On adaptive vs. non-adaptive security of multiparty protocols

Ran Canetti*, Ivan Damgaard, Stefan Dziembowski**, Yuval Ishai***, and Tal Malkin†

Abstract. Security analysis of multiparty cryptographic protocols distinguishes between two types of adversarial settings: In the non-adaptive setting, the set of corrupted parties is chosen in advance, before the interaction begins. In the adaptive setting, the adversary chooses who to corrupt during the course of the computation. We study the relations between adaptive security (i.e., security in the adaptive setting) and non-adaptive security, according to two definitions and in several models of computation. While affirming some prevailing beliefs, we also obtain some unexpected results. Some highlights of our results are:

- According to the definition of Dodis-Micali-Rogaway (which is set in the information-theoretic model), adaptive and non-adaptive security are equivalent. This holds for both honest-but-curious and Byzantine adversaries, and for any number of parties.
- According to the definition of Canetti, for honest-but-curious adversaries, adaptive security is equivalent to non-adaptive security when the number of parties is logarithmic, and is strictly stronger than non-adaptive security when the number of parties is super-logarithmic. For Byzantine adversaries, adaptive security is strictly stronger than non-adaptive security, for any number of parties.

1 Introduction

Security analysis of cryptographic protocols is a delicate task. A first and crucial step towards meaninful analysis is coming up with an appropriate definition of security of the protocol problem at hand. Formulating good definitions is non-trivial: They should be compehensive and stringent enough to guarantee security against a variety of threats and adversarial behaviors. On the other hand, they should be as simple, workable, and as permissive as possible, so as to facilitate design and analysis of secure protocols, and to avoid unnessecary requirements.

Indeed, in contrast with the great advances in constructing cryptographic protocols for a large variety of protocol problems, formalizing definitions of security for crypographic protocol problems has been progressing more slowly. The

* IBM Watson.

** Aarhus University, BRICS (Basic Research in Computer Science, Center of the Danish National Research Foundation).

*** DIMACS and AT&T. Work partially done while the author was at the Technion and while visiting IBM Watson.

† AT&T Labs – Research. Work partially done while the author was at MIT and while visiting IBM Watson.

first protocols appearing in the literature use only intuitive and ad-hoc notions of security, and rigorous security analysis was virtually non-existent. Eventually, several general definitions of security for cryptographic protocols have appeared in the literature. Most notable are the works of Goldwasser and Levin [GL90], Micali and Rogaway [MR91], Beaver [B91], Canetti [C00] and Dodis and Micali [DM00] (that concentrate on the task of *secure function evaluation* [Y82,Y86,GMW87]), and Pfitzmann and Waidner [PW94], Pfitzmann Schunter and Waidner [PSW00], and Canetti [C00a] (that discuss general reactive tasks). In particular, only recently do we have precise and detailed definitions that allow rigorous study of "folklore beliefs" regarding secure protocols.

This work initiates a comparative study of notions of security, according to different definitions. We concentrate on secure function evaluation, and in particular the following aspect. Adversarial behavior of a computational environment is usually modelled via a single algorithmic entity, the adversary, the capabilities of which represent the actual security threats. Specifically, in a network of communicating parties the adversary is typically allowed to control (or, corrupt) some of the parties. Here the following question arises: How are the corrupted parties chosen? One standard model assumes that the set of corrupted parties is fixed before the computation starts. This is the model of non-adaptive adversaries. Alternatively, the adversary may be allowed to corrupt parties during the course of the computation, when the identity of each corrupted party may be based on the information gathered so far. We call such adversaries adaptive.

Indeed, attackers in a computer network (hackers, viruses, insiders) may break into computers during the course of the computation, based on partial information that was already gathered. Thus the adaptive model seems to better represent realistic security threats, and so provide a better security guarantee. However, defining and proving security of protocols is considerably easier in the non-adaptive model. One quintessential example for the additional complexity of guaranteeing adaptive security is the case of using encryption to transform protocols that assume ideally secure channels into protocols that withstand adversaries who hear all the communication. In the non-adaptive model standard Chosen-Ciphertext-Attack secure encryption [DDN91,CS98,S99] (or even plain semantically secure encryption [GM84], if used appropriately) is sufficient. To obtain adaptively secure encryption, it seems that one needs to either trust data erasures [BH92], or use considerably more complex constructs [CFGN96,B97,DN00].

Clearly, adaptive security implies non-adaptive security, under any reasonable definition of security. However, is adaptive security really a stronger notion than non-adaptive security? Some initial results (indicating clear separation in some settings) are provided in [CFGN96]. On the other hand, it is a folklore belief that in an "information theoretic setting" adaptive and non-adaptive security should be equivalent. Providing more complete answers to this question, in several models of computation, is the focus of this work. While some of our results affirm common beliefs, other results are quite surprising, and may considerably simplify the design and analysis of protocols.

Models of computation. We study the additional power of adaptive adversaries in a number of standard adversary models, and according to two definitions (the definition of Dodis, Micali, and Rogaway [MR91,DM00], and that of Canetti [C00]). To develop the necessary terminology for presenting our results let us very shortly outline the structure of definitions of security of protocols. (The description below applies to both definitions. The [MR91,DM00] definition imposes some additional requirements, sketched in a later section.)

As mentioned above, both definitions concentrate on the task of Secure Function Evaluation. Here the parties wish to jointly evaluate a given function at a point whose value is the concatenation of the inputs of the parties. In a nutshell, protocols for secure function evaluation are protocols that "emulate" an *ideal process* where all parties privately hand their inputs to an imaginary trusted party who privately computes the desired results, hands them back to the parties, and vanishes. A bit more precisely, it is required that for any adversary $\mathcal{A}$, that interacts with parties running a secure protocol π and induces some global output distribution, there exists an "ideal-process" adversary $\mathcal{S}$, that manages to obtain essentially the same global output distribution *in the ideal process.* The global output contains the adversary's output (which may be assumed to be his entire view of the computation), together with the identities and outputs of the uncorrupted parties. (Adversary $\mathcal{S}$ is often called a simulator, since it typically operates by simulating a run of $\mathcal{A}$.) The following parameters of the adversarial models turn out to be significant for our study.

Adversarial activity: The adversary may be either passive (where even corrupted parties follow the prescribed protocol, and only try to gather additional information), or active, where corrupted parties are allowed to arbitrarily deviate from their protocol. Passive (resp., active) adversaries are often called honest-but-curious (resp., Byzantine).

