Is it safe to lie? Revisiting Incoercible Multi-Party Computation in the Universal Composability Framework

Saskia Bayreuther, Robin Berger, Eva Hetzel, and Jörn Müller-Quade

KASTEL Security Research Labs, Karlsruhe Institute of Technology, Karlsruhe, Germany
{firstname.lastname}@kit.edu

Abstract. In a multi-party protocol, incoercibility aims to protect protocol participants by allowing them to use their honest inputs even in the presence of a coercer. A coercer is an adversary who pressures protocol parties into deviating from the protocol. One goal of an incoercible protocol is that the coercer cannot distinguish if the coerced party obeys their command or is deceiving: The coerced party should be able to *plausibly deny* their deception to ensure their safety. In the past, several attempts at modeling incoercibility in the Universal Composability framework have been made. In Alwen, Ostrovsky, Zhou, and Zikas' model (CRYPTO 2015), the deception is modeled as a mapping of protocol messages that are sent and received by the coerced party beyond the reach of the coercer. Similar to proofs in plain UC, in order to prove the incoercibility of a protocol, one shows the indistinguishability of ideal and real worlds. Alwen et al.'s incoercibility definition makes use of four worlds: In both the ideal and the real worlds either coercion or deception takes place. If the environment can neither distinguish between the two coercion worlds nor between the two deception worlds for all ideal deception strategies, the protocol incoercibly UC-realizes the ideal functionality. While this shows that it is always possible for a coerced party to deviate from the coercer's instructions and deceive them, it does not cover if this deception can be detected by the coercer.

In this paper, we tackle the question how the perceived difference between deception and coercion can be modeled in UC. In particular, we make the following contributions. First, we refine the incoercibility notion by Alwen et al. by limiting the deception strategies to ones that allow the coerced party to plausibly deny the deception. We propose a multi-party computation protocol, derived from the one presented by Alwen et al., that fulfills this notion. The protocol uses hardware tokens to make UC-protocols, now including reactive functionalities, incoercible, where coerced parties can deceive the coercer about inputs as well as outputs. Secondly, we define a notion for plausible deniability, which we call Γ-deniability, and show that when the evaluated function is a differentially private mechanism, the deceiving party can plausibly deny their actions.

Keywords: Incoercibility · Plausible Deniability · Universal Composability · Differential Privacy.

1 Introduction

Multi-party computation (MPC) allows mutually distrusting parties to jointly perform a computation while guaranteeing privacy of inputs and correctness of outputs, even if a subset of the parties is corrupted. Coercions are a special type of corruption in MPC. A coerced party is persuaded or forced by their coercer to use a dictated input and then reveal details about the computation, which can serve as a proof that the coercer's instructions were followed. The security notion which guarantees that a protocol can mitigate coercion attacks is called *incoercibility*. Intuitively, if a protocol is incoercible, the coerced party is able to *deceive* the coercer and pretend that they followed the coercer's instructions faithfully, while actually acting according to their own will.

Incoerciblity is a desirable property, *e.g.*, of voting protocols. Voters are not supposed to obtain any receipt, which could prove to a third party what they voted for (receipt-freeness). This includes scenarios like blackmail as well as voters who would voluntarily give up the confidentiality of the ballot to sell their vote. While incoercibility can be achieved in paper-based voting by, *e.g.*, prohibiting taking photos of ballots, it is harder to guarantee in electronic voting. Voters want to be convinced that their vote was processed correctly, while at the same time no other party should be able to verify the vote.

Another context, where incoercibility is relevant, is whistleblowing. Whistleblowers want to be able to deny that certain information was leaked by them in order to protect themselves from persecution. Therefore, incoercible messaging and authentication become important as well.

While the applications of incoercible protocols are numerous, it is still an open question how to model this property. Many previous works [Alw+15; CGP15; UM10] consider the notion in the Universal Composability framework [Can01], which is a popular proof framework in contexts where multiple protocols running concurrently have to be considered. However, in particular the models by Alwen et al. [Alw+15] and Unruh and Müller-Quade [UM10] do not depict and sometimes even contradict the intuition of what incoercibility is supposed to mean: According to their definitions, a protocol can be proven to be incoercible, even if the coercer can always distinguish between a party following their instructions and a party trying to deceive them. This is because the environment, having access to all inputs, can compare the output where coerced parties deceived to the output where coerced parties obeyed. The central question of the coerced party therefore remains unanswered: Is it safe to lie to the coercer, or will the lie be detected?

1.1 Contributions and Technical Overview

Similar to corruption, there are various levels of coercion. Semi-honest coercion, for example, is comparable to passive corruption, where the adversary does not

interfere with the protocol execution, but receives the view of the coerced party, including the protocol transcript. Upon active coercion, the interfaces of the coerced party are controlled by the adversary, as they would be upon malicious corruption. What sets coercion apart from corruption, is the deception strategy. Although a coerced party might be surveilled or controlled, they remember their original input and can deceive the coercer: They can use their original input, but pretend to the adversary that they had followed their instructions. We therefore want to give security guarantees towards coerced parties, although they are deviating from the protocol.

In the literature about modeling coercion corruptions in UC, the concept of deception strategies has been treated very differently. Unruh and Müller-Quade [UM10] use the concept of a distinct party, the deceiver, that allows a party to thwart the coercion. In the model by Alwen et al. [Alw+15], the deception is modeled as additional interactive turing machines, which apply a mapping to the messages from and to the coerced party. They explicitly require their definition of incoercibility to hold for all possible deception strategies. Canetti, Goldwasser, and Poburinnaya [CGP15] hard code the deception strategy into the protocol and let the environment decide if a party obeys the coercer or deploys the deception strategy. They only require one deception strategy per protocol since they only consider semi-honest coercions.

In contrast to the model by Canetti, Goldwasser, and Poburinnaya [CGP15], we want to regard active coercions in this paper. Since not all coercion scenarios can be covered by a single fixed deception strategy with active coercions, our definition of incoercibility allows for multiple deception strategies (Def. 4). However, unlike Alwen et al. [Alw+15], we require them to have certain properties, which are necessary for the deception not to be obvious. We weaken the UC incoercibility definition by Alwen et al. [Alw+15] by restricting the set of allowed deception strategies to ones that use the original honest input of the coerced party and that only claim outputs from the image of the function that is computed. Since the deception strategy is supposed to keep the coerced party safe, we exclude deception strategies that allow for a trivial distinction between coercion and deception.

The second requirement regarding the output of the computation is motivated by scenarios where every party receives their own private output and the coerced party wants to deceive the coercer about the output they received. Consider, *e.g.*, a private set intersection protocol with a sender and a receiver. The sender might be a server storing a list of all clients of a messaging service. The receiver wants to register at the messaging service and find out, which of their contacts are using the messaging service as well. Since the receiver does not want to share their contacts with the server, they use a private contact discovery protocol, which only reveals the intersection to the receiver [Ara+22]. The output of the protocol still contains the receiver's contacts, which they might want to lie about to a coercer. In such a case, the deception strategy would be required to hide the output of the computation in addition to the input and to protect the coerced party, the claimed output has to be plausible.

In a next step, we apply this idea of denying the output of a computation to the hardware token protocol introduced by Alwen et al. [Alw+15], which serves as a compiler turning UC-secure protocols into incoercible ones. Instead of broadcasting the output, as it is done by Alwen et al. [Alw+15], the parties receive ciphertexts of their outputs and have to use their hardware token again for decryption, making it possible to lie about the output. We prove that our adapted protocol fulfills our weakened UC incoercibility definition while allowing the parties to lie about the outputs they received.

Theorem 1 (informal). *Any generic MPC-protocol can be converted into a protocol which incoercibly performs the same computation under active coercion, assuming a symmetric encryption scheme, a signature scheme, honestly generated hardware tokens and at least one honest party.*

The first research question, which we answer affirmatively in this paper, is therefore:

Can we use hardware tokens to make UC-secure protocols incoercible and deceive the coercer about the inputs and outputs of the protocol execution?

One component of the incoercibility definition, which was adopted from Alwen et al. [Alw+15], is the explicit difference between ideal coercion and deception. This difference Δ quantifies the advantage of the environment in distinguishing between the ideal execution under coercion and the ideal execution, where the deception strategy is implemented additionally.

The main incentive to choose an incoercible protocol is that a party will be able to use their original input, even under coercion. If the coercer is able to tell whether a party is following their coercion instructions or the deception strategy, the party cannot profit from the incoercibility of the protocol, since they have to fear the consequences of their deception. Even if a protocol is incoercible according to the definition of Alwen et al. [Alw+15], Δ might be big. The party therefore wants to choose a deception strategy that is *plausibly deniable*, *i.e.*, where the coercion execution is as plausible as the deception execution. This leads to our second research question:

What is the probability that a deception is discovered by the coercer?

The plausibility of the lie to the coercer depends on various factors: the deception strategy, the coercer's knowledge, and the resulting distinguishing advantage Δ. We call our definition of plausible deniability Γ-deniability (Def. 6) and bound the distinguishing advantage Δ by Γ. To model the coercer's limited knowledge, we restrict the environment, since in the UC experiment, the environment is the entity which tries to distinguish between executions and which controls the adversary. We then look at the deception strategies and environments from the set of considered environments which maximize Δ to determine the likelihood of a lie being discovered. Jumping ahead, even though the model

is based on the Universal Composability framework, Γ-deniability does not compose in the sense that a deceiving party will not have the same chances of being detected after every time they try to deceive the coercer.

To give an intuition on the parameters ϵ and δ, we show how they can be computed if the protocol evaluates an (ϵ, δ)-differentially private mechanism.

