Incoercible Multi-party Computation and Universally Composable Receipt-Free Voting

Joël Alwen[1(✉)], Rafail Ostrovsky[2], Hong-Sheng Zhou[3], and Vassilis Zikas[4]

[1] IST Austria, Klosterneuburg, Austria
jalwen@ist.ac.at
[2] UCLA, Los Angeles, USA
rafail@cs.ucla.edu
[3] Virginia Commonwealth University, Richmond, USA
hszhou@vcu.edu
[4] ETH Zurich, Zurich, Switzerland
vzikas@inf.ethz.ch

Abstract. Composable notions of incoercibility aim to forbid a coercer from using anything beyond the coerced parties' inputs and outputs to catch them when they try to deceive him. Existing definitions are restricted to weak coercion types, and/or are not universally composable. Furthermore, they often make too strong assumptions on the knowledge of coerced parties—e.g., they assume they known the identities and/or the strategies of other coerced parties, or those of corrupted parties—which makes them unsuitable for applications of incoercibility such as e-voting, where colluding adversarial parties may attempt to coerce honest voters, e.g., by offering them money for a promised vote, and use their own view to check that the voter keeps his end of the bargain.

In this work we put forward the first universally composable notion of incoercible multi-party computation, which satisfies the above intuition and does not assume collusions among coerced parties or knowledge of the corrupted set. We define natural notions of UC incoercibility corresponding to standard coercion-types, i.e., receipt-freeness and resistance to full-active coercion. Importantly, our suggested notion has the unique property that it builds *on top* of the well studied UC framework by Canetti instead of modifying it. This guarantees backwards compatibility, and allows us to inherit results from the rich UC literature.

We then present MPC protocols which realize our notions of UC incoercibility given access to an arguably minimal setup—namely honestly generate tamper-proof hardware performing a very simple cryptographic operation—e.g., a smart card. This is, to our knowledge, the first proposed construction of an MPC protocol (for more than two parties) that is incoercibly secure *and* universally composable, and therefore the first construction of a universally composable receipt-free e-voting protocol.

Keywords: Multi-party computation · Universal composition · Receipt-freeness

V. Zikas—Research partly done while the author was at UCLA.

R. Gennaro and M. Robshaw (Eds.): CRYPTO 2015, Part II, LNCS 9216, pp. 763–780, 2015.
DOI: 10.1007/978-3-662-48000-7_37

1 Introduction

Secure multi-party computation (MPC) allows n mutually distrustful parties to securely perform some joint computation on their inputs even in the presence of cheating parties. To capture worst-case (collaborative) cheating, a central adversary is assumed who gets to corrupt parties and uses them to attack the MPC protocol. Roughly speaking, security requires that the computation leaks no information to the adversarial parties about the inputs and outputs of uncorrupted, aka honest, parties (privacy) and that the corrupted parties cannot affect the output any more than choosing their own inputs (correctness).

The seminal works on MPC [3,12,18,36] established feasibility for arbitrary functions and started a rich and still evolving literature. Along the way, additional desired properties of MPC were investigated. Among these, *universal composability* guarantees that the protocol preserve its security even when executed within an online adversarial environment, e.g., along-side other (potentially insecure) protocols. Various frameworks for defining universal composability have been suggested [2,30], with Canneti's UC framework [6] being the most common.

The above frameworks make use of the so called *simulation-based* paradigm for defining security which, in a nutshell, can be described as follows: Let f denote a specification of the task that the parties wish to perform. Security of a protocol Π for f is defined by comparing its execution with an ideal scenario in which the parties have access to a fully trusted third party, the *functionality*, which takes their inputs, locally computes f, and returns to the parties their respective outputs. More concretely, a protocol Π is secure if for any adversary $\mathcal{A}$ attacking Π, there exists an ideal adversary $\mathcal{S}$ attacking the above ideal evaluation scenario, which simulates the attack (and view) of $\mathcal{A}$ towards any environment $\mathcal{Z}$ that gets to choose the parties' inputs and see their outputs.[1]

Arguably, UC security captures most security guarantees that one would expect from a multi-party protocol. Nonetheless, it does not capture *incoercibility* a property which is highly relevant for one of a prototypical application of MPC, namely secure e-voting. Intuitively, incoercibility ensures that even when some party is forced (or *coerced*) by some external entity into executing a strategy other than its originally intender, e.g., coerced to use a different input or even a different protocol, then the party can disobey (i.e., deceive) its coercer, e.g., use its originally intended input, without the coercer being able to detect it.