Number of players: We distinguish between the case of a small number of players, where n, the number of players, is $O(\log k)$, and a large number of players, where n is $\omega(\log k)$. (Here k is the security parameter.)

Complexity of adversaries: We consider three cases. Information-Theoretic (IT) security does not take into account any computational complexity considerations. That is, both adversaries $\mathcal{A}$ and $\mathcal{S}$ have unbounded resources regardless of each other's resources. Universal security allows $\mathcal{A}$ unbounded resources, but requires $\mathcal{S}$ to be efficient (i.e., expected polynomial) in the complexity of $\mathcal{A}$. Computational security restricts both $\mathcal{A}$ and $\mathcal{S}$ to expected polynomial time (in the security parameter). Note that universal security implies both IT security and computational security (all other parameters being equal). However, IT security and computational security are incomparable. See [C00] for more discussion on the differences between these notions of security and their meaning.

Quality of emulation: We consider either perfect emulation (where the output distributions of the real-life computation and of the ideal process must be identically distributed), or statistical emulation (where the output distri-

butions shuld be statistically indistinguishable), or computational emulation (where the output distributions shuld be computationally indistinguishable).

The rest of the Introduction overviews the state of affairs regarding the added power of adaptivity, as discovered by our investigation. We do not attempt here to explain "why" things are as they are. Such (inevitably subjective) explanations require more familiarity with the definitions and are postponed to the body of the paper.

Our results: Canetti's definition. This definition is stated for several models of computation. We concentrate by default on the *secure channels* model, where the communication channels are perfectly secret and *universal security* is required. The same results hold also for the *computational* setting, where the adversary sees all communication but is restricted to polynomial time. Finally, we also consider a weaker variant of this definition, not considered in [C00], where only IT security is required (and the communication channels are secure).

The most distinctive parameter here seems to be whether the adversary is active or passive. If the adversary is active (i.e., Byzantine) then adaptive security is strictly stronger than non-adaptive security, regardless of the values of all other parameters. We show this via a protocol for three parties, that is non-adaptively universally secure with perfect emulation, but adaptively *in*secure, even if the adversary is computationally bounded and we are satisfied with computational emulation. This is the first such example involving only a constant number of players, for *any* constant.

In the case of passive adversaries the situation is more involved. Out of the nine settings to be considered (IT, universal, or computational security, with perfect, statistical, or computational emulation), we show that for one – IT security and perfect emulation – adaptive and non-adaptive security are equivalent, for any number of players. In all other eight settings we show that, roughly speaking, adaptive security is equivalent to non-adaptive security when the number of players is small, and is strictly stronger when the number of players is large. We elaborate below.

For a large number of players, it follows from an example protocol shown in [CFGN96] that for statistical or computational emulation, adaptive security is strictly stronger than non-adaptive security. We show separation also for perfect emulation, where universal or computational security is required. We complete the picture by showing that for a small number of players, and perfect emulation, adaptive and non-adaptive security are equivalent. Equivalence holds even in the case of statistical or computational emulation, if n is $O(\log k / \log\log k)$. (Notice that there is a small gap between this equivalence result and the known separating example for $n \in \omega(\log k)$. To close this gap, we also show that if one relaxes slightly the demands to the complexity of simulators and allows them to be expected polynomial time *except with negligible probability*, then this gap can be closed: equivalence holds for all $n \in O(\log k)$. In many cases, this definition of "efficient simulation" seems to be as reasonable as the standard one.)

Equivalence of adaptive and non-adaptive security for the case of passive adversaries and a small number of players is very good news: Many protocol prob-

lems (for instance, those related to threshold cryptography) make most sense in a setting where the number of parties is *fixed.* In such cases, when concentrating on passive adversaries, adaptivity comes "for free", which significantly simplifies the construction and analysis of these protocols.

Our results: Dodis-Micali-Rogaway definition. This definition holds for the *secure channels* setting only. It is incomparable to the definition of [C00]: On the one hand, it makes a number of additional requirements. On the other hand, only IT security is required. Here, to our surprise, adaptive and non-adaptive security turn out to be equivalent, even for active adversaries, and regardless of the number of players.

Two properties of the Dodis-Micali-Rogaway definition are essential for our proof of equivalence to work. The first is that only IT security is required. The second property may be roughly sketched as follows. It is required that there exists a stage in the protocol execution where all the parties are "committed" to their contributed input values; this stage must occur strictly before the stage where the output values become known to the adversary. (In order to formally state this requirement one needs to make some additional technical restrictions, amounting to what is known in the jargon as "one-pass black-box simulation". See more details within.)

Organization. Section 2 presents our results relating to the definition of [C00]. Section 3 presents our results relating to the definition of Dodis-Micali-Rogaway.

2 Adaptivity vs. Non-adaptivity in the definition of Canetti

This section describes our results relative to the [C00] definition of security. The main aspects of the definition that we will rely on were shortly described in the Introduction. A more detailed overview is deleted for lack of space and appears in the full version of this work [CDDIM01]. Section 2.1 shows a separating example for the case of active adversary, Section 2.2 describes separating examples for passive adversary and a large number of players, Section 2.3 proves the equivalence for passive adversaries and a small number of players, and Section 2.4 shows the equivalence for passive adversaries in the setting of IT security and perfect emulation.

2.1 Separation for active adversaries

This section shows that adaptive and non-adaptive security are not equivalent in the case of active adversaries, for all settings considered here: information-theoretic, universal, and computational security, with perfect, statistical, or computational emulation. This is proved by an example of a simple protocol for secure function evaluation which is non-adaptively secure, but adaptively insecure, in all above settings.

Our protocol involves three players D, R_1, R_2, where R_1, R_2 have no input, and D's input consists of two bits $s_1, s_2 \in \{0,1\}$. The function f_{act} to be computed is the function that returns no output for D, s_1 for R_1, and s_2 for R_2. The *adversary structure* $\mathcal{B}$ (the collection of player subsets that can be corrupted) contains all subsets of $\{D, R_1\}$, namely the only restriction is that R_2 cannot be corrupted. The protocol π_{act} proceeds as follows.

1. D sends s_1 to R_1.
2. D sends s_2 to R_2.
3. Each R_i outputs the bit that was sent to it by D, and terminates. D outputs nothing and terminates.

