathcal{S}^*_k) = 0$. Further, for any player-I-strategy S'_1, there is a strategy $-S'_1$ that always ends in a 0-leaf when S'_1 ends in a 1-leaf and vice versa. Then $H_{G_k}(S'_1, \mathcal{S}^*_k) = -H_{G_k}(-S'_1, \mathcal{S}^*_k)$. Let $|S'_1| = S'_1$ if $H_{G_k}(S'_1, \mathcal{S}^*_k) \geq 0$ and $|S'_1| := -S'_1$ otherwise. Together with the definition of a nash-equilibrium it follows that for all strategies S'_1 and S'_2 it is

$$|H_{G_k}(S'_1, \mathcal{S}^*_k)| = H_{G_k}(|S'_1|, \mathcal{S}^*_k) \leq |H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^*_k)| \qquad \text{and} \qquad |H_{G_k}(\mathcal{Z}^*_k, S'_2)| \geq |H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^*_k)|. \tag{6}$$

Let $\mathcal{Z}'$ and $\mathcal{S}'$ be arbitrary (not necessarily (p,p)-sized) environment and simulator, and let k be an arbitrary security parameter.

By the definition of the statistical distance, there is a family of functions $T_k : \Sigma^* \to \{0,1\}$ s.t.,

$$\Delta\big(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k)\big) = \Delta\big(T_k(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}(k)), T_k(\mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k))\big) \tag{7}$$

We now define the environment $\mathcal{Z}^T$ to behave as does $\mathcal{Z}'$, except that instead of giving an output $x \in \Sigma^*$, it outputs $T_k(x)$. $\mathcal{Z}^T$ has one-bit output. It then follows that

$$\Delta\big(T_k(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}(k)), T_k(\mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k))\big) = \Delta\big(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^T}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^T}(k)\big). \tag{8}$$

Further, we construct $\mathcal{Z}^p$ from $\mathcal{Z}^T$ as follows: Every message send and received is truncated to length at most p, at most p messages are sent and received from the protocol and at most p messages are sent and received from the adversary/simulator. Then $\mathcal{Z}^p$ is (p,p)-sized. Since π, ρ

[8] We could make the theorem slightly stronger at this point by giving a runtime-bound t, a size-bound s, and a bound r on the number of random bits used for protocol and adversary, since only the last two will appear exponentially in the overall running time of $\mathcal{Z}^*$ and $\mathcal{S}^*$. We have ommitted this lest the statement of the Theorem should get to complicated. But in cases where t is superpolynomially larger than s and r, redoing the calculations in this paragraph may give a lower runtime bound.

and $\mathcal{A}$ are p-sized, and $\mathcal{S}^*$ is (p,p)-sized, this only truncates outgoing data that is not read and ingoing data that may never be sent when communicating with π and $\mathcal{A}$ or with ρ and $\mathcal{S}^*$, so

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^T}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^T}(k)) = \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^p}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^p}(k)). \tag{9}$$

In an analogous fashion we convert $\mathcal{S}'$ to a (p,p)-sized simulator $\mathcal{S}^p$ and get

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}',\mathcal{Z}^*}(k)) = \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^p,\mathcal{Z}^*}(k)). \tag{10}$$

Therefore we can calculate

$$\begin{aligned}\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^p}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^p}(k)) &\overset{(5)}{=} |2H_{G_k}(\mathcal{Z}^p_k, \mathcal{S}^*_k)| \\ &\overset{(6)}{\le} |2H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^*_k)| \overset{(5)}{=} \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k))\end{aligned}$$

and then from this and (7,8,9) equation (1) follows with $\varepsilon = 0$. Further we have

$$\begin{aligned}\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^p,\mathcal{Z}^*}(k)) &\overset{(5)}{=} |2H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^p_k)| \\ &\overset{(6)}{\ge} |2H_{G_k}(\mathcal{Z}^*_k, \mathcal{S}^*_k)| \overset{(5)}{=} \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k))\end{aligned}$$

from which (2) follows with $\varepsilon = 0$ using (10). So $\mathcal{Z}^*$ and $\mathcal{S}^*$ are perfectly universal. □

Note that in the preceding proof, the only place where we used the fact that π, ρ and $\mathcal{A}$ are p-time and not just p-sized was in showing that the game tree of G_k can be constructed in EXP(p)-time. This again was only needed to get a bound on the complexity of $\mathcal{Z}^*$ and $\mathcal{S}^*$. Therefore we immediately have the following corollary:

Corollary 1. *Let p be a positive function in k. Let π and ρ be* p-sized *protocols, and let $\mathcal{A}$ be a* p-sized *adversary.*

Then there exist perfectly universal environment $\mathcal{Z}^*$ and simulator $\mathcal{S}^*$ for π, ρ and $\mathcal{A}$.

Both $\mathcal{Z}^*$ and $\mathcal{S}^*$ are $O(p)$-sized and do not maintain any state between invocations with exception of their respective views. $\mathcal{Z}^*$ has one-bit output.

Proof. Exactly as that of Theorem 2, with the calculation of the runtime of $\mathcal{Z}^*$ and $\mathcal{S}^*$ omitted. □

Finally, a further variant of Theorem 2 is the following, where we reduce the complexity of environment and simulator to exponential instead of almost-exponential time. The cost of this is that environment and simulator are now only statistically and not perfectly universal. However, for most purposes (in particular all corollaries in the next section) statistical universality is sufficient.