Theorem 2 (informal). *If the function that is evaluated by the ideal functionality is an (ϵ, δ)-differentially private mechanism, the protocol which is incoercibly UC-realizing the functionality fulfills our definition of plausible deniability with parameters that can be derived from ϵ and δ.*

To summarize, our contributions are the following:

- We refine the incoercibility definition by Alwen, Ostrovsky, Zhou, and Zikas [Alw+15, Definiton 1] by limiting the ideal deception strategies. We modify the token protocol proposed by Alwen et al., which can be used as a compiler to make certain protocols incoercible, to cover reactive functionalities and guarantee output deniability.
- We give a definition for plausible deniability and show that in this setting, we lose composability. To better capture a coercer's knowledge realistically, we restrict the environment in this step. We show which deniability guarantees result from evaluating (ϵ, δ)-differentially private mechanisms.

1.2 Related Work

The incoercibility security property has received attention in the design of various protocols and cryptographic building blocks like authentication [Dod+09; FL25], key exchange in messaging in general [UG15; UG18] as well as the handshake of the Signal protocol in particular [FJ24], and electronic voting schemes [BMQR07; Dam+25; Par+24].

The first formal definition of incoercibility was introduced by Canetti and Gennaro [CG96]. They model incoercibility as a standalone property for multiparty computation and require that a party, given two inputs x and x', can produce indistinguishable transcripts using either input. Later, this definition was translated to the UC framework [CGP15] and has since been further improved [CP20]. In [CGP15], coerced parties ask the environment whether to continue with real or fake inputs and randomness. Every protocol comes with a specified faking procedure in case the party is instructed to deceive the adversary using the fake inputs. In the ideal world, the ideal functionality gets to either provide real or fake information, again according to the environment's instructions. If the real-world adversary and the ideal-world simulator gain the same information, the protocol is considered incoercible.

Further frameworks capturing incoercibility in UC have been proposed by Unruh and Müller-Quade [UM10] and Alwen et al. [Alw+15]. In the Composable Incoercibility framework UC/c [UM10], the deceiver is introduced as an additional adversarial entity. A protocol is considered incoercible if the advantage of the adversary to distinguish between a party obeying them and following

a deception strategy is not greater than the advantage would be using only public information, *e.g.*, the tally in a voting protocol.

Alwen et al. [Alw+15] propose their incoercibility notion $\mathtt{C} - IUC$, where they require indistinguishability between the real and ideal coercion worlds and the real and ideal deception worlds. For every ideal deception strategy, there has to exist a real deception strategy as well. They allow for a difference between both ideal worlds, which they call Δ.

However, all of these works have in common that it remains unclear whether a coerced party can plausibly deny acting according to their own will and therefore safely deceive their coercer.

In the literature on anonymous communication networks, plausible deniability refers to the adversary not being able to link certain actions to a specific user, *e.g.*, sending a message [Kuh+21]. In our setting, it is not necessary to find a party that is indistinguishable from the coerced one in order to deceive, but the deception strategy has to fake evidence that could prove to the coercer which input was used and which output the coerced party received.

Besides anonymous communication, plausible deniability is also a desirable property in signed credentials. In this context, it usually refers to a property also called non-transferability, which is fulfilled when the verifier of the credential cannot convince a third party of its validity, since the verifier has the ability to create genuine-looking transcripts of the credential presentation themself [BL26].

In this work, we build upon the incoercibility model by Alwen et al. [Alw+15] and define plausible deniability in the broader context of generic MPC-protocols.

1.3 Outline

The preliminaries are presented in Sect. 2, including an overview of the Universal Composability framework and the UC incoercibility notion by Alwen et al. [Alw+15]. Section 3 contains our weakened definition of incoercibility in UC and an adaptation of the protocol using hardware tokens by Alwen et al. [Alw+15], which now allows for reactive functionalities and deniable outputs. In Sect. 4, we define Γ-deniability to quantify the risk of a coerced party trying to deceive their coercer. We discuss the meaning of our definition in the context of differential privacy and show that once we restrict the advantage of the environment, composability is lost. Section 5 concludes the paper.

2 Preliminaries

This section introduces notations used in this paper and gives an introduction to the Universal Composability (UC) framework. We also give an introduction to incoercibility in UC based on the work by Alwen et al. [Alw+15].

2.1 Notation

Let $\kappa \in \mathbb{N}$ denote the security parameter. Let $[n]$ be the set of integers $\{1, ..., n\}$. Let $x \leftarrow X$ denote that some probabilistic algorithm X returns x. Let $y \xleftarrow{\$} Y$

denote that y is sampled uniformly at random from some set Y. Let $\mathsf{negl}(\kappa)$ denote an unspecified negligible function in κ. Let $\perp$ denote a null value.

2.2 Universal Composability

The following provides a brief overview of the Universal Composability (UC) framework [Can01; Can20]. In contrast to game-based security notions, UC is based on simulation. In a nutshell, a protocol π is UC-secure, if executing π is indistinguishable from executing an ideal protocol using an ideal functionality $\mathcal{F}$, that fulfills all the required security properties by definition.

The Real Experiment. The experiment, in which the parties execute the protocol π, is called the real experiment and denoted by $\mathsf{Exec}(\pi, \mathcal{A}, \mathcal{Z})$, where $\mathcal{A}$ is the adversary and $\mathcal{Z}$ the environment. The environment $\mathcal{Z}$ is the PPT machine trying to distinguish the real experiment from the ideal experiment. While trying to distinguish between the experiments, the environment can interactively take part in them. It chooses the inputs of the parties, controls the adversary, and is reported all communication between parties over the network as well as the output of the protocol execution. Through the adversary $\mathcal{A}$, it can allow messages to be delivered, but it can also delay, drop, alter or inject messages. If a party is corrupted maliciously by the adversary, the adversary has full control over their interfaces.

The Ideal Experiment. In the ideal experiment, $\mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathcal{S}, \mathcal{Z})$, the ideal functionality $\mathcal{F}$ performs all computations, while the parties are replaced with dummy parties, which forward their inputs to $\mathcal{F}$ and the outputs from $\mathcal{F}$. $\mathcal{S}$ is the ideal-world adversary and is called the simulator. The task of the simulator is to recreate the real experiment by faking protocol messages and the internal state of corrupted parties, which are consistent with the simulated protocol execution. In contrast to the protocol parties, the ideal functionality is incorruptible. The ideal protocol executed by $\mathcal{F}$ and the dummy parties is denoted by $\text{IDEAL}(\mathcal{F})$.

UC-Security. If the environment is unable to distinguish between the real experiment and the ideal experiment, we say that π UC-realizes $\mathcal{F}$ and is therefore as secure as the ideal functionality. Referring to the ideal protocol instead of the functionality itself, π is UC-secure if it UC-emulates the ideal protocol $\text{IDEAL}(\mathcal{F})$ ($\pi \geq \text{IDEAL}(\mathcal{F})$) according to Def. 1:

Definition 1 (UC-emulation, restricted model [Can01, Definition 1]). *Protocol π UC-emulates protocol ϕ if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that, for any PPT environment $\mathcal{Z}$, the ensembles $\mathsf{Exec}(\pi, \mathcal{A}, \mathcal{Z})$ and $\mathsf{Exec}(\phi, \mathcal{S}, \mathcal{Z})$ are indistinguishable. That is, for any input, the probability that $\mathcal{Z}$ outputs 1 after interacting with $\mathcal{A}$ and π differs by at most a negligible amount from the probability that $\mathcal{Z}$ outputs 1 after interacting with $\mathcal{S}$ and ϕ.*

UC Framework Conventions. Every protocol instance is uniquely identified by the tuple of session identifier and code (sid, μ). Party identifiers (PIDs) are used additionally to refer to a party within a session. Every machine is uniquely identified by its extended identity (sid, μ, pid). We refer to the extended identity of party i by simply writing P_i. Ideal functionalities have the PID $\perp$.

2.3 UC Incoercibility

Let $\mathcal{I} \subseteq [n]$ be a set of coerced parties chosen statically at the beginning of the protocol execution by $\mathcal{Z}$. The execution of protocol π with coercion $\mathtt{C}$ corresponds to executing the replaced protocol machine $\mathtt{C}_i(\pi)$ for each party $i \in \mathcal{I}$. Alwen et al. [Alw+15] define incoercibility as follows:

Definition 2 (UC Incoercibility [Alw+15, Definition 1]). *Let π be an n-party protocol and for an n-party functionality $\mathcal{F}$ let* IDEAL($\mathcal{F}$) *denote the dummy $\mathcal{F}$-hybrid protocol, and let $\mathtt{C}$ be a coercion. We say that π $\mathtt{C} - IUC$ realizes $\mathcal{F}$ if for every $i \in [n]$ and every ideal deception strategy DI_i there exists a real deception strategy DR_i with the following property. For every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any set $\mathsf{DI}_\mathcal{I} = \{\mathsf{DI}_i : i \in \mathcal{I}\}$ and every environment $\mathcal{Z}$:*

$$\mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{dum}_\mathcal{I}, \mathcal{S}, \mathcal{Z}) \leq \mathsf{Exec}(\pi, \mathtt{C}_\mathcal{I}, \mathcal{A}, \mathcal{Z}) \wedge$$
$$\mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{DI}_\mathcal{I}, \mathcal{S}, \mathcal{Z}) \leq \mathsf{Exec}(\pi, \mathsf{DR}_\mathcal{I}, \mathcal{A}, \mathcal{Z})$$

where $\mathtt{dum}$ *denotes the dummy coercer, which relays messages from $\mathcal{Z}$ or $\mathcal{S}$ intended for $\mathcal{F}$ to $\mathcal{F}$. Other messages from $\mathcal{Z}$ and messages from $\mathcal{F}$ are forwarded to $\mathcal{S}$. Other messages from $\mathcal{S}$ are forwarded to $\mathcal{Z}$.*

They therefore require indistinguishability between the real and ideal coercion worlds and between the real and ideal deception worlds, but allow both ideal worlds and therefore also the real worlds to be distinguishable. This discrepancy between the ideal coercion world and the ideal deception world is called Δ and is defined as the distinguishing advantage of the environment, which maximizes its advantage. The advantage of the environment in distinguishing between the two real wolds only negligibly deviates from Δ.