Claim 1 *The protocol π_{act} non-adaptively, perfectly emulates f_{act} with universal security, against active adversary structure $\mathcal{B}$.*

Proof. Consider a non-adaptive real-life adversary $\mathcal{A}$ that corrupts D. The ideal-process simulator $\mathcal{S}$ proceed as follows. $\mathcal{S}$ corrupts D in the ideal model, and provides $\mathcal{A}$ with the inputs s_1, s_2 of D. $\mathcal{A}$ generates s'_1 to be sent to R_1 and s'_2 to be sent to R_2. $\mathcal{S}$ gives s'_1, s'_2 to the trusted party as D's input, outputs $\mathcal{A}$'s output, and terminates. It is easy to see that the global output generated by $\mathcal{S}$ in the ideal model is identical to the global output with the real-life $\mathcal{A}$.

The above simulator can be easily modified for the case that $\mathcal{A}$ breaks into both D and R_1 (here $\mathcal{S}$ may hand in to the trusted party $0, s'_2$ as the input of D, where s'_2 is the message prepared by $\mathcal{A}$ to be sent to R_2).

Finally, consider $\mathcal{A}$ that corrupts only R_1. The simulator $\mathcal{S}$ proceeds as follows. $\mathcal{S}$ corrupts R_1 in the ideal model, hands the empty input to the trusted party, and obtains the output s_1 in the ideal model. $\mathcal{S}$ then hands s_1 to $\mathcal{A}$ as the message that was sent from D to R_1, outputs $\mathcal{A}$'s output, and terminates. Again it is easy to see that the global output generated by $\mathcal{S}$ is identical to the global output with $\mathcal{A}$.

Claim 2 *The protocol π_{act} is adaptively insecure for evaluating the function f_{act}, with either universal, IT or computational security, against active adversary structure $\mathcal{B}$.*

Proof. We show an adaptive efficient real life adversary $\mathcal{A}$, such that there is no (even computationally unbounded) adaptive ideal-model adversary (simulator) $\mathcal{S}$ that can emulate the global view induced by $\mathcal{A}$ (even if the emulation is only required to be computational). Intuitively, the goal of our adversary is to ensure that whenever $s_1 = s_2$, R_2 will output 0, whereas we do not care what happens in other cases. $\mathcal{A}$ starts by corrupting R_1 and receiving s_1 in the first stage of the protocol. If $s_1 = 0$, $\mathcal{A}$ terminates. If $s_1 = 1$, $\mathcal{A}$ corrupts D and sends $s'_2 = 0$ to R_2 in the second stage of the protocol.

To prove that this $\mathcal{A}$ cannot be simulated in the ideal world, note that in the real world, $\mathcal{A}$ never corrupts D when D's input is $s_1 = s_2 = 0$, but always corrupts D when D's input is $s_1 = s_2 = 1$. In both these cases, R_2 always outputs 0. Now let $\mathcal{S}$ be an arbitrary unbounded adaptive ideal-process simulator.

(Below "overwhelming probability" refers to $1-neg$ for some negligible function neg.) If, when interacting with $\mathcal{S}$ in the ideal model, whenever $s_1 = s_2 = 1$, R_2 outputs 0 with overwhelming probability, then it must be that with overwhelming probability, whenever $s_1 = s_2 = 1$, $\mathcal{S}$ corrupts D before D hands s_1, s_2 to the trusted party. However, in the ideal process, before the trusted party takes the inputs and computes the function, corrupting a party provides only its input, and no other information. Thus, in our case, before D is corrupted $\mathcal{S}$ cannot gain any information. It follows that $\mathcal{S}$ corrupts D before D hands s_1, s_2 to the trusted party with the same probability for any input s_1, s_2, and in particular when the input is $s_1 = s_2 = 0$. However in the real world, $\mathcal{A}$ never corrupts D in this case, and so the global views are significantly different.

Claim 1 and Claim 2 together imply that our example separates adaptive security from non-adaptive security for active adversaries in all settings considered. Thus we have:

Theorem 3. *For active adversaries, adaptive security is strictly stronger than non-adaptive security, under any notion of security, as long as there are at least three parties.*

Discussion. The essential difference between adaptive and non-adaptive security is well captured by the simplicity of the protocol used in our separating example, which at first look may seem like a very "harmless" protocol. Indeed, π_{act} is a straight-forward implementation of the function f_{act}, which just "mimics" the ideal-world computation, replacing the trusted party passing input from one party to the output of another party, by directly sending the message between the parties. For the non-adaptive setting, this intuition translates into a proof that any adversary $\mathcal{A}$ can be simulated by an adversary $\mathcal{S}$ in the ideal world. However, as we have shown, the protocol is susceptible to an attack by an adaptive adversary.

In the heart of this separation is the idea that some information in the protocol (the value of s_1 in our example) is revealed prematurely before the parties have "committed" to their inputs. Thus, an adaptive adversary may take advantage of that by choosing whether to corrupt a party (and which one) based on this information, and then changing the party's input to influence the global output of the execution.

On the other hand, as we will show, for a passive adversary and information theoretic security, non-adaptive security is equivalent to adaptive security. This may suggest the intuition that even for active adversaries, in the information-theoretic setting, adaptive and non-adaptive security may be equivalent for a subclass of protocols that excludes examples of the above nature; that is, for protocols where "no information is revealed before the parties have committed to their inputs". This is in fact the case for many existing protocols (cf., [BGW88,CDM98]), and furthermore, the definition of Dodis-Micali-Rogaway *requires* this condition. In Section 3 we indeed formalize and prove this intuition, showing equivalence for the definition of Dodis-Micali-Rogaway.

Finally, we remark that for *two* parties and active adversaries, the situation is more involved: In the IT setting, adaptive security is equivalent to non-adaptive security. In the universal and computational settings, we have a separating example showing that adaptive security is strictly stronger, assuming perfectly hiding bit-commitment exists (which holds under standard complexity assumptions). However, this example heavily relies on a technical requirement, called post-execution corruptibility (PEC), which is part of the definition of adaptive security, needed in order to guarantee secure composability of protocols (the technical meaning of the requirement is described along with the definition in [CDDIM01]). In contrast, the above three party separating example holds in all settings, regardless of whether the PEC requirement is imposed or not.[1]

2.2 Separation for passive adversaries and a large number of players

In [CFGN96], Canetti et al. show an example protocol that separates adaptive and non-adaptive security for passive adversaries and a large number of players, when only statistical or computational emulation is required. This separation holds for universal, IT, and computational security. Very roughly, the protocol is based on sharing a secret among a large set of players, making the identity of the set very hard to guess for a non-adaptive adversary, but easy for an adaptive one. We refer the reader to [CFGN96] for details of the example.