Corollary 2. *Let p be a positive time-constructible*[9] *function in the security parameter k. Let π and ρ be p-time protocols, and let $\mathcal{A}$ be a p-time adversary.*

Then there exist statistically universal environment $\mathcal{Z}^*_\approx$ and simulator $\mathcal{S}^*_\approx$ for π, ρ and $\mathcal{A}$.

Both $\mathcal{Z}^*_\approx$ and $\mathcal{S}^*_\approx$ run in EXP(p)*-time, are $O(p)$-sized, and do not maintain any state between invocations with exception of their respective view. $\mathcal{Z}^*_\approx$ has one-bit output.*

[9] A function f is called *time-constructible* if there is a deterministic multi-tape Turing machine that computes $f(k)$ in time $O(f(k))$. It is easy to see that all positive polynomials are time-constructible, and that with f also 2^f and cf (with $c \in \mathbb{N}$) are time-constructible.

Proof. Let $\mathcal{Z}^*$ and $\mathcal{S}^*$ be the perfectly universal environment and simulator for π, ρ and $\mathcal{A}$ given by Theorem 2. Now, since $\mathcal{Z}^*$ and $\mathcal{S}^*$ are almost-EXP(p)-time, there are time-constructible functions $f, g \in \mathrm{EXP}(p)$, s.t. the probability that the running time of $\mathcal{Z}^*$ or $\mathcal{S}^*$ exceeds $f(2k)$ or $g(2k)$, resp., is bounded by 2^{-k} where k denotes the security parameter. We now replace $\mathcal{Z}^*$ by a machine $\mathcal{Z}^*_{\approx}$ that simulates $\mathcal{Z}^*$, excepts that it terminates when $\mathcal{Z}^*$ runs more than $f(2k)$ steps. Similarly, we construct $\mathcal{S}^*_{\approx}$ simulating $\mathcal{S}^*$ and terminating after $g(2k)$ steps. The probability that $\mathcal{Z}^*_{\approx}$ or $\mathcal{S}^*_{\approx}$ behaves differently from $\mathcal{Z}^*$ or $\mathcal{S}^*$, resp., is bounded by 2^{-k}. Therefore all of the following are bounded by $2 \cdot 2^{-k}$:

$$\begin{aligned}
&\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*}(k), \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*_{\approx}}(k)), \qquad \Delta(\mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*_{\approx},\mathcal{Z}^*_{\approx}}(k))\\
&\Delta(\mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*_{\approx},\mathcal{Z}'}(k)), \qquad \Delta(\mathrm{EXEC}_{\rho,\mathcal{S}',\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}',\mathcal{Z}^*_{\approx}}(k)).
\end{aligned}$$

Combining this with the fact that (1) and (2) holds for $\mathcal{Z}^*$ and $\mathcal{S}^*$ and $\varepsilon = 0$, we get (by the triangle inequality of Δ) for all $\mathcal{Z}'$ and $\mathcal{S}'$:

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*_{\approx}}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*_{\approx},\mathcal{Z}^*_{\approx}}(k)) \geq \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}'}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*_{\approx},\mathcal{Z}'}(k)) - 4 \cdot 2 \cdot 2^{-k}$$

and

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*_{\approx}}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*_{\approx},\mathcal{Z}^*_{\approx}}(k)) \leq \Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}^*_{\approx}}(k), \mathrm{EXEC}_{\rho,\mathcal{S}',\mathcal{Z}^*_{\approx}}(k)) + 4 \cdot 2 \cdot 2^{-k}.$$

Since $4 \cdot 2 \cdot 2^{-k}$ is negligible, it follows that $\mathcal{Z}^*_{\approx}$ and $\mathcal{S}^*_{\approx}$ are statistically universal.

Further, since $\mathcal{Z}^*_{\approx}$ and $\mathcal{S}^*_{\approx}$ simulate at most $f(2k) \in \mathrm{EXP}(p)$ or $g(2k) \in \mathrm{EXP}(p)$ steps of $\mathcal{Z}^*$ or $\mathcal{S}^*$, resp., they are both EXP(p)-time. The other properties stated in the corollary are automatically inherited from $\mathcal{Z}^*$ and $\mathcal{S}^*$. □

The reader might ask whether to require bounded-size protocols and adversaries is necessary for the existence of universal environment or adversary. Concerning the protocols, [HU05a] gives an answer. They give a separating example for UC and specialised-simulator UC in the statistical case. It can be easily seen that for that example and—say—the dummy-adversary,[10] there is are no ε-universal environment and simulator for any $\varepsilon < \frac{1}{2}$. The seperating example consists of continuously polynomial protocols[11] which is one of the smallest classes of protocols that contains non-bounded-size protocols. The question however, whether a bounded-size adversary is necessary for the existence of a universal simulator, remains open.

5 Consequences

In this section we demonstrate some interesting consequences of the results about universal environments and simulators from the last chapter. Most noteworthy are the facts that with statistical security, the order of quantifiers (i.e., whether to use UC or specialised-simulator UC) is not relevant for bounded-size protocols (Corollary 4), that statistical specialised-simulator UC guarantees composition of bounded-size protocols (Corollary 5), and that we can w.l.o.g. assume exponential adversarial entities instead of unlimited ones for polynomial protocols (Corollary 3).

Prior to stating the first corollary, we need the following definition:

[10] With a sufficiently large bound on its communication complexity, cf. the discussion before Def. 4.

[11] Continuously polynomial protocols, as defined in [HMQU05] are—roughly spoken—protocols that run polynomially in the size of the input they receive from the environment and the adversary.