In the special case of e-voting, where parties are voters, this would mean that a coercer, e.g., a vote buyer that offers a voter money in exchange of his vote for some candidate c, is not able to verify whether the voter indeed voted for c or for some other candidate. In other words, the voter cannot use his transcript as a receipt that he voted for c, which is why in the context of voting the above type of incoercibility is often referred to as *receipt-freeness*.

[1] In strong (UC) definitions, it is required that this simulation is sound even in an on-line manner, i.e., $\mathcal{S}$ is not only required to simulated the view of $\mathcal{A}$ but has to do so against an online environment that might talk to the adversary at any point.

Which guarantees can we expect from a general definition of incoercibility? Clearly, if the coercer can use the outputs of the function to be computed to check upon the coerced party it is impossible to deceive him. Considering our voting scenario (concretely, majority election) if there are two candidates c_1 and c_2 and a set V of voters with $|V| = 2m + 1$ for some m, and the coercer coercing $v_i \in V$ knows that half of the parties in $V \setminus \{v_i\}$ voted for c_1 and the other half voted for c_2, then v_i cannot deceive its coercer, as his input uniquely defines the outcome of the election. Therefore, composable notions of incoercibility [9,35] aim for the next best thing, namely allow the parties to deceive their coercer within the "space of doubt" that the computed function allows them. In other words, an informal description of incoercibility requires that the parties can deceive their coercer when they are executing the protocol as good as they can deceive someone who only observes the inputs and outputs of the computation.

Of course, the above intuition becomes tricky to formulate when the protocol is supposed to be incoercible and simultaneously tolerate malicious adversaries. There are several parameters to take into account when designing such a definition. In the following we sketch those that, in our opinion, are most relevant.

Coercion Type. This specifies the power that the coercer has on the coerced party. Here we one can distinguish several types of coercion: *I/O-coercion* allows the coercer to provide an input to the party and only use its output. This is the simplest (and weakest) form of coercion as it is implied by UC security. A stronger type is *receipt-freeness* or *semi-honest* coercion; here, the coercer gets to provide an input to the coerced party, but expects to see a transcript which is consistent to this input. This type corresponds to the notion of coercion introduced in [9,10] and abstracts the receipt-freeness requirement in the voting literature [19,20,23,27,28,31–33].[2] Finally, *active coercion* is the strongest notion of coercion, where the adversary instructs the coerced party which messages to send in the protocol and expects to see all messages he receives (also in an online fashion). This type of coercion has been considered, explicitly or implicitly, in the stand-alone setting (i.e., without universal composition) by Moran and Naor [32] and more recently in the UC setting by Unruh and Müller-Quade [35].

Adaptive vs. Static. As with corruption, we can consider coercers who choose the set of parties to coerce at the beginning of the protocol execution, i.e., in a *static* manner, or *adaptively* during the protocol execution depending on their view so far—e.g., by observing the views of other coerced parties.

Coercer/Deceiver-Collusions. The vast majority of works in the multi-party literature assumes a so called *monolithic* adversary who coordinates the actions of corrupted parties. This naturally captures the worst-case scenario in which cheaters work together to attack the protocol. Analogously, works on incoercible computation [9,10,32,35] assume a monolithic coercer, i.e., a single entity which is in charge of coordinating coerced parties. This has the following counter-intuitive side-effect: in order for a coerced party to be able to deceive any such

[2] For the special case of encryption, resiliency to semi-honest coercion corresponds to the well-known concept of *deniability* [8].

a monolithic coercer it needs to coordinate its deception strategy with other coerced (or with honest) parties. In fact, in recent universally composable notions of incoercibility this deceiver coordination is explicit. For example, in [35] an even stronger requirement is assumed: the coerced parties which attempt a deception know the identities and deception strategies of other coerced parties, and even the identities of all corrupted parties. This is an unrealistic assumption in scenarios such as e-voting, where a potential vote-seller is most likely oblivious to who is cheating or to who else is selling its vote.

In order to avoid the above counter intuitive situation, in this work we assume that deception (therefore also coercion) is local to each coerced party, i.e., coercers of different parties are not *by default* colluding. Alas, casting our definition in the UC framework makes coercer collusion explicit: Although coercers are local, they can still be coordinated via an external channel, e.g., through the environment. In fact, in our definition the worst-case environment implicitly specifies such a worst-case coercion scenario.

Informants and Dependency between Corruption and Deception. Another question which is highly relevant for incoercibility, is whether or not coerced parties know the identities of the cheaters/adversaries. In particular, a worst case coercion scenario is the one in which the coercer and the adversary work together to check on the coerced parties—stated differently, the coercer uses corrupted parties as informants against coerced parties to detect if they are attempting to deceive him. (In the context of receipt-free voting, this corresponds to checking the view/receipt of vote sellers against the corresponding views of malicious parties.) Clearly, if a coerced party knows who are the informants then it is easier to deceive its coercer. (This is the approach taken in [35], where the identities of corrupted parties are accessible to the deceivers via a special register.) Arguably, however, this is not a realistic assumption as it reduces the effect of using informants—a vote buyer is unlikely to tell the vote seller how he can check upon him. The modeling approach taken in this work implies that real-world deceivers have no information on who is corrupted (or coerced).