To complete the picture, we show an example that, under standard complexity assumptions, separates adaptive and non-adaptive security even when perfect emulation is required, for the universal or computational security model. The example is only sketched here, and the complete proof and definitions of the standard primitives used, are deferred to the final version of the paper.

Our example relies on the existence of perfectly hiding bit commitment schemes and collision-intractable hash functions.[2] For n players, we will need to hash n commitments in a collision-intractable manner. Thus, the number of players required depends on the strength of the assumption: For n that is polynomial in the security parameter k, this is a standard assumption, whereas for $n = \omega(\log k)$ this requires a stronger assumption. For simplicity, we refer below to a large number of players, instead of making the explicit distinction based on the quality of computational assumption.

The protocol involves players $P_0, P_1, \ldots, P_n$, where the input of P_0 is a function h from a family of collision intractable hash functions, and a public key pk for a perfectly hiding bit commitment scheme. The input of each other P_i is a bit b_i. The output of each player is h, pk. The protocol proceeds as follows:

1. P_0 sends h, pk to all other players.

[1] The setting of two parties *without* PEC is only of interest if we are considering a 2-party protocol as a standalone application, without composing it with multi-player protocols. For this setting, we can prove equivalence of adaptive and non-adaptive security in the secure channels model or when the simulation is black box.

[2] This example is an extension of another example given in [CFGN96], which uses only bit commitment, and works only for black-box simulators.

2. Each P_i, $i \geq 1$ computes and broadcasts a commitment $c_i = commit(pk, b_i, r_i)$.
3. All players output h, pk.

We allow the adversary to corrupt P_0 and in addition any subset of size $n/2$ of the other players.

Then this protocol is non-adaptively universally secure, with perfect emulation (since the bit commitment used is perfectly hiding). However, the protocol is adaptively insecure (both universally and computationally): Consider an adversary $\mathcal{A}$ that first corrupts P_0, computes the hash function on all the commitments sent, and interprets it as a subset of $n/2$ players to which $\mathcal{A}$ subsequently breaks. It can then be shown that any simulator for $\mathcal{A}$ can be used to either break the commitment scheme, or find collisions in h.

We thus have the following theorem.

Theorem 4. *For passive adversaries and a large number of parties, adaptive security is strictly stronger than non-adaptive security, under all notions of security except IT with perfect emulation. This holds unconditionally for either statistical or computational emulation, and under the assumption that a perfectly hiding bit commitment scheme and a collision intractable hash function family exist, for perfect emulation.*

2.3 Equivalence for passive adversaries and a small number of parties

This section proves that adaptive and non-adaptive security against a *passive* adversary are equivalent when the number of parties is small.

Before going into our results, we need to elaborate on a simplifying assumption we make in this section. As previously mentioned, the [C00] definition of adaptive security (as well as [B91,MR91], in different ways) include a special technical requirement, called post-execution corruptibility (PEC). This requirement is in general needed in order to guarantee secure composition of protocols in the adaptive setting (see [CDDIM01] for more technical details about PEC).

However, in the particular setting of this section, i.e. passive adversaries and a small number of players, it turns out that PEC is an "overkill" requirement for guaranteeing composability of protocols. Very informally, the argument for this is the following. Let π and ρ be protocols that are adaptively secure without the PEC property. These protocols are (of course) also non-adaptively secure. Since the non-adaptive definition of security is closed under (non-concurrent) composition [C00], it follows that the 'composed' protocol, $\pi \circ \rho$, is non-adaptively secure. By our result given below, the composed protocol is also adaptively secure (without PEC).

We conclude that in the setting of this section, PEC is not needed to guarantee adaptively secure composition, and therefore we discuss in this section only results that hold without assuming the PEC requirement.[3]

[3] If we were to assume the PEC requirement, we can in fact show a two-party protocol which is non-adaptively secure, but adaptively insecure (this is the same example

We first note that the general definition takes a simpler form in the passive case. In particular, in the passive case we may assume without loss of generality that the real-life adversary waits until the protocol terminates, and then starts to adaptively corrupt the parties; corrupting parties at an earlier stage is clearly of no advantage in the passive case. Similarly, the ideal-process adversary may be assumed to corrupt parties after the ideal function evaluation terminates. To further ease the exposition, we will make in the remainder of this section the following simplifying assumptions: (1) assume that the adversary is deterministic; (2) assume that the function computed by the protocol is deterministic; and (3) ignore auxiliary inputs. The results in this section generalize to hold without the above assumptions.

The card game In attempting to prove equivalence between non-adaptive and adaptive security, it may be helpful to picture the following game. Let $\mathcal{B} \subseteq 2^{[n]}$ be a monotone adversary structure. The game involves two players, the *adversary* and the *simulator*, and n distinct cards. The two players are bound to different rules, as specified below.

Adversary. When the adversary plays, the faces of the n cards are picked from some (unknown) joint distribution $V = (V_1, \ldots, V_n)$ and are initially covered. The adversary proceeds by sequentially uncovering cards according to a fixed deterministic strategy; that is, the choice of the next card to be uncovered is determined by the contents of previously uncovered cards. Moreover, the index set of uncovered cards should always remain within the confines of the structure $\mathcal{B}$. After terminating, the adversary's output consists of the identity and the contents of all uncovered cards.

Simulator. The simulator plays in a different room. It is initially given n distinct *blank* cards, all of which are covered. Similarly to the adversary, it is allowed to gradually uncover cards, as long as the set of uncovered cards remains in $\mathcal{B}$. Its goal is to fill the blank uncovered cards with content, so that the final configuration (including the identity and contents of uncovered cards) is "similarly" distributed to the adversary's output. (The precise sense of this similarity requirement will depend on the specific security setting.) Note that unless the simulator has some form of access to the unknown distribution V, the game would not make much sense. Indeed, we grant the simulator the following type of restricted access to V. At each stage, when the set of uncovered cards is some $b \in \mathcal{B}$, the simulator may freely sample from some fixed distribution $\tilde{V}_b$ which is guaranteed to be "similar" to V_b, the restriction of V to b. (Again, the type of this similarity depends on the setting.) The $|\mathcal{B}|$ distributions V_b may be arbitrarily (or adversarially) fixed, as long as they conform to the above similarity condition.