Definition 14 (Closed under combination). *Let M be a class of machines. We call M* closed under combination*, if for any two machines $A, B \in M$, there is a machine $C \in M$ that simulates A and B.*[12]

This property is vital for the application of the dummy-adversary technique, where we combine simulator and adversary into one machine at a given point. As is easy to see, the following classes of machines are closed under combination: polynomial-time and exponential-time with and without auxiliary input. The closedness property is often used in simulatability-related proofs, but is usually not explicitly stated, since it is obvious for polynomial-time machines.

We are now ready to state the first result of this section:

Corollary 3 (Exponential adversaries are "unlimited"). *Let p be a positive time-constructible function, and let π and ρ be p-time protocols.*

Then, let $C_\mathcal{Z}$ and $C_\mathcal{S}$ be classes of machines containing $\mathrm{EXP}(p)$*-time, let $C_\mathcal{S}$ be closed under combination, and let $C_\mathcal{A} \subseteq C_\mathcal{S}$ be a class containing* $\mathrm{poly}(p)$*-time.*

Then π implements ρ with respect to unlimited adversarial entities if and only if π implements ρ with respect adversaries in $C_\mathcal{A}$, environments in $C_\mathcal{Z}$ and simulators in $C_\mathcal{S}$.

This also holds for specialised-simulator UC, for one-bit specialised-simulator UC, and for bounded-risk UC.

In particular, if π and ρ are polynomial-time protocols, then we can choose $C_\mathcal{Z} = C_\mathcal{S} = C_\mathcal{A} =$ EXP*-time, i.e., security with respect to unlimited and to exponential-time adversarial entities coincide in this case.*

To show this corollary, we roughly proceed as follows: Using the universal environment and simulator we show that one-bit specialised-simulator UC with respect to the dummy-adversary, to environments in $C_\mathcal{Z}$ and to unlimited simulators (a very weak notion) implies bounded-risk UC with respect to the dummy-adversary, to unlimited environments and to simulators in $C_\mathcal{S}$ (a very strong notion). From this, we can then easily show that all notions the security notions appearing in the statement of the corollary are equivalent. The actual proof is not complicated, but rather lengthy due to the many different cases that are covered by the corollary:

Proof. We show something more general:

Let $C^1_\mathcal{Z} \subseteq C_\mathcal{Z}$ denote the set of environments in $C_\mathcal{Z}$ having one-bit output.

If π, ρ are p-sized, and there is a pair of statistically universal environment $\mathcal{Z}^*$ and simulator $\mathcal{S}^*$ for π, ρ and the p-dummy-adversary (as in Def. 4), s.t. $\mathcal{Z}^* \in C^1_\mathcal{Z}$, and $\mathcal{S}^* \in C_\mathcal{S}$, and $C_\mathcal{S}$ is closed under combination, and the p-dummy-adversary lies in $C_\mathcal{A} \subseteq C_\mathcal{S}$, then the following statements are equivalent:

(BR)	π implements ρ with respect to bounded-risk UC and unlimited adversarial entities.
(BRp)	…with respect to bounded-risk UC, the p-dummy-adversary, and unlimited environments and simulators.
(BRps)	…with respect to bounded-risk UC, the p-dummy-adversary, unlimited environments, and simulators in $C_\mathcal{S}$.
(BRas)	…with respect to bounded-risk UC, adversaries in $C_\mathcal{A}$, unlimited environments, and simulators in $C_\mathcal{S}$.

[12] We admit that this definition is somewhat informal. In the proof of the following Corollary 3, we use it only when showing implication (BRps) $\Rightarrow$ (BRas) using the dummy-adversary-technique. So the reader in need of more details may look into the proof. Or, instead of requiring the closedness of $\mathcal{Z}$, one could add (BRps) $\Rightarrow$ (BRas) to the premises of the corollary (i.e., explicitly require that the dummy-adversary-technique is applicable to simulators in $\mathcal{S}$), getting rid of this definition altogether. However, we believe that using this definition is helpful for the presentation of Corollary 3.

(BRazs)	...with respect to bounded-risk UC, adversaries in $C_{\mathcal{A}}$, environments in $C_{\mathcal{Z}}$, and simulators in $C_{\mathcal{S}}$.
(UC)	...with respect to UC and unlimited adversarial entities.
(UCazs)	...with respect to UC, adversaries in $C_{\mathcal{A}}$, environments in $C_{\mathcal{Z}}$, and simulators in $C_{\mathcal{S}}$.
(SS)	...with respect to specialised-simulator UC and unlimited adversarial entities.
(SSas)	...with respect to specialised-simulator UC, adversaries in $C_{\mathcal{A}}$, unlimited environments, and simulators in $C_{\mathcal{S}}$.
(SSazs)	...with respect to specialised-simulator UC, adversaries in $C_{\mathcal{A}}$, environments in $C_{\mathcal{Z}}$, and simulators in $C_{\mathcal{S}}$.
(OB)	...with respect to one-bit specialised-simulator UC and unlimited adversarial entities.
(OBpz)	...with respect to one-bit specialised-simulator UC, the p-dummy-adversary, environments in $C_{\mathcal{Z}}$ and unlimited simulators.
(OBazs)	...with respect to one-bit specialised-simulator UC, adversaries in $C_{\mathcal{A}}$, environments in $C_{\mathcal{Z}}$, and simulators in $C_{\mathcal{S}}$.