Our Contributions. In this work we provide the first security definition of incoercible multi-party computation which is universally composable (UC) and makes minimal assumptions on the coerced parties' ability to deceive their coercer. Our definition offers the same flexibility on addressing different classes of coercion as standard security notion offers for corruptions. Indicatively, by instantiating it with different types of coercion we devise definitions of UC incoercibility against *semi-honest* coercions—corresponding to the classical notion of receipt-freeness—as well as of the more powerful *active coercions* corresponding to the strong receipt-freeness notion introduced in [32]. As a sanity check, we show that if the coercers only see the output of coerced parties (a notion which we call I/O-incoercibility), then any UC secure protocol is also incoercible.

In addition to flexibility, our definition has the following intuitive properties:

Universal composability and compatibility with standard UC. We prove universal composition theorems for all the suggested types of incoercibility,

which imply that an incoercible protocol can be composed with any other incoercible protocol. Because our definition builds on top of the UC framework instead of modifying is (e.g., as in [10,35]), our protocols are automatically also universally composable with standard (coercible) UC protocols, at the cost, of course, of giving up incoercibility; that is, when composing an incoercibly UC protocol with a standard (coercible) UC protocol, we still get a UC secure protocol. We note in passing that defining incoercibility in UC has the additional advantage that it protects even against on-line coercer, e.g., vote-buyer that expect the receipt to be transmitted to them while the party is voting.

Minimal-knowledge deceptions. The deceivers in the real-world have no knowledge of who is coerced or corrupted, nor do they know which strategy other coerced parties will follow. Thus they need to deceive assuming that any party might be an informant.

Last but not least, we present a UC incoercible protocol for arbitrary multi-party computation which tolerates any number of actively corrupted and any number of coerced parties (for both semi-honest and active coercion), as long as there is at least one honest party. Our protocols make use of an arguably minimal and realistic assumption (see the discussion below), i.e., access to a simple honestly generated hardware token. To our knowledge, ours is the first protocol construction, which implements *any* functionality in the *multi-party* ($n > 2$ parties) setting. In fact, our construction can be seen a compiler, in this token-hybrid model, of UC secure to incoercible UC secure protocols. Therefore, when instantiated with a fast UC secure protocol it yields a realistic candidate for construction for UC secure incoercible e-voting.

Our protocol is proved secure against static coercion/corruption, but our proofs carry through (with minimal modifications) to the adaptive setting. In fact, our protocols realize an even stronger security definition in which the coercers, but *not* the coerced parties (i.e., the deceivers), might coordinate their strategies. However, we chose to keep the definition somewhat weaker, to leave space for more solutions or possibly different assumptions.[3]

The Ideal Token Assumption. Our protocols assume that each party has access to a hardware token which might perform fresh encryptions with to some hidden keys that are shared among the parties. The goal of the token is to offer the parties a source of hidden randomness that allows them to deceive their coercer. A setup of this type seems to be necessary for such a strong incoercibility notion when nearly everyone might be corrupted, since if the coerced parties have no external form of hidden randomness, then it seems impossible for them to deceive—the coercer might request their entire state and compare it with messages received by its informants, which would require the coerced party to align its lie with message it sends to the informants (whose the identities are unknown).

[3] Recall that our definition *does* allow coercer coordination through the environment.

On top of being minimal in the above sense, our encryption token assumption is also very easy to implement in reality for a system with a bounded number of participants—this is typically the case in elections: Let N be an upper bound on the voters; the voting registration authority (i.e., the token creator and distributor) computes N keys $k_1, \ldots, k_N$, one for every potential voter; every p_i who registers receives his ith token along with a vector of N random strings $(k_{1i}, \ldots, k_{Ni})$, corresponding to his keys-shares; the last p_i who registers (i.e., the last to be in the registration desk before it closes) receives his token, say the n-th token, along with the vector $(k_{1n}, \ldots, k_{Nn}) = (k_1, \ldots, k_n) \oplus \bigoplus_{i=1}^{n}(k_{1i}, \ldots, k_{Ni})$, where $\oplus$ denotes the component-wise application of the bit-wise xor operation. Note that the assumption of a hardware token (capturing pre-distributed smart cards) has been used extensively in practice, e.g., in the university elections in Austria and even the national elections in Finland and Estonia [14].