DI_i and DR_i are the ideal-world and real-world deception strategies of party P_i, respectively. A deception strategy $\mathsf{D}_i(\pi_i)$ of P_i is defined via three interconnected interactive Turing machines (ITMs): $(\mathsf{D}_i^1, \pi_i, \mathsf{D}_i^2)$. The first ITM, D_i^1, is placed in between the environment and the coerced party P_i running the ITM π_i. It can replace all messages between the environment and P_i, in both directions. [1] D_i^2 sits in between P_i and the hybrid ideal functionalities in the real world and between P_i and $\mathcal{F}$ in the ideal world and can also replace messages in both directions. Figure 1 illustrates the placement of the ITMs in the real world.

[1] To keep consistent with this definition of a deception strategy, the UC experiment has to be modified in such a way that the environment can still address the coerced party directly, but its messages are rerouted through D_i^1.

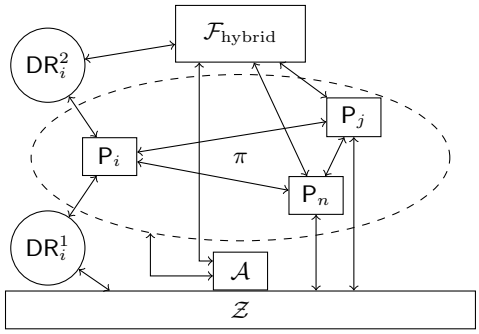


Fig. 1: Illustration of a real deception strategy $\mathsf{DR}_i(\pi_i)$

In the ideal world, π_i is usually the dummy party of party P_i, $\mathsf{dum}_{\mathsf{P}_i}$. In the real world, we have $\mathsf{DR}_i(\mathtt{C}_i)$, which means that P_i runs the coercion code. Alwen et al. [Alw+15] distinguish between three types of coercion, depending on the abilities of the coercer.

I/O Coercion. In the setting of input/output coercion, the adversary is only able to instruct the coerced party $\mathtt{C}^{\text{io}}(\pi_i)$ to use a certain input and will receive the output of the protocol as evidence.

Semi-honest Coercion. Upon semi-honest coercion $\mathtt{C}^{\text{sh}}$, the adversary is still not able to interfere with the protocol execution (besides providing a coercion input), but it now observes the entire view of the coerced party.

Active Coercion. Analogously to fully malicious corruptions, the environment controls all interfaces of actively coerced parties ($\mathtt{C}^{\text{A}}$) via the adversary.

Coercion and Deception. The definition of a deception strategy via three ITMs, $(\mathsf{D}_i^1, \pi_i, \mathsf{D}_i^2)$, is very general. However, depending on the type of coercion, not all ITMs are relevant. In the case of semi-honest coercion, D_i^1 is relevant for the deception to be able to change the output of a party towards the environment. When active coercion is considered, the environment has direct control over the party's interfaces via the adversary, whose messages to π_i are not changed by the deception strategy. The environment communicates with the coerced party via the adversary, not the party's input and output interfaces.

The ITM D_i^2 is relevant under active coercion, since it is able to change protocol messages even after the party was instructed by the adversary to give a certain input to a hybrid functionality. This is not necessary under semi-honest coercion, where the input was already altered by D_i^1 and where the adversary did not have any influence on the party's interfaces afterwards.

2.4 Ideal Functionality for General MPC

To extend the incoercible token protocol by Alwen et al. [Alw+15] from secure function evaluation to reactive protocols, we use the ideal functionality defined in Def. 3 [Bay+26]. $\mathcal{F}_f$ can be used to express any well-formed ideal functionality according to [Can+02].

Definition 3 (The ideal functionality [Bay+26]). *Let $\mathcal{P}$ be a set of parties. Let t be a polynomial and let $f\colon \{0,1\}^* \times \{0,1\}^{t(\kappa)} \times \mathcal{P} \times \{0,1\}^* \mapsto \{0,1\}^* \times (\mathcal{P} \cup \{\bot\}) \times \{0,1\}^*$ be an efficiently computable function, then we define the ideal functionality $\mathcal{F}_f$ as follows:*

$\mathcal{F}_f$

- On first invocation, set state $\coloneqq \varepsilon$.
- On input inp from party P with $pid_{\mathsf{P}} \in \mathcal{P}$,
  - Notify the adversary about this input by sending $(|\mathsf{inp}|, \mathsf{P})$ and wait for the adversary to acknowledge this.
  - Sample $r \xleftarrow{\$} \{0,1\}^{t(\kappa)}$.
  - Compute $(\text{state}', \mathsf{P}', \mathsf{out}) \coloneqq f(\text{state}, r, \mathsf{P}, \mathsf{inp})$.
  - Set state $\coloneqq$ state$'$.
  - If $\mathsf{P}' \neq \bot$, send a public delayed output out to P'.
  - If $\mathsf{P}' = \bot$, send (`"output"`) to $\mathcal{A}$.
- Ignore all inputs from other parties.

3 Incoercibility

In this section, we give a weakened definition of UC incoercibility, where the ideal deception strategies are restricted to a specified set. We further adapt the hardware token protocol by Alwen et al. [Alw+15] to allow for reactive functionalities and output deniability.

3.1 Incoercibility Definition

Intuitively, the incoercibility notion by Alwen et al. [Alw+15] captures incoercibility similar to any other security property in UC: A protocol is said to be incoercible, if the act of deceiving the coercer in the ideal world is indistinguishable from the act of deceiving the coercer in the real world. In other words, for every party deceiving in the ideal world there must exist a matching deception strategy in the real world. This captures the ability of coerced parties to deviate from the coercer's will, both in the ideal and real world. To guarantee indistinguishability between ideal and real executions, this clearly suffices since any ability a coerced party has in the ideal world should also exist in the real world. The notion quantifies over all deception strategies a party might use. This might not always be desirable, as illustrated in the following.

One example of an incoercible protocol is randomized response [War65]. It is used in surveys to make participants feel more comfortable to admit the correct answer. The scheme works as follows: Before answering the survey, the participants get to roll dice. If they throw a six, they answer with `yes`, independently of their true answer. If the die shows any other number, they answer truthfully. If someone now steals the participants' questionnaire before anonymization, each participant who answered with `yes` would be able to blame it on the dice. A coercer forcing a party to use `no` as their input could therefore be convinced that a party wanting to answer with `yes` tried to follow their instructions, but following the protocol, it was forced to answer with `yes`. In the other direction, however, if the coercer wanted the party to answer with `yes`, there would be no deception strategy that could explain why the party answered with `no`, since in this protocol, the die can only force the party to answer with `yes`. Since only one type of deception is intended by the randomized response scheme, it makes sense to exclude deception strategies that try to lie about answering with `no`.

Another example are deception strategies which lead to the coercer being able to trivially distinguish between coercion and deception, *e.g.*, deception strategies, where parties lie about having received an output that is not part of the output space of the computation.

To address this issue, we present our definition of incoercibility below. With this definition, it is possible to exclude certain deception strategies. The difference between our definition in Def. 4 and Def. 2 by Alwen et al. [Alw+15] is highlighted in purple.

Definition 4 ($\mathcal{DI}$-Weak UC Incoercibility for individual deception strategies, adapted from [Alw+15, Definition 1]). *Let π be an n-party protocol. For an n-party functionality $\mathcal{F}$ let* IDEAL($\mathcal{F}$) *denote its ideal protocol, and let* $\mathtt{C}$ *be a coercion. We say that π $(\mathtt{C}, \mathcal{DI})-wIUC$ realizes $\mathcal{F}$ if for every $i \in [n]$ and for every ideal deception strategy $\mathsf{DI}_i \in \mathcal{DI}$ there exists a real deception strategy DR_i with the following property. For every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any set $\mathsf{DI}_\mathcal{I} = \{\mathsf{DI}_i : i \in \mathcal{I}\}$ and every environment $\mathcal{Z}$:*

$$\mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{dum}_\mathcal{I}, \mathcal{S}, \mathcal{Z}) \leq \mathsf{Exec}(\pi, \mathtt{C}_\mathcal{I}, \mathcal{A}, \mathcal{Z}) \wedge \tag{1}$$

$$\mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{DI}_\mathcal{I}, \mathcal{S}, \mathcal{Z}) \leq \mathsf{Exec}(\pi, \mathsf{DR}_\mathcal{I}, \mathcal{A}, \mathcal{Z}) \tag{2}$$

where dum *denotes the dummy coercer.*

We call this incoercibility notion weak UC incoercibility, because we restrict the ideal deception strategies that it has to hold for to the specific ones in the set $\mathcal{DI}$. The definition given by Alwen et al. [Alw+15] quantifies over all ideal deception strategies, independent of their effect. The ideal deception strategies in $\mathcal{DI}$ have to fulfill the following properties in order to limit the advantage of the environment in distinguishing between coercion and deception:

- They use the deception input of coerced parties as input to the computation. The deception input is the original input of a party, before the coercion.
- The output of the computation corresponds to an evaluation of the function computed by the functionality $\mathcal{F}$.

Depending on the application and protocol, the set $\mathcal{DI}$ can be further restricted.