From these equivalences we get Corollary 3 as follows: The p-dummy-adversary $\mathcal{A}_p$ is poly(p)-time, so by Corollary 2 we see that $\mathcal{Z}^*$ and $\mathcal{S}^*$ are in EXP(poly(p)) = EXP(p)-time. Therefore $C_{\mathcal{Z}}$, $C_{\mathcal{S}}$, and $C_{\mathcal{A}}$ as in the statement of the corollary fulfil the conditions given above, so the equivalence (UC) $\Leftrightarrow$ (UCazs) (and (SS) $\Leftrightarrow$ (SSazs), (OB) $\Leftrightarrow$ (OBazs), and (BR) $\Leftrightarrow$ (BRazs) for specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC, resp.) holds and the corollary is proven.

We now proceed to show these equivalences. Our aim is to show all implications in the following diagram, from these the equivalence of (BR)–(OBazs) follows:

$$\begin{tikzcd}
 & \text{(SS)} \arrow[r, "T"] & \text{(OB)} \arrow[r, "T"] & \text{(OBpz)} \arrow[dd, "U"] & \text{(OBazs)} \arrow[l, "T"'] & & \text{(SSazs)} \arrow[ll, bend right, "T"'] \\
\text{(UC)} \arrow[ur, bend left, "T"] & & & & & \text{(UCazs)} \arrow[ul, "T"] & \\
 & \text{(BR)} \arrow[ul, bend left, "T"] & \text{(BRp)} \arrow[l, "D"] & \text{(BRps)} \arrow[l, "T"] \arrow[r, "D"] & \text{(BRas)} \arrow[r, "T"] \arrow[rr, bend right, "T"'] & \text{(BRazs)} \arrow[u, "T"] & \text{(SSas)} \arrow[uu, "T"']
\end{tikzcd} \tag{11}$$

Each of the implications marked with T is trivial using the definitions of UC, specialised-simulator UC and bounded-risk UC and the fact that $C_{\mathcal{A}}$ contains the p-dummy-adversary.

We are left to show the three implications (BRp) $\Rightarrow$ (BR), (BRps) $\Rightarrow$ (BRas), and (OBpz) $\Rightarrow$ (BRps). Of these, (BRp) $\Rightarrow$ (BR) and (BRps) $\Rightarrow$ (BRas) (marked with a D) are shown using a variation of the dummy-adversary-technique [Can05]. Of these two, we only give the proof for (BRps) $\Rightarrow$ (BRas), the implication (BRp) $\Rightarrow$ (BR) is shown completely analogously.

Assume therefore that (BRps) holds. Then by definition there are a negligible function μ and a simulator $\tilde{S}_p \in C_{\mathcal{S}}$, s.t. for any environment $\mathcal{Z}_{\mathcal{A}}$ and all $k \in \mathbb{N}$, it is

$$\Delta(\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}_{\mathcal{A}}}(k), \mathrm{EXEC}_{\rho,\tilde{\mathcal{S}}_p,\mathcal{Z}_{\mathcal{A}}}(k)) \le \mu(k) \tag{12}$$

where $\tilde{\mathcal{A}}_p$ denotes the p-dummy-adversary.

To show (BRas), it is sufficient to demonstrate that for any adversary $\mathcal{A} \in C_{\mathcal{A}}$ there is a simulator $\mathcal{S}(\mathcal{A}) \in C_{\mathcal{S}}$, s.t. for any environment $\mathcal{Z}$ and all $k \in \mathbb{N}$, it is

$$\Delta(\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k), \mathrm{EXEC}_{\rho,\mathcal{S}(\mathcal{A}),\mathcal{Z}}(k)) \le \mu(k). \tag{13}$$

For this, let the simulator $\mathcal{S}(\mathcal{A})$ be the machine simulating both $\mathcal{A}$ and $\tilde{\mathcal{S}}_p$ and rerouting all messages that $\mathcal{A}$ sends or expects from the protocol π through $\tilde{\mathcal{S}}_p$. Further, for any environment $\mathcal{Z}$, let the environment $\mathcal{Z}_\mathcal{A}$ denote the machine simulating $\mathcal{Z}$ and $\mathcal{A}$, and internally forwarding all communication between these, and forwarding the messages that $\mathcal{A}$ sends to and expects from the protocol π to the adversary/simulator (which is expected to be the p-dummy-adversary or its simulator). Clearly, since $\mathcal{A} \in C_\mathcal{A} \subseteq C_\mathcal{S}$ and $\tilde{\mathcal{S}}_p \in C_\mathcal{S}$, and since $C_\mathcal{S}$ is closed under combination, it is $\mathcal{S}(\mathcal{A}) \in C_\mathcal{S}$. Now, by construction we have that

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k) = \mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}_\mathcal{A}}(k) \qquad \text{and} \qquad \mathrm{EXEC}_{\rho,\mathcal{S}(\mathcal{A}),\mathcal{Z}}(k) = \mathrm{EXEC}_{\rho,\tilde{\mathcal{S}}_p,\mathcal{Z}_\mathcal{A}}(k).$$

Using this, from (12) follows (13), and (BRazs) is shown.