---

based on perfectly hiding bit commitment which was mentioned in the end of Section 2.1). Thus, strictly speaking, there is a separation in this setting under the [C00] definition. The results in other sections hold regardless of whether the PEC requirement is imposed or not.

Let us briefly explain the analogy between the above game and the question of non-adaptive versus adaptive security. Fix some n-party protocol π computing a deterministic function f, and suppose that π is non-adaptively secure against a passive $\mathcal{B}$-limited adversary. The n cards correspond to the n parties. The distribution V corresponds to the parties' joint view under an input x, which is a-priori unknown. Uncovering the i-th card by the adversary and learning V_i corresponds to corrupting the i-th party P_i in the real-life process and learning its entire view: its input, random input, communication messages, and output. Uncovering the i-th card by the simulator corresponds to corrupting P_i in the ideal-model process. Finally, each distribution $\tilde{V}_b$ from which the simulator can sample corresponds to a simulation of a non-adaptive adversary corrupting b, which exists under the assumption that π is non-adaptively secure. Note that the simulator can access $\tilde{V}_b$ only when all cards in b are uncovered; this reflects the fact that the non-adaptive simulation cannot proceed without learning the inputs and outputs of corrupted parties. The types of similarity between V_b and $\tilde{V}_b$ we will consider are *perfect*, *statistical*, and *computational*, corresponding to the type of non-adaptive emulation we assume. We will also consider the relation between the computational complexity of the adversary and that of the simulator, addressing the security variants in which the simulator is computationally bounded.

Remark. The above game models a secure channels setting, in which the adversary has no information before corrupting a party. To model open channels (or a "broadcast" channel), the distribution V should be augmented with an additional entry V_0, whose card is initially uncovered. The analysis that will follow can be easily adapted to deal with this more general setting.

Perfect emulation We first deal with perfect emulation, i.e., the case where $\tilde{V}_b = V_b$ for all $b \in \mathcal{B}$. In this setting, we show how to construct an adaptive simulator running in (expected) time polynomial in the time of the adversary and the size of the adversary structure. The construction from this section will allow us to prove equivalence of non-adaptive and adaptive security both in the information-theoretic case (see Section 2.4) and, when the adversary structure is small, in the universal case.

A black-box simulator. To prove equivalence between non-adaptive and adaptive security it suffices to show that for any adversary strategy $\mathcal{A}$ there exists a simulator strategy $\mathcal{S}$, such that under any distribution V the simulator wins. In fact, we will construct a single simulator $\mathcal{S}$ with a black-box access to $\mathcal{A}$, and later analyze it in various settings.

A convention for measuring the running time of black-box simulators. In the following we view adaptive simulators as algorithms supplied with two types of oracles: *distribution oracles* $\tilde{V}_b$, implemented by a non-adaptive ideal-process adversary (to be referred to as a *non-adaptive simulator*), and an adaptive *adversary oracle* $\mathcal{A}$. In measuring the running time of a simulator, each oracle call will count as a single step. This convention is convenient for proving

universal security: If the protocol has universal non-adaptive security and the black-box simulator $\mathcal{S}$ runs in expected time poly(k) then, after substituting appropriate implementations of the oracles, the expected running time of $\mathcal{S}$ is polynomial in k and the expected running time of $\mathcal{A}$.[4]

In the description and analysis of $\mathcal{S}$ we will use the following additional notation. By v_b, where v is an n-tuple (presumably an instance of V) and $b \subseteq [n]$ is a set, we denote the restriction of v to its b-entries. For notational convenience, we assume that the entries of a partial view v_b, obtained by restricting v or by directly sampling from $\tilde{V}_b$ or V_b, are labeled by their corresponding b-elements (so that b can be inferred from v_b). We write $v \xrightarrow{\text{A}} b$ if the joint card contents (view) v leads the adversary $\mathcal{A}$ to uncover (corrupt) the set b *at some stage*. For instance, $v \xrightarrow{\text{A}} \emptyset$ always holds. An important observation is that whether $v \xrightarrow{\text{A}} b$ holds depends only on v_b. This trivially follows from the fact that cards cannot be covered once uncovered. Hence, we will also use the notation $v' \xrightarrow{\text{A}} b$, where v' is a $|b|$-tuple representing a partial view.

In our description of the simulator we will adopt the simplified random variable notation from the game described above, but will revert to the original terminology of corrupting parties rather than uncovering cards.

Before describing our simulator $\mathcal{S}$, it is instructive to explain why a simpler simulation attempt fails. Consider a "straight line" simulator which proceeds as follows. It starts by corrupting $b = \emptyset$. At each iteration, it samples $\tilde{V}_b$ and runs the adversary on the produced view to find the first party outside b it would corrupt. The simulator corrupts this party, adds it to b, and proceeds to the next iteration (or terminates with the adversary's output if the adversary would terminate before corrupting a party outside b). This simulation approach fails for the following reason. When sampling $\tilde{V}_b$, the produced view is independent of the event which has lead the simulator to corrupt b. This makes it possible, for instance, that the simulator corrupts a set which cannot be corrupted at all in the real-life execution. The simulator $\mathcal{S}$, described next, will fix this problem by insisting that the view sampled from $\tilde{V}_b$ be consistent with the event that the adversary corrupts b.

Algorithm of $\mathcal{S}$:

1. Initialization:
   Let $b_0 = \emptyset$. The set b_i will contain the first i parties corrupted by the simulator.
2. For $i = 0, 1, 2, \ldots$ do:
   (a) Repeatedly sample $v' \xleftarrow{\text{R}} \tilde{V}_{b_i}$ (by invoking the non-adaptive simulator) until $v' \xrightarrow{\text{A}} b_i$ (i.e., the sampled partial view would lead $\mathcal{A}$ to corrupt b_i). Let v_i be the last sampled view. (Recall that v' includes the identities of parties in b_i.)

[4] Note that when the protocol has universal non-adaptive security, a distribution $\tilde{V}_b$ can be sampled in expected polynomial time from the view of an ideal-process adversary corrupting b.