3.2 Incoercibility via Hardware Tokens

Alwen et al. [Alw+15] present a construction using honestly generated hardware tokens which can turn a protocol performing secure function evaluation into an incoercible one. They assume static corruptions, semi-honest and active coercions, and at least one honest party. The idea of the construction is the following: Every protocol participant receives key shares for the other parties' keys by the hybrid token functionality $\mathcal{T}_{\texttt{ThEnc}}$ ($\mathcal{F}_{\text{token}}$ in our construction) upon setup. To encrypt their inputs under their keys, they have to send the inputs to the token functionality. The parties then participate in a (UC-)secure function evaluation protocol to compute authenticated secret shares of the output. This protocol realizes the modified functionality $\hat{\mathcal{F}}$ ($\mathcal{F}_{\text{MPC}}$ in our construction), which receives the encrypted inputs and shares of the decryption keys from all parties, reconstructs the decryption keys, decrypts the inputs, computes the output according to the ideal functionality $\mathcal{F}$ and then outputs shares of the output to the parties. In the end, the parties broadcast their output shares and reconstruct the output value. Alwen et al. [Alw+15] claim that the protocol can easily be adjusted to private outputs.

This construction only considers coercions regarding the parties' inputs, which leads to two issues: First, parties being coerced to reveal the output of a computation are not covered by this construction. Second, a public output almost certainly leads to a very big distinguishing advantage Δ. We therefore want to consider a protocol that allows the deception strategies to protect the parties against coercions regarding their inputs as well as the outputs of the computation.

Authenticated Secret Sharing. We propose an ideal functionality similar to the ideal functionality $\hat{\mathcal{F}}$ in the paper by Alwen et al. [Alw+15]. In contrast to $\hat{\mathcal{F}}$, it allows for the evaluation of reactive functionalities, since it is based on $\mathcal{F}_f$ (Def. 3). It collects all key shares together with the parties' encrypted inputs. The functionality then reconstructs the keys, decrypts the inputs, computes the new state according to f and encrypts the outputs before sending them back to the parties.

Since an authenticated secret sharing scheme is used, a key share $\langle k \rangle_i$ consists of the share k_i itself, a signature σ_i, and the verification key vk [Alw+15].

Definition 5 (The ideal functionality $\mathcal{F}_{\text{MPC}}$). *Let $\mathcal{P}$ be a set of parties. Let $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric IND-CPA secure encryption scheme. Let $\Sigma = (\mathsf{KeyGen}', \mathsf{Sign}, \mathsf{Verify})$ be an EUF-CMA secure signature scheme. Let $\mathcal{F}_f$ be the ideal functionality defined in Def. 3. We define the ideal functionality $\mathcal{F}_{\text{MPC}}$ as follows:*

$\mathcal{F}_{\text{MPC}}$

- On first invocation, set state $:= \varepsilon$.
- On input $(\mathsf{inp}, \{\langle k_j \rangle_i\}_{j \in [n]})$ from party P_i with $pid_{\mathsf{P}_i} \in \mathcal{P}$:

- Store $\{k_{j,i}, \sigma_i, vk\}_{j\in[n]} \coloneqq \{\langle k_j \rangle_i\}_{j\in[n]}$.
- If all shares of all n keys have been stored,
  - Reconstruct all keys $\{k_i\}_{i\in[n]}$ from the shares. Check that the verification keys vk are the same and verify the signatures before reconstruction. If any reconstruction fails, return $\perp$.
  - For each $i \in [n]$ set $x_i \coloneqq \mathsf{SKE.Dec}(k_i, \mathsf{inp})$. If any decryption fails, return $\perp$.
  - For each $i \in [n]$,
    - Notify the adversary about this input by sending $(|x_i|, \mathsf{P}_i)$ and wait for the adversary to acknowledge this.
    - Sample $r \xleftarrow{\$} \{0,1\}^{t(\kappa)}$.
    - Compute $(\text{state}', \mathsf{P}_j, \mathsf{out}) \coloneqq f(\text{state}, r, \mathsf{P}_i, x_i)$.
    - Set state $\coloneqq$ state$'$.
    - If $\mathsf{P}_j \neq \perp$, encrypt the output to $c_{\text{out},j} \coloneqq \mathsf{SKE.Enc}(k_j, \mathsf{out})$. Send a public delayed output $c_{\text{out},j}$ to P_j.
    - If $\mathsf{P}_j = \perp$, send (`"output"`) to $\mathcal{A}$.

Remark 1. The reconstruction of the keys only works if all parties send their key shares to $\mathcal{F}_{\text{MPC}}$. Parties that do not have an input for $\mathcal{F}_{\text{MPC}}$ therefore have to explicitly receive an empty input from the environment such that they input their key shares even if they do not contribute to the evaluation of f.

The Protocol. We present a protocol based on the token protocol by Alwen et al. [Alw+15], which can be used as a compiler to make UC-secure protocols for general computations incoercible. The assumptions still include static corruptions and at least one honest party, while our focus lies on active coercion. Semi-honest coercions are also covered by our construction, with the same modifications as implemented by Alwen et al. [Alw+15]. We introduce two major changes:

- Reactive Functionalities: We extend the protocols from pure secure function evaluation to reactive functionalities, using the ideal functionality $\mathcal{F}_f$ (Def. 3) as a baseline.
- Deniable Outputs: Coerced parties should not only be able to lie about their inputs, but also about the outputs of the protocol. Therefore, in our modified protocol $\pi_{f,\text{deny}}$, the protocol outpus are encrypted and the token has to be queried again to decrypt a party's output. This way, no output shares are broadcast in the plain and as long as at least one party remains honest, the adversary cannot recover the keys to decrypt the outputs.

The differences between the token functionality and protocol by Alwen et al. [Alw+15] and the ones presented here are again highlighted in purple.

Functionality $\mathcal{F}_{\text{token}}$

Let $\mathcal{P}$ denote a set of parties. Let $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric IND-CPA secure encryption scheme. Let $\Sigma = (\mathsf{KeyGen}', \mathsf{Sign}, \mathsf{Verify})$ be an EUF-CMA secure signature scheme.

- Upon receiving message $(\texttt{"keyGen"}, i)$ from $\mathsf{P}_i \in \mathcal{P}$, if P_i is honest and message has already been received, ignore. Otherwise,
  1. Sample $k_i \leftarrow \mathsf{SKE.KeyGen}(\kappa)$.
  2. If the signature and verification keys have not been sampled yet, sample $(sk, vk) \leftarrow \Sigma.\mathsf{KeyGen}'(\kappa)$.
  3. Sample n-out-of-n authenticated sharing $\{\langle k_i \rangle_j\}_{j\in[n]} = \{k_{i,j}, \sigma_j, vk\}_{j\in[n]}$ of k_i, where $k_{i,j}$ is the jth share of k_i and $\sigma_j = \Sigma.\mathsf{Sign}(k_{i,j}, sk)$.
  4. Return $\texttt{"ok"}$.
- Upon receiving message $(\texttt{"keyShare"}, j, i)$ from $\mathsf{P}_i \in \mathcal{P}$, if key k_j has not been sampled, return $\perp$. Otherwise, return $\langle k_j \rangle_i$.
- Upon receiving message $(\texttt{"encrypt"}, m, i)$ from P_i for some $m \in \{0,1\}^{|m|}$, if k_i has not been sampled, return $\perp$. Otherwise, compute and return $c_{\text{in},i} = \mathsf{SKE.Enc}(k_i, m)$.
- Upon receiving message $(\texttt{"decrypt"}, c_{\text{out},i})$ from P_i, if k_i has not been sampled, return $\perp$. Otherwise, compute and return $y_i = \mathsf{SKE.Dec}(k_i, c_{\text{out}})$.

Protocol $\pi_{f,\text{deny}}$ for party P_i

On first activation with identifier *sid*:

- Send message $(\texttt{"keyGen"}, i)$ to $\mathcal{F}_{\text{token}}$ and receive $\texttt{"ok"}$.
- For each $j \in [n]$, send $(\texttt{"keyShare"}, j, i)$ to $\mathcal{F}_{\text{token}}$. Receive and store $\langle k_j \rangle_i$.

On input x_i with identifier *sid*:

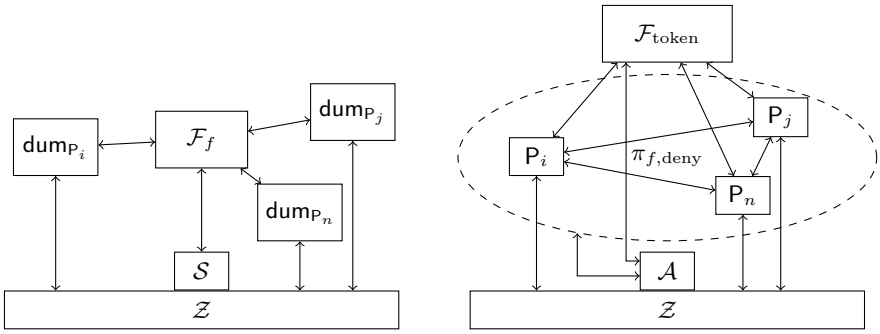
- Send message $(\texttt{"encrypt"}, x_i, i)$ to $\mathcal{F}_{\text{token}}$ and receive $c_{\text{in},i}$.
- Send $c_{\text{in},i}$ and received key shares to π_{MPC}.

On message $c_{\text{out},i}$ from π_{MPC}:

- Send $(\texttt{"decrypt"}, c_{\text{out},i})$ to $\mathcal{F}_{\text{token}}$ and receive y_i.
- Output y_i.