Now, only the implication (OBpz) $\Rightarrow$ (BRps) (marked with U in (11)) is missing. Here we finally use the existence of universal environment and simulator. Assume that (OBpz) holds. Then by the definition of one-bit specialised-simulator UC for any environment $\mathcal{Z} \in C^1_\mathcal{Z}$ there is a (possibly unlimited) simulator $\mathcal{S}_\mathcal{Z}$ and a negligible function $\mu_\mathcal{Z}$ s.t.

$$\Delta(\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}}(k), \mathrm{EXEC}_{\rho,\mathcal{S}_\mathcal{Z},\mathcal{Z}}(k)) \le \mu_\mathcal{Z}(k)$$

where $\tilde{\mathcal{A}}_p$ denotes the p-dummy-adversary. Since $\mathcal{Z}^* \in C^1_\mathcal{Z}$ and $\mathcal{S}^* \in C_\mathcal{Z}$ are ε-universal for some negligible function ε, it follows from Definition 13 and the inequality above that for any (possibly unlimited) environment $\mathcal{Z}'$

$$\begin{aligned}
&\Delta(\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}'}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}'}(k)) \\
&\le \Delta(\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}^*,\mathcal{Z}^*}(k)) + \varepsilon(k) \\
&\le \Delta(\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}}_p,\mathcal{Z}^*}(k), \mathrm{EXEC}_{\rho,\mathcal{S}_{\mathcal{Z}^*},\mathcal{Z}^*}(k)) + 2\varepsilon(k) \\
&\le \mu_{\mathcal{Z}^*}(k) + 2\varepsilon(k) =: \mu^*.
\end{aligned}$$

Since $\mathcal{S}^* \in C_\mathcal{S}$, and both $\mathcal{S}^*$ and the negligible function μ^* do not depend on $\mathcal{Z}'$, this implies that π implements ρ with respect to bounded-risk UC, the p-dummy-adversary, unlimited environments, and simulators in $C_\mathcal{S}$, i.e., (BRps) follows and the implication (OBpz) $\Rightarrow$ (BRps) is proven.

We have seen that all implications depicted in (11) hold, therefore (BR)–(OBazs) are equivalent and the theorem follows as discussed above. □

A question that can be asked at this point is in what respect these upper bounds on the running time of the adversarial entities are tight? We do not know the answer to that question, but a noticeably smaller bound would have interesting complexity-theoretic implications: If we can w.l.o.g. restrict to given class $C_\mathcal{Z}$ and $C_\mathcal{S}$ of environments $\mathcal{Z}$ and simulators $\mathcal{S}$, machines in these classes are able to solve any computational puzzle (like finding the preimage of one-way functions, or solving Dist-NP-complete problems [Lev86]) with non-negligible probability (in the case of the simulator even with overwhelming probability). This due to the fact that one can easily construct pairs π, ρ of protocols which (a) tell whether we find ourselves in the real or ideal protocol when getting a correct solution from $\mathcal{Z}$, or (b) hide that information from $\mathcal{Z}$ only when getting a correct solution from $\mathcal{S}$. Since unlimited machines can easily make use of these possibilities, machines in $C_\mathcal{Z}$ and $C_\mathcal{S}$ have to be able to make use of these, too.

The next interesting consequence of the results in Section 4 is given by the next corollary.

Corollary 4 (Equivalence of security notions). *UC, specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC with respect to unlimited adversarial entities coincide for bounded-size protocols.*

In the case of bounded-time protocols, these notions additionally coincide with UC, specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC with respect to Turing-unlimited adversarial entities and with respect to Turing-unlimited adversarial entities with auxiliary input.

To show this, we can almost completely reuse the proof of Corollary 3:

Proof. Let π and ρ be bounded-size protocols. Then there is some p, s.t. π and ρ are p-sized.

Let $C_{\mathcal{S}}$, $C_{\mathcal{Z}}$, and $C_{\mathcal{A}}$ be the set of all (unlimited) machines. Then by Corollary 1 there are perfect universal environment $\mathcal{Z}^* \in C_{\mathcal{Z}}$ and simulator $\mathcal{S}^* \in C_{\mathcal{S}}$ for π, ρ and the p-dummy-adversary. Trivially, $C_{\mathcal{S}}$ is closed and $C_{\mathcal{A}} \subseteq C_{\mathcal{S}}$ contains the p-dummy-adversary. So the conditions at the beginning of the proof of Corollary 3 are fulfilled. The equivalences (UC) $\Leftrightarrow$ (SS) $\Leftrightarrow$ (OB) $\Leftrightarrow$ (BR) then give the equivalence of UC, specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC with respect to unlimited adversarial entities.

Let now π and ρ be additionally bounded-time. Then there is some (time-constructible[13]) p, s.t. π and ρ are p-time. Since bounded-time implies bounded-size, the equivalences shown above still hold. Now, let $C_{\mathcal{S}}$, $C_{\mathcal{Z}}$, and $C_{\mathcal{A}}$ be the set of all Turing-unlimited machines. Obviously, the conditions for Corollary 3 are fulfilled, so UC with respect to unlimited adversarial entities is equivalent to UC with respect to Turing-unlimited adversarial entities. This also holds for specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC. By letting $C_{\mathcal{S}}$, $C_{\mathcal{Z}}$, and $C_{\mathcal{A}}$ be the set of all Turing-unlimited adversarial entities with auxiliary input, we analogously get the equivalence between UC with respect to unlimited adversarial entities and UC with respect to Turing-unlimited adversarial entities with auxiliary input. The same holds for specialised-simulator UC, one-bit specialised-simulator UC, and bounded-risk UC. So in the case of bounded-time protocols, all notions given in the statement of the corollary are equivalent. □

The restriction to bounded-size protocols in the above corollary is necessary, since in [HU05a] it was shown that UC and specialised-simulator UC differ in the case of statistical security (using the separating example already mentioned at the end of Section 4). The separating protocols given there were continuously polynomial protocols[14].

Finally, the following is a simple consequence of the above corollaries:

Corollary 5 (Universal Composition Theorem).