(b) Invoke $\mathcal{A}$ on v_i to find the index p_{i+1} of the party which $\mathcal{A}$ is about to corrupt next (if any). If there is no such party (i.e., $\mathcal{A}$ terminates), output v_i. Otherwise, corrupt the p_{i+1}-th party, let $b_{i+1} = b_i \cup \{p_{i+1}\}$, and iterate to the next i.

The analysis of the simulator $\mathcal{S}$, appearing in [CDDIM01], shows that in the case of a perfect non-adaptive emulation ($\tilde{V}_b \stackrel{\mathrm{d}}{=} V_b$): (1) $\mathcal{S}$ perfectly emulates $\mathcal{A}$, and (2) the expected running time of $\mathcal{S}$ is linear in $|\mathcal{B}|$. We may thus conclude the following:

Theorem 5. *For function evaluation protocols with passive adversary, universal perfect security, and $n = O(\log k)$ parties, adaptive and non-adaptive security are equivalent.*

Imperfect Emulation We next address the cases of statistical and computational security against a passive adversary. Suppose that we are given an imperfect (statistical or computational) non-adaptive simulator and attempt to construct an adaptive one. If we use exactly the same approach as before, some technical problems arise: with imperfect non-adaptive emulation, it is possible that a real life adversary $\mathcal{A}$ corrupts some set with a very small probability, whereas this set is *never* corrupted in emulated views. As a result, the loop in step (2a) of the algorithm of $\mathcal{S}$ will never terminate, and the expected time will be infinite. Consequently, it is also unclear whether $\mathcal{S}$ will produce a good output distribution when given access to imperfect non-adaptive simulation oracles $\tilde{V}_b$.

We start by showing that when the size of the adversary structure is polynomial, the simulator $\mathcal{S}$ will indeed produce a (statistically or computationally) good output distribution even when given access to (statistically or computationally) imperfect non-adaptive simulators. Moreover, it will turn out that when the adversary structure is polynomial, the expected running time of $\mathcal{S}$ is polynomial *except with negligible probability*. Later, we define a more sophisticated simulator $\mathcal{S}'$ which achieves strict expected-polynomial time simulation, at the expense of requiring a stronger assumption on the size of the adversary structure.

Specifically, these results can be summarized by the following theorem, whose proof appears in [CDDIM01].

Theorem 6. *For function evaluation protocols with passive adversary and $n = O(\log k / \log\log k)$ parties, adaptive and non-adaptive security are equivalent under* any *notion of security. Moreover, with a relaxed notion of efficiency allowing a negligible failure probability, the bound on the number of parties can be improved to $n = O(\log k)$.*

We remark that Theorem 6 is essentially tight in the following sense: when $n = \omega(\log k)$, adaptive security is separated from non-adaptive security even if the adaptive simulator is allowed to be computationally unbounded.

2.4 Equivalence for passive adversaries and IT security

The analysis of the simulation from the previous section implies the following:

Theorem 7. *For function evaluation protocols with passive adversary and perfect information-theoretic security, adaptive and non-adaptive security are equivalent.*

Note that there is no dependence on the number of players in the above theorem.

3 Adaptivity vs. Non-adaptivity in the definition of Dodis-Micali-Rogaway

3.1 Review of the definition

For completeness, we start with a very short summary of the definition of secure multiparty computation by Micali and Rogaway, more specifically the version that appears in the paper by Dodis and Micali [DM00]. For additional details, please refer to [DM00].

We have n players, each player P_i starts with a value x_i as input and auxiliary input a_i. We set $a = (a_1, ...a_n); x = (x_1, ..., x_n)$.

To satisfy the definition, a protocol π must have a fixed *committal round* CR, the point at which inputs become uniquely defined, as follows: The *traffic* of a player consists of all messages he sends and receives. π must specify *input- and output functions* that map traffic to input- and output values for the function f computed. The *effective inputs* $\hat{x}_1^\pi, ..., \hat{x}_n^\pi$ are determined by applying the input functions to the traffic of each player up to and including CR. So these values are the ones that players "commit to" as their inputs. The *effective outputs* $\hat{y}_1^\pi, ..., \hat{y}_n^\pi$ are determined by applying the output functions to the entire traffic of each player.

For adversary $\mathcal{A}$ (taking random input and auxiliary input α), random variable $View(\mathcal{A}, \pi)$ is the view of $\mathcal{A}$ when attacking π. We define:

$$History(\mathcal{A}, \pi) = View(\mathcal{A}, \pi), \hat{x}^\pi, \hat{y}^\pi$$

The way $\mathcal{A}$ interacts with the protocol is as follows: in each round, $\mathcal{A}$ sees all messages from honest players in this round. He may then issue some number of corruption requests adaptively, and only then must he generate the messages to be sent to the remaining honest players.

The definition calls for existence of a simulator $\mathcal{S}$ which may depend on the protocol in question, but not the adversary. The goal of the simulator is to sample the distribution of $History(\mathcal{A}, \pi)$. To do so, it is allowed to interact with $\mathcal{A}$, but it is restricted to one-pass black-box simulation with no bound on the simulator's running time, i.e., $\mathcal{A}$ interacts with $\mathcal{S}$ in the same way it interacts with π, and $\mathcal{S}$ is not allowed to rewind $\mathcal{A}$. The simulator $\mathcal{S}$ gets an oracle O as help (where the oracle knows x, a):

- If P_j is corrupted before CR, the oracle sends x_j, a_j to $\mathcal{S}$.

- At CR, $\mathcal{S}$ applies the input functions to the view of $\mathcal{A}$ it generated so far to get effective inputs of corrupted players $\hat{x}_j^{\mathcal{S}}$. It sends these values to O. O computes the function choosing random input r and using as input the values it got from $\mathcal{S}$ for corrupted players and the real x_j's for honest players. The result is $\hat{y}^{\mathcal{S}} = (\hat{y}_1^{\mathcal{S}}, ..., \hat{y}_n^{\mathcal{S}})$. O sends the results for corrupted players back to $\mathcal{S}$.
- If P_j is corrupted in or after CR, O sends $x_j, a_j, \hat{y}_j$ to $\mathcal{S}$.