The protocol π_{MPC} can be any protocol which UC-realizes $\mathcal{F}_{\text{MPC}}$, *e.g.*, the CLOS protocol [Can+02]. Figure 2 illustrates the interplay between the protocol parties and the hybrid token functionality $\mathcal{F}_{\text{token}}$ in the real world and the ideal functionality $\mathcal{F}_f$ and the dummy parties in the ideal world. The protocol $\pi_{f,\text{deny}}$ enhances π_{MPC} by making it incoercible. To achieve incoercibility, it uses the hybrid functionality $\mathcal{F}_{\text{token}}$. $\mathcal{F}_f$ is therefore realized incoercibly.

Theorem 3. *Let π_{MPC} be an n-party protocol which UC-realizes $\mathcal{F}_{\text{MPC}}$. Assuming SKE is an IND-CPA secure, perfectly correct encryption scheme and Σ is an*



(a) Ideal world (b) Real world

Fig. 2: Illustration of the components relevant in Thm. 3. The dummy party of party P is denoted with $\mathsf{dum}_{\mathsf{P}}$.

EUF-CMA secure signature scheme, $\pi_{f,\text{deny}}$ $(\mathsf{C}^{\text{A}}, \mathcal{DI}) - wIUC$ *realizes* $\mathcal{F}_f$ *in the* $\mathcal{F}_{\text{token}}$*-hybrid model for every* $\mathcal{DI}$ *under static corruption, if at least one party is neither corrupted nor coerced.*

The proof of Thm. 3 consists of two parts, the first one showing that Eq. (1) of Def. 4 is fulfilled and the second one proving the indistinguishability in Eq. (2). For the first part, we start by defining a simulator. In a nutshell, the simulator intercepts all calls to $\mathcal{F}_{\text{token}}$ and simulates them in its head instead. Since it emulates $\mathcal{F}_{\text{token}}$ itself, it knows the symmetric keys and can decrypt coerced and corrupted parties' inputs. Actively coerced parties are treated as corrupted parties by the simulator, because the environment controls their interfaces via the adversary. Compared to the simulator for the token protocol under active coercion by Alwen et al. [Alw+15], there is no need to generate messages consistent with the output of $\mathcal{F}_f$ for honest parties, since the output is only sent to the parties in an encrypted way. This simulator is the same one in the ideal coercion world as it is in the ideal deception world. Since π_{MPC} UC-realizes $\mathcal{F}_{\text{MPC}}$, there exists a simulator $\mathcal{S}_{\text{MPC}}$ which proves this relation and can be used in a black-box manner in the proof.

In the second part, we show that there is a real deception strategy DR for every ideal deception strategy $\mathsf{DI} \in \mathcal{DI}$, such that the real and ideal deception worlds are indistinguishable. We define the real deception strategy to essentially use the same mappings as the ideal deception strategy. With another set of hybrid games, we show how the distinguishing advantage of the environment changes after taking over parts of the deception strategy. We end up with a modified environment trying to distinguish between the coercion experiments, which have been shown to be indistinguishable at this point. The proof is inspired by the ones by Alwen et al. [Alw+15]. We defer the full proof to App. A.

Remark 2. One important property of the token is that it belongs to a specific party. Under active coercion, incoercibility could not be retained, if the tokens

were to be replaced with enclaves (Trusted Execution Environments). Enclave models such as the one presented by Pass, Shi, and Tramèr [PST17] do not tie an enclave to the party hosting the enclave. Even though an active coercer could not monitor the communication between a party and their enclave, a malicious party could spawn a new enclave with the same enclave program and input the coercion input into the enclave. The environment could then force the coerced party to resume the protocol with the outputs of this enclave.

In the case of semi-honest coercion, on the other hand, enclaves could be used instead of hardware tokens, since the coercer does not have any influence on the coerced party during the protocol execution.

3.3 Impersonating Hardware Tokens

The hardware tokens used as a setup for the protocol $\pi_{f,\text{deny}}$ presented above as well as for its predecessor introduced by Alwen et al. [Alw+15] are assumed to be generated honestly. Relaxing this assumption, one might argue that the hardware tokens offer a new attack surface. They do not only learn the (possibly confidential) inputs of honest parties, but they can also replace them. A party cannot check if the token really encrypted their input or something else. It might as well have replaced their input by an adversarially chosen one, essentially impersonating the party in the computation.

One approach to mitigate these kinds of attacks is to use two hardware tokens per protocol participant. Instead of sending the input to a single token for encryption, every party now does the following steps:

1. Sample a MAC key.
2. Compute a MAC tag for the input using the MAC key.
3. Sample a 2-out-of-2 secret sharing of the input concatenated with the MAC tag.
4. Send one share to token A and the other one to token B.
5. Receive the encrypted shares from the tokens and send them to the other protocol participants together with secret shares of the MAC key.

In the MPC, the parties' input shares are then decrypted, the input is reconstructed from the shares, the MAC key is reconstructed as well, and the validity of the MAC tag is verified.

Assuming that the MAC scheme is EUF-CMA secure and at least one of the tokens is honest, a corrupted token only learns one of the shares and can therefore not reconstruct the input. If the token behaves maliciously and does not encrypt the share it received as an input, but some other share, the MAC tag becomes invalid with overwhelming probability. This, in turn, causes the verification of the MAC tag to fail in the MPC and for the computation to abort.

While using additional tokens protects the confidentiality and integrity of the inputs used for the computation even if one token is corrupted, the same does not hold for incoercibility. Even when each party uses two hardware tokens, incoercibility cannot be retained if one of the tokens is corrupted, as parties can only lie about inputs they made to an honest token and they may not know which token is honest and, in turn, which input share they can lie about.

4 Plausible Deniability

This section presents the second contribution of the paper: a definition for plausible deniability, which helps a coerced party quantify their risk upon deceiving the coercer. We show how this definition relates to the leakage of differentially private mechanisms and discuss how the plausibility degrades upon composition.

4.1 Defining Plausible Deniability

Incoercible protocols guarantee that a coercer cannot prevent coerced parties from deceiving them. However, the deceiving party's input might lead to a different output than the input the party would use upon coercion. This means that the coercer expects a certain output but receives a different one. This discrepancy can be quantified with the distinguishing advantage of the environment between the ideal execution with coercion and the ideal execution with deception, called Δ. We say that if a party can plausibly, except with a probability of Γ or less, deny which input they used, the protocol is Γ-deniable.

Definition 6 (Γ-Deniability for incoercible protocols). *Let π be a protocol that $(\mathsf{C}, \mathcal{DI}) - wIUC$ realizes an ideal functionality $\mathcal{F}$. Let* $\text{IDEAL}(\mathcal{F})$ *denote the ideal protocol of ideal functionality $\mathcal{F}$. Let $\mathcal{Z}$ be an environment, where $\mathcal{Z} \in \zeta$. Let* $\max_{\mathcal{Z}\in\zeta,\mathsf{DI}_i\in\mathcal{DI}} |\Pr[1 \leftarrow \mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{dum}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z})] - \Pr[1 \leftarrow \mathsf{Exec}(\text{IDEAL}(\mathcal{F}), \mathsf{DI}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z})]| =: \Delta$. *We say that π is Γ-deniable if the following holds:*

$$\Delta \leq \Gamma. \tag{3}$$

For this definition, we only consider parties that want to plausibly deny using a certain input and receiving a certain output. We do not consider abstention attacks, where a party wants to deny their participation in the protocol. This also includes environments that coerce a party into using an input that leads to an abort of the protocol. Abstention attacks are different in nature and a party could only plausibly deny participating in a protocol, if the environment was not able to observe the network.

Remark 3. If the above definition is fulfilled, it follows for the real coercion and deception worlds that $\max_{\mathcal{Z}\in\zeta,\mathsf{DI}_i\in\mathcal{DI}} |\Pr[1 \leftarrow \mathsf{Exec}(\pi, \mathsf{C}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z})] - \Pr[1 \leftarrow \mathsf{Exec}(\pi, \mathsf{DR}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z})]| \leq \Delta + \mathsf{negl}(\kappa)$.

While we want Δ to be small enough that the environment cannot distinguish between coercion and deception, a value of $\Delta \leq \mathsf{negl}$ would imply that the coerced party's input to the protocol would only negligibly affect the output or that the coercion input is the same as the deception input, making their participation in the protocol meaningless.

Consider a voting scheme where parties can vote for one of two choices and the result is revealed. An environment coercing one party into voting a certain way and knowing the inputs of all other parties can trivially distinguish the

coercion and the deception world by looking at the outcome of the election. In order to still allow for a small Δ while allowing each party's input to meaningfully contribute to the outcome of the computation, we restrict the set of environments considered for the notion of Γ-deniability to environments in a set ζ. One such set ζ could for example contain all environments that, while still choosing honest parties' inputs, "forget" these inputs after they were made.

When considering the Γ-deniability of $\pi_{f,\text{deny}}$ (Sect. 3.2), the protocol can be further improved by requiring the symmetric encryption scheme SKE to guarantee ciphertext integrity (INT-CTXT) in addition to IND-CPA security. While this is not necessary for the proof of the weak UC incoercibility of the protocol to go through, the environment can maximize its distinguishing advantage by exploiting a weak encryption scheme.

Ciphertext integrity is a property of symmetric encryption schemes preventing the adversary from creating a fresh ciphertext which can be decrypted successfully. If the chosen encryption scheme is IND-CPA, but not INT-CTXT secure, the environment could generate a valid ciphertext c and coerce parties into using c in the MPC protocol instead of the one from the token. As all messages that are sent over the network are visible by the adversary, the only way one can hope to achieve a deniability with small Δ is for the party to use c. If the chosen encryption scheme was also INT-CTXT-secure, the environment could not produce such a valid ciphertext. Thus, either the same ciphertext corresponding to the party's input to the token is used, or the ciphertext is invalid, leading to the honest party aborting the protocol execution.