(i) Specialised-simulator UC with respect to unlimited adversarial entities guarantees polynomially-bounded general composability [Lin03] of bounded-size protocols.
(ii) Specialised-simulator UC with respect to Turing-unlimited adversarial entities with or without auxiliary input guarantees polynomially-bounded general composability of bounded-time protocols.
(iii) Specialised-simulator UC with respect to exponential adversarial entities with or without auxiliary input guarantees polynomially-bounded general composability of polynomial-time protocols.

The same holds for one-bit specialised-simulator UC.

[13] p can be chosen to be time-constructible by the following argument: Let M be the deterministic Turing machine that upon input k explores all computational paths of π and ρ for security parameter k for all possible inputs. Let $p(k)$ denote the running time of M on input k. Then p is finite, time-constructible and bounds the runtime of π and ρ.

[14] Continuously polynomial protocols, as defined in [HMQU05] are—roughly spoken—protocols that run polynomially in the size of the input they receive from the environment and the adversary.

Proof. It is known from [Can05] that computational UC with respect to polynomial adversarial entities guarantees polynomially-bounded general composition of polynomial protocols. From the proof of the Universal Composition Theorem in [Can05] it is easy to see that statistical UC (i.e., UC as used in this paper) with respect to unlimited adversarial entities guarantees polynomially-bounded general composition of all protocols.

Case (i) is shown as follows: Corollary 4 guarantees that in the case of bounded-size protocols, UC with respect to unlimited adversarial entities and specialised-simulator UC with respect to unlimited adversarial entities coincide. Since in the case of UC polynomially-bounded general composition is guaranteed, it is therefore also guaranteed in the case of specialised-simulator UC (one needs to note that the composition of bounded-size protocols is also bounded-size).

Case (ii) is shown completely analogously using the other cases of Corollary 4.

For Case (iii) Corollary 3 states the equivalence of specialised-simulator UC with respect to exponential and with respect to unlimited adversarial entities. Using this equivalence, Case (iii) follows from Case (i).

For one-bit specialised-simulator UC the proof is completely analogous (using the corresponding cases in Corollaries 3 and 4). □

In fact, some cases of Corollary 5 were already shown in greater generality in [HU05b] using a completely different approach. They showed that in the case of unlimited adversarial entities and of Turing-unlimited ones with auxiliary input, polynomially-bounded general composition was guaranteed by specialised-simulator UC (without the restriction to bounded-size protocols). However, the cases of Turing-unlimited adversarial entities without auxiliary input and of exponential ones are not covered by the results in [HU05b], and neither is the case of one-bit specialised-simulator UC. Further, our proofs allow to give much tighter runtime bounds for the simulator.

A question that might arise is whether the requirements on the complexity of the adversarial entities are indeed necessary. A partial answer is given by [HU05b] who showed that polynomially-bounded general composition of polynomial protocols is not guaranteed by computational specialised-simulator UC with respect to polynomial adversarial entities.[15]

6 Conclusions

We showed that for bounded-size protocols, there are universal environments and simulators in the sense that the universal environment is the "best response" to the universal simulator, and the universal simulator is the "best response" to the universal environment. Then one can w.l.o.g. restrict the security definitions to such universal environments and simulators. Further, we could give upper bounds on the complexity of the universal environment/simulator.

Using the existence of universal environment/simulator, we were able to show that in the case of statistical security UC and specialised-simulator UC coincide for bounded-size protocols (i.e., that it is not important whether the simulator depends on the environment or vice versa). In particular, this guarantees universal composition of bounded-size protocols in the case of specialised-simulator UC (a result also shown in [HU05b] using a completely different approach). We also examined to what extent this holds for more restricted classes of adversarial entities.

Further, we introduced a new variant of the UC security notion, bounded-risk UC. Security with respect to bounded-risk UC guarantees that concrete security bounds can be given for concrete security parameters. We showed that for bounded-size protocols, bounded-risk UC coincides with UC, too.

[15] Under the assumption, that enhanced trapdoor permutations [Gol04] and time-lock puzzles [RSW96, HU05a] exist.

Additionally, we could show, that when considering polynomial protocols, we can w.l.o.g. assume the adversarial entities to be exponential instead of unlimited.

However, some open questions remain: The most obvious one is to what extent the different runtime-bounds given in this paper are tight.

An important class of protocols is that of continuously polynomial protocols [HMQU05]. These are protocols that do not have an a-priori runtime bound in the security parameter, but instead are polynomial in the input received from environment and adversary. Many protocols given in the literature are indeed continuously polynomial and not polynomial, because they lack an explicit bound on the number of invocations (e.g., a secure message transmission protocol or a public key encryption that can handle multiple messages). In general, there is no universal environment/simulator for continuously polynomial protocols (cf. the discussion at the end of Section 4), but an interesting open question is which of the results in Section 5 still hold (using a different proof, of course).

Finally, in [BOM04, Unr04] the idea of simulatable security (as in the UC framework [Can05] and the RS framework [BPW04b]) has been adapted to the setting of quantum cryptography. It would be interesting to know, which of the results of this paper hold for bounded-size *quantum* protocols (using results from quantum game theory).

Acknowledgements. We thank Dennis Hofheinz and Jörn Müller-Quade for valuable discussions. This work was funded by the EC project ProSecCo under IST-2001-39227.

A Postponed material

A.1 Proofs and references for Figure 1

We give a short overview over the results used for Figure 1. Some of the arrows have straightforward proofs but have not been shown in the literature (to our knowledge). In these cases, we provide a short proof sketch. We use the same abbreviations as in Figure 1.