The random variable $View(\mathcal{A}, \mathcal{S})$ is the view of $\mathcal{A}$ when interacting with $\mathcal{S}$. The effective inputs $\hat{x}^{\mathcal{S}}$ are as defined above, i.e., if a P_j is corrupted before CR, then his effective input $\hat{x}_j^{\mathcal{S}}$ is determined by the input function on his traffic, else $\hat{x}_j = x_j$. The effective outputs $\hat{y}^{\mathcal{S}}$ are defined as what the oracle outputs, i.e. $\hat{y}^{\mathcal{S}} = f(\hat{x}^{\mathcal{S}}, r)$.

$$History(\mathcal{A}, \mathcal{S}) = View(\mathcal{A}, \mathcal{S}), \hat{x}^{\mathcal{S}}, \hat{y}^{\mathcal{S}}$$

We can now define that π computes f securely iff there exists a simulator $\mathcal{S}$ such that for every adversary $\mathcal{A}$, and every x, a, α,

$$History(\mathcal{A}, \mathcal{S}) \equiv History(\mathcal{A}, \pi)$$

i.e., the two variables have identical distributions.

At first sight it may seem strange that the definition does not explicitly require that players who are honest up to CR actually commit to their real inputs, or that players who are never corrupted really receive "correct" values. But this follows from the definition:

Lemma 1. *If π computes f securely, then the input- and output functions are such that if P_j remains honest up to CR, then $\hat{x}_j^{\pi} = x_j$. And if P_j is never corrupted, then $\hat{y}_j^{\pi}$ is the j'th component of $f(\hat{x}^{\pi}, r)$, for a random r.*

Proof. Consider an adversary $\mathcal{A}_j$ that never corrupts P_j. Then the first claim follows from $x_j = \hat{x}_j^{\mathcal{S}}$ and $History(\mathcal{A}_j, \mathcal{S}) \equiv History(\mathcal{A}_j, \pi)$. The second follows from $History(\mathcal{A}_j, \mathcal{S}) \equiv History(\mathcal{A}_j, \pi)$ and the fact that the correlation $\hat{y}_j^{\mathcal{S}} = f(\hat{x}^{\mathcal{S}}, r)_j$ between $\hat{x}^{\mathcal{S}}$ and $\hat{y}^{\mathcal{S}}$ always holds.

Note that this lemma continues to hold, even if we only assume static security.

3.2 Equivalence of adaptive and non-adaptive security

It turns out to be convenient in the following to define the notion of a *partial history*, of an adversary $\mathcal{A}$ that either attacks π or interacts with a simulator. A partial history constrains the history up to a point at the start of, or inside round j for some j. That is, round $j-1$ has been completed but round j has not. If $j \leq CR$, then such a partial history consists of a view of the adversary up to round j, and possibly including some part of round j. If $j > CR$, but the protocol is not finished, a partial history consists of a partial view of $\mathcal{A}$ as described before plus the effective inputs. Finally, if the protocol is finished at

round j, the history is as defined earlier: complete view of $\mathcal{A}$ plus the effective inputs and outputs.

Note that if $\mathcal{S}$ is such that $History(\mathcal{A}, \pi) \equiv History(\mathcal{A}, \mathcal{S})$, then trivially it also holds that the partial histories of $\mathcal{A}, \pi$ and of $\mathcal{A}, \mathcal{S}$ ending at any point are identically distributed. Moreover, since $\mathcal{S}$ never rewinds, the value of the partial history of $\mathcal{A}, \mathcal{S}$ at some point in time will be fixed as soon as $\mathcal{S}$ has reached that point in the simulation.

We can then slightly extend the actions an adversary can take: a *halting adversary* $\mathcal{A}'$ is one that interacts with protocol or simulator in the normal way, but may at any point output a special halting symbol and then stop. In the simulation, if the simulator receives such a symbol, the simulation process also stops. The histories $History(\mathcal{A}', \pi), History(\mathcal{A}', \mathcal{S})$ are defined to be whatever the partial history is at the point when $\mathcal{A}$ stops.

Trivially protocol π is secure in the above definition if and only if, for any halting adversary $\mathcal{A}'$, $History(\mathcal{A}', \pi) \equiv History(\mathcal{A}', \mathcal{S})$. Note that this extension of the definition does not capture any new security properties, it is simply a "hack" that turns out to be convenient in the proof of the following theorem.

In the following we assume that there exists a static (non-adaptive) simulator $\mathcal{S}_0$ such that for every *static* adversary $\mathcal{A}_0$, and every x, a, α,

$$History(\mathcal{A}_0, \mathcal{S}_0) \equiv History(\mathcal{A}_0, \pi)$$

We want to make a general simulator $\mathcal{S}$ that shows that π in fact is secure against any adaptive adversary $\mathcal{A}$, in other words, we claim

Theorem 8. *Adaptive and non-adaptive security are equivalent under the Dodis-Micali-Rogaway definition.*

To this end, we construct a static adversary $\mathcal{A}_B$ (of the halting type), for every set B that it is possible for $\mathcal{A}$ to corrupt. $\mathcal{A}_B$ plays the following strategy, where we assume that $\mathcal{A}_B$ is given black-box access to (adaptive) adversary $\mathcal{A}$, running with some random and auxiliary inputs $r_{\mathcal{A}}$ and α[5]:

Algorithm of $\mathcal{A}_B$

1. Corrupt the set B initially. For each $P_j \in B$, initialize the *honest* algorithm for P_j, using as input x_j, a_j learnt from corrupting P_j (and fresh random input).
2. Start executing the protocol, initially letting the players in B play honestly, but keeping a record of their views. At the same time, start running $\mathcal{A}$.
3. Whenever $\mathcal{A}$ issues a corruption request for player P_j, we do the following: if $P_j \in B$, we provide $\mathcal{A}$ with x_j, a_j and all internal data of P_j. After this point, all messages for P_j are sent to $\mathcal{A}$, and we let $\mathcal{A}$ decide the actions of P_j from this point. If $P_j \notin B$, output a halt symbol and stop.

[5] We could also have given $r_{\mathcal{A}}, \alpha$ as input to $\mathcal{A}_B$, letting it simulate the algorithm of $\mathcal{A}$, but the set-up we use is more convenient in the following.

The idea in the following is to use the assumed ability (by $\mathcal{S}_0$) to generate histories of $\mathcal{A}_B$ attacking π to generate histories of $\mathcal{A}$ attacking π. Note that in any round of π, the current history of $\mathcal{A}_B$ contains both the (so far honest) history of 0 or more players that $\mathcal{A}$ has not yet corrupted, plus the view so far of $\mathcal{A}$. So for any such (partial) history u of $\mathcal{A}_B$, we let $Aview(u)$ be the view of $\mathcal{A}$ that can be extracted from u in the natural way.