The exact Γ-deniability of $\pi_{f,\text{deny}}$ using an INT-CTXT secure encryption scheme strongly depends on the computed function f and the set of considered environments ζ. Let us assume, *e.g.*, a function f, which computes the logical OR of its inputs. We restrict the set of environments ζ to those which do not know the inputs to honest parties. These are instead drawn uniformly at random.

To determine the environment's advantage Δ in distinguishing between ideal coercion and deception, we consider the environment $\mathcal{Z} \in \zeta$, which maximizes this advantage. Since the environment can control all inputs of corrupted and coerced parties and it knows the output of the computation, it chooses to corrupt or coerce as many parties as it can. By assumption, one party remains honest. To further maximize its advantage, the environment can instruct the corrupted and actively coerced parties to input a 0 into the functionality.

Since the environment gains more information from a party via a corruption than a coercion, we assume that the environment chooses to instruct the adversary to coerce one party. Under coercion, this party is going to input a 0, and considering deception, the deception strategy is going to replace this input with a 1. If the randomly drawn input of the honest party is 1, the output is going to be 1, independent of the coerced party's input. The environment would not be able to distinguish between coercion and deception in this case. If the input of the honest party is 0 instead, the output of the execution in the ideal coercion world is going to be 0, while the output in the ideal deception world is

going to depend on the deception input. In this case, it would be trivial for the environment to distinguish between coercion and deception.

Overall, the advantage of the environment in distinguishing between coercion and deception is $\Delta = 0.5$, making the protocol 0.5-deniable. The following section considers a special class of functions f, namely differentially private ones.

4.2 Differential Privacy

Differential privacy is a notion that is typically used in the context of databases. If fulfilled, differential privacy ensures that no information about individuals is leaked when analyzing a database. Put more formally, given a database DB with a private entry, the result of analyzing the database using a randomized mechanism M with the private entry relates to the result of analyzing the database using the same mechanism without the private entry as follows [Dwo+06b]:

$$\forall T \subseteq \mathcal{Y} \ \Pr[\mathsf{M}(\mathsf{DB}) \in T] \leq \exp(\epsilon) \cdot \Pr[\mathsf{M}(\mathsf{DB}') \in T] + \delta, \tag{4}$$

where the mechanism M maps into a set in $\mathcal{Y}$. This mechanism is then said to be (ϵ, δ)-differentially private, where $\epsilon \geq 0$ and $\delta \in [0, 1]$.

Theorem 4. *Let π be a protocol which $(\mathtt{C}, \mathcal{DI}) - wIUC$ realizes $\mathcal{F}_f$. If f is an (ϵ_q, δ_q)-differentially private mechanism in round q, then π is $(\exp(\epsilon) - 1 + \delta)$-deniable with*

$$\epsilon \coloneqq \sum_{q=1}^{R} \epsilon_q \cdot |\mathcal{I}|, \tag{5}$$

$$\delta \coloneqq \sum_{q=1}^{R} \frac{e^{|\mathcal{I}| \cdot \epsilon_q} - 1}{e^{\epsilon_q} - 1} \delta_q, \tag{6}$$

where $|\mathcal{I}|$ is the number of coerced parties and R is the number of rounds of computation.

Proof. Assuming that f is an (ϵ_q, δ_q)-differentially private mechanism in round q, Eq. (4) holds for $\mathsf{DB} \coloneqq \{x_i\}_{i \in [n]}$ in the respective round. According to Guerra-Balboa et al. [GB+24, Theorem VI.5], the adaptive sequential composition of differentially private mechanisms leads to f being $(\sum_{q=1}^{R} \epsilon_q, \sum_{q=1}^{R} \delta_q)$-differentially private, where R is the total number of rounds. The outputs of adaptive-composed mechanisms can depend on the outputs of all previously computed mechanisms.

Changing one input according to the coercer or the deception strategy corresponds to the databases differing in one entry, producing a leakage of ϵ_q and δ_q. In our setting, multiple parties can be coerced at the same time. This corresponds to changing multiple entries in the database, also known as group differential privacy. In such a group differential privacy setting, we effectively get $\epsilon = k \cdot \epsilon_q$ and $\delta = (e^{k\epsilon_q} - 1)/(e^{\epsilon_q} - 1) \cdot \delta_q$ [SU17, Fact 2.2]. Setting k to be the number of coerced

parties $\mathcal{I}$ and considering multiple different entries in the setting of adaptive sequential composition, we arrive at ϵ and δ defined in Eqs. (5) and (6). Since the environment does not have any other advantage at distinguishing coercion from deception than this leakage, Eq. (3) holds. This concludes the proof.

Interestingly, Thm. 4 holds without any restrictions on the allowed deception strategies and environments. Since the differential privacy is an inherent property of the mechanism that is computed, the deception strategy only has to replace the coerced party's input. On the other hand, any mapping the deception strategy of a party could apply to their input would only affect this single input, which is covered by differential privacy and therefore would not lead to an increase in leakage. For the party to enjoy the differential privacy guarantees, no inputs of other parties have to be hidden from the environment.

4.3 Composability

While the incoercibility notion of Def. 4 is defined in the UC framework and guarantees composability, Γ-deniability does not compose in a similar manner. Although we require that a Γ-deniable protocol π is $\mathcal{DI}$-weak UC incoercible, we lose composability as soon as we restrict Δ and ζ. Let us again consider, *e.g.*, a randomized response scheme, where a party rolls dice before answering questions in a survey. The party is protected by the scheme in the sense that they can lie about answering with `yes`, since they could have rolled a six instead of answering truthfully. However, if the survey would be repeated, this excuse would become less plausible each time. Therefore, the plausibility of the deception declines with every execution, despite the composability of the incoercibility notion.

This intuition has already been formalized in the context of differential privacy [Dwo+06a; DRV10; KOV15; MV16; VW21; GB+24]. Under *basic composition*, if η (ϵ, δ)-differentially private mechanisms $\mathsf{M}_0, ..., \mathsf{M}_{\eta-1}$ are executed independently, their composition is $(\eta\epsilon, \eta\delta)$-differentially private [Dwo+06a], which means that the leakage grows linearly with the number of mechanisms. Defining the leakage of the composed mechanism as $\epsilon' \coloneqq \eta\epsilon$ and $\delta' \coloneqq \eta\delta$, Thm. 4 can again be applied. If f is not a differentially private mechanism, we cannot make such statements about Γ-deniability in general.

5 Conclusion and Outlook

Incoercibility is an important security property in many contexts like voting and whistleblowing. While previous models in the Universal Composability framework have overlooked the central problem of quantifying the risk of a deception being detected by the coercer, we presented a definition of plausible deniability. We further analyzed the deniability of differentially private mechanisms and enhanced the protocol of Alwen et al. [Alw+15], which uses honestly generated hardware tokens to make generic protocols incoercible. To improve the plausible deniability of the protocol, we added a mechanism for output deniability.

We plan to further investigate measures against corrupted hardware tokens in the future, also considering solutions that do not require additional tokens.

Acknowledgments. This work was funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[Alw+15] Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas. "Incoercible Multi-party Computation and Universally Composable Receipt-Free Voting". In: *Advances in Cryptology – CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2015, pp. 763–780. DOI: 10.1007/978-3-662-48000-7_37.

[Ara+22] Diego F. Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. "Laconic Private Set-Intersection From Pairings". In: *ACM CCS 2022: 29th Conference on Computer and Communications Security*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. Los Angeles, CA, USA: ACM Press, 2022, pp. 111–124. DOI: 10.1145/3548606.3560642.

[Bay+26] Saskia Bayreuther, Robin Berger, Felix Dörre, Eva Hetzel, Yufan Jiang, Christian Martin, Jeremias Mechler, and Jörn Müller-Quade. *More from Less: Composable General Multi-Party Computation with Global Public Verifiability from a Single Enclave Only*. Cryptology ePrint Archive, Report 2026/1005. 2026. URL: https://eprint.iacr.org/2026/1005.

[BL26] Magdalena Bertram and Anja Lehmann. *Deniability for Signed Credentials: Revisiting the Authenticated Channel vs. Signed Data Debate in the EUDI Wallet*. Cryptology ePrint Archive, Report 2026/1648. 2026. URL: https://eprint.iacr.org/2026/1648.

[BMQR07] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. "Bingo voting: Secure and coercion-free voting using a trusted random number generator". In: *E-Voting and Identity: First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers 1*. Springer. 2007, pp. 111–124.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[Can+02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. "Universally composable two-party and multi-party secure computation". In: *34th Annual ACM Symposium on Theory of Computing*. Montréal, Québec, Canada: ACM Press, 2002, pp. 494–503. DOI: 10.1145/509907.509980.

[Can20] Ran Canetti. "Universally Composable Security". In: *J. ACM* 67.5 (2020), pp. 1–94. DOI: 10.1145/3402457.

[CG96] Ran Canetti and Rosario Gennaro. "Incoercible Multiparty Computation (extended abstract)". In: *37th Annual Symposium on Foundations of Computer Science*. Burlington, Vermont: IEEE Computer Society Press, 1996, pp. 504–513. DOI: 10.1109/SFCS.1996.548509.

[CGP15] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. "Adaptively Secure Two-Party Computation from Indistinguishability Obfuscation". In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer Berlin Heidelberg, Germany, 2015, pp. 557–585. DOI: 10.1007/978-3-662-46497-7_22.

[CP20] Ran Canetti and Oxana Poburinnaya. "Towards Multiparty Computation Withstanding Coercion of All Parties". In: *TCC 2020: 18th Theory of Cryptography Conference, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. Lecture Notes in Computer Science. Durham, NC, USA: Springer, Cham, Switzerland, 2020, pp. 410–438. DOI: 10.1007/978-3-030-64378-2_15.