The implication comp. UC $\Rightarrow$ comp. poly. is shown in [Can05]. The implication comp. poly. $\Rightarrow$ comp. O(1) is trivial by definition, and comp. spec. $\Leftrightarrow$ comp. $O(1)$ was shown in [Lin03]. So we have comp. UC $\Rightarrow$ comp. poly. $\Rightarrow$ comp. spec. $\Leftrightarrow$ comp. $O(1)$. The proofs of [Can05] and [Lin03] easily generalise to the case of perfect and statistical security, so we also have perf. UC $\Rightarrow$ perf. poly. $\Rightarrow$ perf. spec. $\Leftrightarrow$ perf. $O(1)$ and stat. UC $\Rightarrow$ stat. poly. $\Rightarrow$ stat. spec. $\Leftrightarrow$ stat. $O(1)$.

In [HU05a] it was shown that perf. spec. $\Rightarrow$ perf. UC, the equivalence of the four cases in the first row (perf...) of Figure 1 follows.

Corollary 4 gives us stat. spec $\Rightarrow$ stat. UC, the equivalence of the four cases in the second row (stat...) of Figure 1 follows. Note that the implication stat. spec $\Rightarrow$ stat. UC was also shown in [HU05b] without the restriction to bounded-size protocols, however, the complexity-bounds given for the simulator in our work are much tighter.

The implication perf. UC to stat. UC is trivial by definition, the other implications from the first to the second row follow using the equivalences shown above. The non-implication stat. UC $\not\Rightarrow$ perf. UC is shown using the following easy counterexample: Let π be a protocol that never gives output, and ρ a protocol that gives output with probability 2^{-k} when queried by the environment. Clearly π implements ρ with respect to statistical UC, but not with respect to perfect UC. The other non-implications from the second to the first row follow.

The non-implication comp. spec. $\not\Rightarrow$ comp. poly. is shown in [HU05b] (assuming the existence of enhanced trapdoor permutations [Gol04] and time-lock puzzles [RSW96, HU05a]). In [HU05a], a counterexample was given to show comp. spec. $\not\Rightarrow$ comp. UC (assuming the existence of time-lock puzzles). It is easy to see, that the protocols in their counterexample are not only secure with respect to specialised-simulator UC, but do also compose concurrently, i.e., satisfy the polynomially-bounded general composability. So we have comp. poly. $\not\Rightarrow$ comp. UC (assuming the existence of time-lock puzzles). We have now shown all the arrows in the third row.

Assume the existence of weakly-verifiable computational puzzles that are hard even in the presence of auxiliary input (as defined in [CHS05]; just *puzzles* in the following text).[16] To see that comp. UC. $\not\Rightarrow$ stat. UC. consider the following counterexample: Both π and ρ send a puzzle to the environment $\mathcal{Z}$ upon first activation. When receiving a solution from $\mathcal{Z}$, π outputs whether it is correct or wrong. ρ always outputs that it is wrong. Clearly, π does not implement ρ with respect to statistical UC, since an unbounded environment can distinguish by solving the puzzles. However, in the computational case, the environment cannot solve the puzzles and is therefore unable to distinguish. So, comp. UC. $\not\Rightarrow$ stat. UC., and the other non-implications from the third to the second row follow.

To see that stat. spec. $\not\Rightarrow$ comp. spec. consider the following counterexample: When activated by the adversary or simulator, both π and ρ give a puzzle to the adversary/simulator. When receiving a solution, ρ outputs whether it is correct or not to the environment. When π receives a solution, it always outputs that it is correct. Clearly, an unbounded simulator can make the real and ideal model indistinguishable by always giving correct solutions. However, a computationally-limited simulator cannot solve the puzzles, so it cannot mimic the behaviour of an adversary that gives random solutions. So π implements ρ with respect to statistical specialised-simulator UC, but not to computational specialised-simulator UC, therefore we have stat. spec. $\not\Rightarrow$ comp. spec.. The other non-implications from the second to the third row follow.

A.2 An auxiliary lemma

Lemma 1 (Dice-throwing algorithm). *There is an algorithm that runs in almost-polynomial-time in the length of its input and achieves the following:*

Given a list of rational numbers $a_1, \dots, a_n$ with $\sum_i a_i = 1$ as input, it outputs i with probability a_i. (Rational numbers are given as nominator/denominator-pair, the length of a rational number is therefore the added lengths of nominator and denominator.)

Proof. Let $(r_i) \in \{0,1\}^{\mathbb{N}}$ denote the content of the random tape of the algorithm. Let further $A_i := \sum_{\mu=1}^{i} a_\mu$, $m_t := \sum_{i=1}^{t} r_i 2^{-i}$, $m'_t := m_t + 2^{-t}$. Then the algorithm proceeds as follows: For each $t = 1, 2, \dots$ (in that order) it checks whether there is a $i \in \{1, \dots, n\}$ s.t., $m_t, m'_t \in [A_{i-1}, A_i]$. If so, i is output and the algorithm terminates.

We first investigate the correctness of the algorithm. Let $m_\infty := \sum_{i=1}^{\infty} r_i 2^{-i}$. Then m_∞ is uniformly distributed on $[0,1]$. So for any $i = 1, \dots, n$ the probability that $m_\infty \in (A_{i-1}, A_i)$ is a_i. Further, $m_\infty \in [m_t, m'_t]$ for all t and $m_t, m'_t \to m_\infty$. So if $m_\infty \in (A_{i-1}, A_i)$, $m_t, m'_t \in [A_{j-1}, A_j]$ may never hold for any $j \neq i$, and because $m_t, m'_t \to m_\infty$ it will eventually hold for $j = i$. So with probability a_i, the algorithm outputs i.