In particular, if u is a history of $\mathcal{A}_B$ that ends after the final round of the protocol, then $Aview(u)$ is a complete view of $\mathcal{A}$ where $\mathcal{A}$ corrupted only players in B, whereas if u ends before the protocol is complete, $Aview(u)$ ends in the some round where $\mathcal{A}$ requested to corrupt some player outside B.

We are now ready to describe how a simulator $\mathcal{S}$ can be constructed. The full algorithm and analysis of $\mathcal{S}$ are omitted for lack of space, and appear in [CDDIM01]. Here we give only the basic idea:

From the beginning, A has not corrupted any players. So we can create the start of a history by running $(A_\emptyset, S_0)$ (recall that $A_\emptyset$ runs A "in the background"). This will stop as soon as A corrupts the first player P_j. Say this happens in round i. Let v be the view of A we obtain from this. Recall that S_0 provides perfect emulation. This means that in real life when A attacks F, we would with the same probability obtain a history where up to round i, A obtains view v and all players including P_j have been honest.

Now, by construction of $A_{\{P_j\}}$ this same history up to round i can also be realized by $A_{\{P_j\}}$ attacking F: the only difference is that from the beginning $A_{\{P_j\}}$ and not the j'th player runs the honest algorithm of P_j. And again by assumption on S_0, the history can also be realized by $A_{\{P_j\}}$ interacting with S_0.

We can therefore (by exhaustive search over the random inputs) generate a random history of S_0 interacting with $A_{\{P_j\}}$, conditioned on the event that the view v for A is produced in the first i rounds (and moreover, this can be done without rewinding A). This process may be inefficient, but this is no problem since we consider IT-security here. Once we succeed, we let $(S_0, A_{\{P_j\}})$ continue to interact until they halt, i.e., we extend the history until the protocol is finished or A corrupts the next player (say $P_{j'}$). In the former case, we are done, and otherwise we continue in the same way with $A_{\{P_j, P_{j'}\}}$.

Once we finish the CR, the effective inputs will be determined, and we will get resulting outputs from the oracle. Note here that since we only consider one-pass blackbox simulation, we will never need to rewind back past the CR, which might otherwise create problems since then A could change its mind about the effective inputs. Thus also the one-pass black-box requirement is essential for the proof.

References

[B91] D. Beaver, "Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority", J. Cryptology, Springer-Verlag, (1991) 4: 75-122.

[B97] D. Beaver, "Plug and Play Encryption", CRYPTO 97.

[BH92] D. Beaver and S. Haber, "Cryptographic Protocols Provably secure Against Dynamic Adversaries", *Eurocrypt,* 1992.

[BGW88] M. Ben-Or, S. Goldwasser and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", *20th Symposium on Theory of Computing (STOC),* ACM, 1988, pp. 1-10.

[C00] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols", *Journal of Cryptology,* Vol. 13, No. 1, Winter 2000. On-line version at http://philby.ucsd.edu/cryptolib/1998/98-18.html.

[C00a] R. Canetti, "A unified framework for analyzing security of Protocols", manuscript, 2000. Available at http://eprint.iacr.org/2000/067.

[CDDIM01] R. Canetti, I. Damgaard, S. Dziembowski, Y. Ishai and T. Malkin, "On adaptive vs. non-adaptive security of multiparty protocols", http://eprint.iacr.org/2001.

[CFGN96] R. Canetti, U. Feige, O. Goldreich and M. Naor, "Adaptively Secure Computation", *28th Symposium on Theory of Computing (STOC),* ACM, 1996. Fuller version in MIT-LCS-TR #682, 1996.

[CDM98] R.Cramer, I.Damgaard and U.Maurer: *General Secure Multiparty Computation from Any Linear Secret-Sharing Scheme*, EuroCrypt 2000.

[CCD88] D. Chaum, C. Crepeau, and I. Damgaard. Multi-party Unconditionally Secure Protocols. In *Proc. 20th Annual Symp. on the Theory of Computing (STOC)*, pages 11–19, ACM, 1988.

[CS98] R. Cramer and V. Shoup, "A paractical public-key cryptosystem provably secure against adaptive chosen ciphertext attack", *CRYPTO '98,* 1998.

[DN00] I. Damgaard and J. Nielsen, "Improved non-committing encryption schemes based on a general complexity assumption", CRYPTO 2000.

[DM00] Y. Dodis and S. Micali, "Parallel Reducibility for Information-Theoretically Secure Computation", CRYPTO 2000.

[DDN91] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography", SICOMP, to appear. Preliminary version in STOC 91.

[GMW87] O. Goldreich, S. Micali and A. Wigderson, "How to Play any Mental Game", *19th Symposium on Theory of Computing (STOC),* ACM, 1987, pp. 218-229.

[GL90] S. Goldwasser, and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", *CRYPTO '90, LNCS 537,* Springer-Verlag, 1990.

[GM84] S. Goldwasser and S. Micali, "Probabilistic encryption", *JCSS,* Vol. 28, No 2, April 1984, pp. 270-299.

[MR91] S. Micali and P. Rogaway, "Secure Computation", unpublished manuscript, 1992. Preliminary version in *CRYPTO '91, LNCS 576,* Springer-Verlag, 1991.

[PW94] B. Pfitzmann and M.Waidner, "A General Framework for Formal Notions of Secure Systems", Hildesheimer Informatik-Berichte, ISSN 0941-3014, April 1994.

[PSW00] B. Pfitzmann, M. Schunter and M.Waidner, "Secure Reactive Systems", IBM Technical report RZ 3206 (93252), May 2000.

[S99] A. Sahai, "Non malleable, non-interactive zero knowlege and adaptive chosen ciphertext security", FOCS 99.

[Y82] A. Yao, "Protocols for Secure Computation", In *Proc. 23rd Annual Symp. on Foundations of Computer Science (FOCS),* pages 160–164. IEEE, 1982.

[Y86] A. Yao, "How to generate and exchange secrets", In *Proc. 27th Annual Symp. on Foundations of Computer Science (FOCS),* pages 162–167. IEEE, 1986.