[Dam+25] Aditya Damodaran, Simon Rastikian, Peter B. Rønne, and Peter Y. A. Ryan. "Hyperion: Transparent End-to-End Verifiable Voting with Coercion Mitigation". In: *ESORICS 2025: 30th European Symposium on Research in Computer Security, Part II*. Ed. by Vincent Nicomette, Abdelmalek Benzekri, Nora Boulahia-Cuppens, and Jaideep Vaidya. Vol. 16054. Lecture Notes in Computer Science. Toulouse, France: Springer, Cham, Switzerland, 2025, pp. 124–143. DOI: 10.1007/978-3-032-07891-9_7.

[Dod+09] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. "Composability and On-Line Deniability of Authentication". In: *TCC 2009: 6th Theory of Cryptography Conference*. Ed. by Omer Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Germany, 2009, pp. 146–162. DOI: 10.1007/978-3-642-00457-5_10.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. "Boosting and Differential Privacy". In: *51st Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, 2010, pp. 51–60. DOI: 10.1109/FOCS.2010.12.

[Dwo+06a] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our Data, Ourselves: Privacy Via Dis-

tributed Noise Generation". In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer Berlin Heidelberg, Germany, 2006, pp. 486–503. DOI: 10.1007/11761679_29.

[Dwo+06b] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *TCC 2006: 3rd Theory of Cryptography Conference*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. New York, NY, USA: Springer Berlin Heidelberg, Germany, 2006, pp. 265–284. DOI: 10.1007/11681878_14.

[FJ24] Rune Fiedler and Christian Janson. "A Deniability Analysis of Signal's Initial Handshake PQXDH". In: *Proceedings on Privacy Enhancing Technologies* 2024.4 (Oct. 2024), pp. 907–928. DOI: 10.56553/popets-2024-0148.

[FL25] Rune Fiedler and Roman Langrehr. "On Deniable Authentication Against Malicious Verifiers". In: *Advances in Cryptology – CRYPTO 2025, Part VIII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16007. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2025, pp. 3–38. DOI: 10.1007/978-3-032-01913-4_1.

[GB+24] Patricia Guerra-Balboa, Àlex Miranda-Pascual, Javier Parra-Arnau, and Thorsten Strufe. "Composition in Differential Privacy for General Granularity Notions". In: *CSF 2024: IEEE 37th Computer Security Foundations Symposium*. Enschede, The Netherlands: IEEE Computer Society Press, 2024, pp. 680–696. DOI: 10.1109/CSF61375.2024.00004.

[KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The composition theorem for differential privacy". In: *International conference on machine learning*. PMLR. 2015, pp. 1376–1385.

[Kuh+21] Christiane Kuhn, Maximilian Noppel, Christian Wressnegger, and Thorsten Strufe. "Plausible deniability for anonymous communication". In: *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*. 2021, pp. 17–32.

[MV16] Jack Murtagh and Salil P. Vadhan. "The Complexity of Computing the Optimal Composition of Differential Privacy". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer Berlin Heidelberg, Germany, 2016, pp. 157–175. DOI: 10.1007/978-3-662-49096-9_7.

[Par+24] Seongho Park, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. *zkVoting : Zero-knowledge proof based coercion-resistant and E2E verifiable e-voting system*. Cryptology ePrint Archive, Report 2024/1003. 2024. URL: https://eprint.iacr.org/2024/1003.

[PST17] Rafael Pass, Elaine Shi, and Florian Tramèr. "Formal Abstractions for Attested Execution Secure Processors". In: *Advances in Cryp-*

tology – EUROCRYPT 2017, Part I. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Paris, France: Springer, Cham, Switzerland, 2017, pp. 260–289. DOI: 10.1007/978-3-319-56620-7_10.

[SU17] Thomas Steinke and Jonathan Ullman. "Between Pure and Approximate Differential Privacy". In: *Journal of Privacy and Confidentiality* 7.2 (2017), pp. 3–22. DOI: 10.29012/jpc.603. URL: https://arxiv.org/abs/1501.06095.

[UG15] Nik Unger and Ian Goldberg. "Deniable Key Exchanges for Secure Messaging". In: *ACM CCS 2015: 22nd Conference on Computer and Communications Security*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. Denver, CO, USA: ACM Press, 2015, pp. 1211–1223. DOI: 10.1145/2810103.2813616.

[UG18] Nik Unger and Ian Goldberg. "Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging". In: *Proceedings on Privacy Enhancing Technologies* 2018.1 (Jan. 2018), pp. 21–66. DOI: 10.1515/popets-2018-0003.

[UM10] Dominique Unruh and Jörn Müller-Quade. "Universally Composable Incoercibility". In: *Advances in Cryptology – CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2010, pp. 411–428. DOI: 10.1007/978-3-642-14623-7_22.

[VW21] Salil P. Vadhan and Tianhao Wang. "Concurrent Composition of Differential Privacy". In: *TCC 2021: 19th Theory of Cryptography Conference, Part II*. Ed. by Kobbi Nissim and Brent Waters. Vol. 13043. Lecture Notes in Computer Science. Raleigh, NC, USA: Springer, Cham, Switzerland, 2021, pp. 582–604. DOI: 10.1007/978-3-030-90453-1_20.

[War65] Stanley L Warner. "Randomized response: A survey technique for eliminating evasive answer bias". In: *Journal of the American statistical association* 60.309 (1965), pp. 63–69.

A Proof of Theorem 3

In this section, we provide a formal proof of the weak UC incoercibility of the protocol $\pi_{f,\text{deny}}$ presented in Sect. 3 under active coercion. For convenience, we repeat the theorem here.

Theorem 3 *Let π_{MPC} be an n-party protocol which UC-realizes $\mathcal{F}_{\text{MPC}}$. Assuming SKE is an IND-CPA secure, perfectly correct encryption scheme and Σ is an EUF-CMA secure signature scheme, $\pi_{f,\text{deny}}$ $(\mathsf{C}^{\text{A}}, \mathcal{DI}) - wIUC$ realizes $\mathcal{F}_f$ in the $\mathcal{F}_{\text{token}}$-hybrid model for every $\mathcal{DI}$ under static corruption, if at least one party is neither corrupted nor coerced.*

To prove the theorem, we first define the simulator. We show the indistinguishability between the ideal coercion execution and the real coercion execution via hybrid games. This corresponds to the expression in Eq. (1) of Def. 4. In a second step, we show how to derive a real deception strategy from a given ideal one. Using the same simulator, this shows that Eq. (2) is fulfilled as well.

Proof. **Definition 7 (Simulator $\mathcal{S}$).**

- *On message* $(\texttt{"keyGen"}, i)$ *from corrupted or coerced* P_i *to* $\mathcal{F}_{\text{token}}$*:*
  - *Sample* $k_i \leftarrow \mathsf{SKE.KeyGen}(\kappa)$.
  - *If the signature and verification keys have not been sampled yet, sample* $(sk, vk) \leftarrow \Sigma.\mathsf{KeyGen}'(\kappa)$.
  - *Compute n-out-of-n authenticated sharing* $\{\langle k_i \rangle_j\}_{j \in [n]}$ *of* k_i *with the signing key* sk.
  - *Send* $\texttt{"ok"}$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
- *On message* $(\texttt{"keyShare"}, j, i)$ *from corrupted or coerced* P_i *to* $\mathcal{F}_{\text{token}}$*:*
  - *If key* k_j *has not been sampled, send* $\perp$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
  - *Otherwise, send* $\langle k_j \rangle_i$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
- *On notification of input with length* ℓ *from honest party* P_i *to* $\mathcal{F}_f$*:*
  - *If* k_i *has not been sampled, sample* $k_i \leftarrow \mathsf{SKE.KeyGen}(\kappa)$.
  - *Draw* $\hat{x}_i \xleftarrow{\$} \{0,1\}^\ell$.
  - *Compute and send* $c_{\text{in},i} = \mathsf{SKE.Enc}(k_i, \hat{x}_i)$ *and the key shares* $\langle k_j \rangle_i$ *to* π_{MPC} *in the name of* P_i.
  - *Send an acknowledgment of the notification to* $\mathcal{F}_f$.
- *On message* $(\texttt{"encrypt"}, x_i, i)$ *from corrupted or coerced* P_i *to* $\mathcal{F}_{\text{token}}$*:*
  - *If* k_i *has not been sampled, send* $\perp$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
  - *Otherwise, compute and send* $c_{\text{in},i} = \mathsf{SKE.Enc}(k_i, x_i)$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
- *On message* $(c_{\text{in},i}, \{\langle k_j \rangle_i\}_{j \in [n]})$ *from corrupted or coerced* P_i *to* π_{MPC}*, continue with simulator* $\mathcal{S}_{\text{MPC}}$. $\mathcal{S}_{\text{MPC}}$ *treats all coerced parties as corrupted parties. Emulate* $\mathcal{F}_{\text{MPC}}$ *for* $\mathcal{S}_{\text{MPC}}$ *once all shares have been received:*
  - *Reconstruct the keys* $\{k_i\}$ *of corrupted and coerced parties and decrypt the ciphertexts with the extracted key shares:* $x_i := \mathsf{SKE.Dec}(k_i, c_{\text{in},i})$. *Abort if the reconstruction or decryption fails.*
  - *Send the decrypted inputs of corrupted and coerced parties to* $\mathcal{F}_f$ *in the name of the respective parties and receive the output* out *from* $\mathcal{F}_f$ *for* P_i.