We now show that the algorithm runs indeed in almost-polynomial-time in the length of its input. Let l denote that length. W.l.o.g. we can assume $l \geq n + 1$. Then, there is a polynomial p, s.t. for each cycle of the main loop (i.e., for each t) the algorithm runs at most $p(t + l)$

[16] These are puzzles of the following kind: A polynomial-time machine can generate these puzzles, and can verify the solutions. Another polynomial-time machine can find the solution. There always is a solution, i.e., an unbounded machine can always solve the puzzle. This is a very weak assumption, which is equivalent to the existence of hard-on-average problems in DistNP (as defined in [Lev86]) and is implied by the existence of one-way functions. For more information, see [CHS05].

steps. Let t_0 denote the t at which the algorithm terminates. If $t_0 > i$ for some i, then $m_\infty \in [A_i - 2^{-i}, A_i + 2^{-i}]$ for some $i = 0, \dots, n$. The probability of this is bounded by $(n+1)2^{-i+1}$, so $P(t_0 > i) \leq P(t_0 > \lfloor i \rfloor) \leq (n+1)2^{-\lfloor i \rfloor + 1} \leq (n+1)2^{-i+2}$ even for non-integral i. The total number of steps T is bounded by $\sum_{t=1}^{t_0} p(t+l)$. Therefore there is a strictly monotonely increasing polynomial p' s.t., $T < p'(t_0 + l)$, and we can further construct a polynomial p'' fulfilling $p''(l - \log\varepsilon) > p'(l - \log\varepsilon + \log l + 2)$ for all $l > 0$ and $\varepsilon \leq 1$. Then it is $P(T > p''(l - \log\varepsilon)) \leq P(T \geq p'(l - \log\varepsilon + \log l + 2)) \leq P(p'(t_0 + l) \geq p'(l - \log\varepsilon + \log l + 2)) \leq P(t_0 \geq -\log\varepsilon + \log l + 2) = (n+1)2^{\log\varepsilon - \log l} = \frac{n+1}{l}\varepsilon \leq \varepsilon$. Therefore the algorithm is almost-p''-time with $p'' \in$ poly. □

References

[BOM04] M. Ben-Or and D. Mayers. General security definition and composability for quantum & classical protocols, September 2004. Online available at http://xxx.lanl.gov/abs/quant-ph/0409062.

[BPW04a] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In Moni Naor, editor, *Theory of Cryptography, Proceedings of TCC 2004*, number 2951 in Lecture Notes in Computer Science, pages 336–354. Springer-Verlag, 2004. Online available at http://www.zurich.ibm.com/security/publications/2004/BaPfWa2004MoreGeneralComposition.pdf.

[BPW04b] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Secure asynchronous reactive systems. IACR ePrint Archive, March 2004. Online available at http://eprint.iacr.org/2004/082.ps.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. Full version online available at http://www.eccc.uni-trier.de/eccc-reports/2001/TR01-016/revisn01.ps.

[Can05] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, January 2005. Online available at http://eprint.iacr.org/2000/067.ps.

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *Theory of Cryptography, Proceedings of TCC 2005*, Lecture Notes in Computer Science, pages 17–33. Springer-Verlag, 2005. Available as IACR ePrint 2004/329.

[GLS88] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. Online available at http://www-gdz.sub.uni-goettingen.de/cgi-bin/digbib.cgi?PPN330894919.

[Gol04] Oded Goldreich. *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, May 2004. Previous version online available at http://www.wisdom.weizmann.ac.il/~oded/frag.html.

[HMQU05] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Polynomial runtime in simulatability definitions. In *18th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2005*, pages 156–169. IEEE Computer Society, 2005. Online available at http://iaks-www.ira.uka.de/home/unruh/publications/hofheinz05polynomial.html.

[HU05a] Dennis Hofheinz and Dominique Unruh. Comparing two notions of simulatability. In *Theory of Cryptography, Proceedings of TCC 2005*, Lecture Notes in Computer Science, pages 86–103. Springer-Verlag, 2005. Online available at http://iaks-www.ira.uka.de/home/unruh/publications/hofheinz05comparing.html.

[HU05b] Dennis Hofheinz and Dominique Unruh. Simulatable security and concurrent composition, September 2005. Submitted to IEEE Symposium on Security and Privacy 2006.

[KM92] Daphne Koller and Nimrod Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4(4):528–552, October 1992. Online available at http://theory.stanford.edu/~megiddo/pdf/recall.pdf (without figures).

[Kuh56] Harold William Kuhn. Extensive games and the problem of information. In Kenneth J. Arrow and Harold William Kuhn, editors, *Contributions to the theory of games*, volume 2, pages 193–216. Princeton University Press, 2 edition, 1956.

[Lev86] Leonid A Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. Online available at `http://www.cs.bu.edu/fac/lnd/research/dvi/rp.dvi`.

[Lin03] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2003*, pages 394–403. IEEE Computer Society, 2003. Online available at `http://www.research.ibm.com/people/l/lindell/PAPERS/gc-uc.ps.gz`.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy, Proceedings of SSP 2001*, pages 184–200. IEEE Computer Society, 2001. Full version online available at `http://eprint.iacr.org/2000/066.ps`.

[Ras89] Eric Rasmusen. *Games and Information*. Basil Blackwell, 1989.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology, February 1996. Online available at `http://theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps`.

[Unr04] Dominique Unruh. Simulatable security for quantum protocols. Preprint on quant-ph/0409125, september 2004. Online available at `http://arxiv.org/ps/quant-ph/0409125`.