  - *Compute* $c_{\text{out},i} := \mathsf{SKE.Enc}(k_i, \mathsf{out})$ *and send it to* P_i *in the name of* π_{MPC}.
- *On message* $(\texttt{"decrypt"}, c_{\text{out},i})$ *from corrupted or coerced* P_i *to* $\mathcal{F}_{\text{token}}$*:*
  - *If* k_i *has not been sampled, send* $\perp$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.
  - *Otherwise, compute and send* $y_i = \mathsf{SKE.Dec}(k_i, c_{\text{out}})$ *to* P_i *in the name of* $\mathcal{F}_{\text{token}}$.

We show the indistinguishability of the following hybrids. Let out_i denote the output of the environment in hybrid G_i.

G_0: Execution with ideal functionality $\mathcal{F}_0 = \mathcal{F}_f$ and the simulator $\mathcal{S}_0 = \mathcal{S}$.

G_1: Execution with an ideal functionality $\mathcal{F}_1$ that behaves the same as $\mathcal{F}_0$, except that it tells the inputs of honest parties to the simulator. The simulator $\mathcal{S}_1$ behaves the same as $\mathcal{S}_0$, except that it also learns the input x_i from $\mathcal{F}_1$ and therefore forwards the real input to $\mathcal{F}_f$. The result of the evaluation performed by $\mathcal{F}_f$ on the honest parties' real inputs is then used by $\mathcal{S}_1$ to compute the output ciphers.

G_2: Execution with an ideal functionality $\mathcal{F}_2$ that behaves the same as $\mathcal{F}_1$. The simulator $\mathcal{S}_2$ behaves the same as $\mathcal{S}_1$, except that it no longer encrypts a randomly sampled value $\hat{x}_i$, but it encrypts a share of the real input x_i instead.

G_3: Execution with ideal functionality $\mathcal{F}_3$, that no longer generates outputs based on f and previous inputs but lets the adversary determine the output. $\mathcal{S}_3$ uses f on the real inputs to compute these outputs.

G_4: Execution with an ideal functionality $\mathcal{F}_4 = \mathcal{F}_3$. Compared to $\mathcal{S}_3$, $\mathcal{S}_4$ does not directly use the inputs to evaluate f, but it does the following pre- and postprocessing: It reconstructs the previously drawn and shared keys. It then encrypts the inputs under these keys, decrypts them, evaluates f and computes the output ciphers. These are then decrypted again. Other than that, it behaves the same as $\mathcal{S}_3$.

G_5: Execution with an ideal functionality $\mathcal{F}_5 = \mathcal{F}_4$. $\mathcal{S}_5$ no longer simulates π_{MPC} itself, but the simulation of the protocol is replaced with an invocation of π_{MPC} with a dummy adversary.

G_6: Execution with an ideal functionality $\mathcal{F}_6 = \mathcal{F}_5$. Instead of intercepting calls to $\mathcal{F}_{\text{token}}$ by the parties participating in π_{MPC}, $\mathcal{S}_6$ now passes through these calls in the case of honest parties or makes the calls itself in the name of corrupted and coerced parties.

G_7: Real execution with $\pi_{f,\text{deny}}$, $\mathcal{F}_{\text{token}}$, and the dummy adversary.

Claim 1. Assuming Σ is an EUF-CMA secure signature scheme, out_0 and out_1 are computationally indistinguishable.

Proof. The environment could only distinguish between G_0 and G_1 if the output of $\mathcal{F}_f$ was different. This could be caused by a corrupted party inputting a false key share including a fake signature under the signing key of $\mathcal{F}_{\text{token}}$, since the verification keys provided with all shares have to match and at least one party is honest. If a corrupted party was able to fake a signature, they would be able to win the EUF-CMA game, directly contradicting the assumption that the signa-

ture scheme is EUF-CMA secure. Therefore, out_0 and out_1 are computationally indistinguishable.

Claim 2. Assuming SKE is an IND-CPA secure encryption scheme, out_1 and out_2 are computationally indistinguishable.

Proof. Let $\mathcal{C}$ be the challenger in the IND-CPA game and $\mathcal{A}$ the adversary. To determine the choice bit b', $\mathcal{A}$ internally runs G_1 and G_2. Assume, $\mathcal{A}$ can distinguish these games with non-negligible probability, then $\mathcal{A}$ has non-negligible probability in winning the IND-CPA game. This is a contradiction to the assumption that SKE is an IND-CPA secure encryption scheme, which concludes this proof.

Claim 3. out_2 and out_3 are indistinguishable.

Proof. G_2 and G_3 are identical, except that now the evaluation of f determining the output of parties is no longer done by $\mathcal{F}_2$, but instead by the simulated protocol π_{MPC} on the same inputs. Since the same inputs are used, the output of f and therefore the output to honest parties and the output to corrupted and coerced parties, that is then further used during simulation is identical to the output in $\mathcal{F}_2$. Therefore, the outputs of the two games are distributed identically.

Claim 4. Assuming SKE is a perfectly correct encryption scheme, out_3 and out_4 are distributed identically.

Proof. Since the encryption scheme is perfectly secure and the correct keys are used, the inputs and outputs are not changed after encryption and decryption.

Claim 5. out_4 and out_5 are indistinguishable.

Proof. Since π_{MPC} UC-realizes $\mathcal{F}_{\mathrm{MPC}}$, there exists a simulator $\mathcal{S}_{\mathrm{MPC}}$ such that the environment cannot distinguish between a simulation and an invocation of the protocol.

Claim 6. out_5 and out_6 are distributed identically.

Proof. The simulator emulates $\mathcal{F}_{\mathrm{token}}$ faithfully in G_5. Since it samples the keys, computes the sharings, decrypts, and encrypts the parties' real inputs and the outputs of $\mathcal{F}_f$ as the token would, its computations can be performed by $\mathcal{F}_{\mathrm{token}}$ itself and out_5 and out_6 are distributed identically.

Claim 7. out_6 and out_7 are distributed identically.

Proof. The difference between these games is only syntactical. Therefore, this claim follows directly from the definition of the hybrid games.

Using Claim 1 to 7, we know that $\mathsf{G}_0 \overset{c}{\approx} \mathsf{G}_7$.

Lastly, we show how to build a real deception strategy $\mathsf{DR}_i = (\mathsf{DR}_i^1, \mathsf{C}_i^{\mathrm{A}}(\pi_i), \mathsf{DR}_i^2)$ from a given ideal deception strategy $\mathsf{DI}_i = (\mathsf{DI}_i^1, \mathsf{dum}_{\mathsf{P}_i}, \mathsf{DI}_i^2) \in \mathcal{DI}$ for

every party P_i. Since we consider active coercion, the environment can directly instruct the coerced parties which protocol messages they should send. The adversary's messages to the coerced party are therefore not routed via DI_i^1 (or DR_i^1). Accordingly, we only need to specify DR_i^2. For this, we follow the same line of thought as in the security proof of the token protocol by Alwen et al. [Alw+15].

DR_i^2 forwards all messages between $\mathsf{C}_i^{\mathrm{A}}(\pi_i)$ and its hybrid functionalities, except for $\mathcal{F}_{\mathrm{token}}$, where it behaves as DI_i^2. We again show the indistinguishability of the following hybrids. Let α_i denote the advantage of the environment in hybrid G_i.

G_0: Experiments with an environment $\mathcal{Z}_0$ and distinguishing advantage $\alpha_0 := |\Pr[1 \leftarrow \mathsf{Exec}(\mathrm{IDEAL}(\mathcal{F}), \mathsf{DI}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z}_0)] - \Pr[1 \leftarrow \mathsf{Exec}(\pi, \mathsf{DR}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z}_0)]|$.

G_1: Experiments with the environment $\mathcal{Z}_1$, which behaves as $\mathcal{Z}_0$, but additionally emulates $\{\mathsf{DI}_i^2\}_{i\in\mathcal{I}}$.

G_2: Experiments $\mathsf{Exec}(\mathrm{IDEAL}(\mathcal{F}), \mathsf{dum}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z}_1)$ and $\mathsf{Exec}(\pi, \mathsf{C}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z}_1)$.

Claim 8. The distinguishing advantage α_1 is the same as α_0.

Proof. In the ideal deception execution $\mathsf{Exec}(\mathrm{IDEAL}(\mathcal{F}), \mathsf{DI}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z}_0)$ in G_0, every dummy party dum_i forwards protocol messages to DI_i^2 as instructed by the simulator. These are then manipulated by DI_i^2 and forwarded to the simulator's emulations of the hybrid functionalities. $\mathcal{Z}_1$ can therefore also directly change the instructions that dum_i receives according to DI_i^2.

In the real deception execution $\mathsf{Exec}(\pi, \mathsf{DR}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z}_0)$, the adversary's instructions are sent to the coerced parties $\mathsf{C}_i^{\mathrm{A}}(\pi_i)$. Since the environment is in control of the interfaces of actively coerced parties, it can instruct them to send messages to $\mathcal{F}_{\mathrm{token}}$ according to DI_i^2 and unchanged to other hybrid ideal functionalities.

Claim 9. $\alpha_2 = \alpha_1$.

Proof. The experiments in G_1 correspond to $\mathsf{Exec}(\mathrm{IDEAL}(\mathcal{F}), \mathsf{dum}_{\mathcal{I}}, \mathcal{S}, \mathcal{Z}_1)$ and $\mathsf{Exec}(\pi, \mathsf{C}_{\mathcal{I}}, \mathcal{A}, \mathcal{Z}_1)$, where the coercion strategy is a part of the environment $\mathcal{Z}_1$. These are the coercion experiments.

As has been shown previously, the distinguishing advantage of the environment between the coercion experiments is negligible. Using Claim 8 to 9, we know that α_0, the distinguishing advantage between the deception experiments, is negligible as well, which conlcudes the proof.