Related Literature. The incoercibility literature can roughly be split in two classes: works that look at the special case of receipt-free voting [1,4,15,16,19,20,22,23,26–28,31,33,34] and works that look at the more general problem of incoercible realization of arbitrary multi-party functions [9,10,32,35]. Below, we focus on the second class which is closer to our goal and refer the reader to the full version of this work for a short survey of the voting-specific literature.

The first to consider incoercibility in the setting of general MPC were Canneti and Gennaro [9]. They put forth a notion of incoercibility for static off-line semi-honest coercions. Unfortunately their notion is only known to be sequentially composable and moreover the definition is not compatible with the more general setting of computing reactive functionalities. On the positive side, deception strategies are both split and oblivious of other deceivers, and [9] does provide a construction realizing a large class of (non-reactive) functions f.

Building on the idea of [9], Moran and Naor [32] define a stronger version of incoercibility against adaptive active coercions using split oblivious deception strategies. They go on to provide a construction implementing a voting functionality. Their model of communication and execution is based on that of [5] and, thus, provides sequential (but not concurrent or universal) composability [6]; also, similarly to [9], it is not clear how to extend the model in [32] to reactive functionalities (such as say a commitment scheme).

More recently Unruh and Müller-Quade [35] provided the first universally composable notion of incoercibility. Due to similarity in goals with our work, we provide a comparison with our definition and results. In a nutshell, the definition in [35] specifies the deception strategy D as an extra form of adversary-like machine. The requirement is that for any such deceiver D in the ideal world, there exists a corresponding real-world deceiver D_S (in [35] D_S is called deceiver simulator) such that for any (real-world) adversary $\mathcal{A}$ there exists and (ideal-world) simulator $\mathcal{S}$ that makes the ideal world where D controls the coerced and $\mathcal{S}$ the corrupted parties, indistinguishable from the real world where D_S controls the coerced and $\mathcal{A}$ the corrupted parties, in the presence of any environment $\mathcal{Z}$.[4] Importantly, in [35] it is explicitly assumed that the deceiver has

[4] In fact, the model of [35] builds on the externalized UC (EUC) model of Canetti et al. [7] which is designed to allow for deniable protocols.

access to a public register indicating which parties are corrupted and which are deceiving. As already mentioned, the above modelling choices of [35] have the following side-effects: (1) the real-world deceiving parties are explicitly allowed out-of-band communication (since deception is coordinated by the monolithic D_S) and (2) they know the identities of the corrupted parties, i.e., of the potentially informants. As discussed above these assumptions are not realistic for e-voting. Furthermore, the model of execution in [35] considerably deviates from the GUC model, e.g., it modifies the number of involved ITMs and the corruption mechanism, which can lead to syntactical incompatibilities with GUC protocols and issues with composition with (coercible) GUC protocols.[5]

An alternative approach to universally composable incoercibility was taken in the most recent revision of Canetti's UC paper, and adopted in [10] for the two-party setting. This definition builds on the idea from [9] and is for semi-honest coercions. Furthermore, the coercion mechanism in the multi-party setting is unspecified and no composition theorem is proved.[6]

In terms of protocols, in [10] a two-party protocol in the semi-honest coercion and corruption model is suggested assuming indistinguishability obfuscation [17]. Their approach is based on Yao's garble circuits and is specifically tailored to the two party setting; as they argue, their protocols are not universally composable under active corruption. On the other hand, in [35] a two-party protocol for computing a restricted class of two-party functionalities was suggested; also here it is unclear whether or not this approach can yield a protocol in the multi-party setting or for a wider class of two-party functionalities. Thus ours is the first UC secure incoercible multi-party protocol, which can be, for example, used for receipt-free voting—an inherently multi-party functionality.[7]

Outline of the Remainder of the Paper. In Sect. 2 we present our UC incoercibility definition. Subsequently, in Sect. 3 we describe instances of our definitions corresponding to the three standard coercion types, namely, I/O, receipt-freeness, and active coercion and corresponding composition theorems. Following that, in Sect. 4 we provide our UC receipt-free protocol for computing any given function. Our protocol is simple enough to be considered a good starting point for an alternative approach to existing e-voting protocol. Finally, in Sect. 5 we prove that our receipt-free protocol can withhold even active coercion attacks. Due to space limitation, the proofs have been moved to the full version of this work.

Preliminaries and Notation. Our definition of incoercibility builds on the Universal Composition framework of Canetti [6] from which we inherit the protocol execution model along with the (adaptive) corruption mechanism. We assume the reader has some familiarity with the UC framework [6] but in the following we recall some basic notation and terminology. We denote by `ITM` the set

[5] For example, the corruption mechanism as described in [35] does not specify that (let alone how) the deceiver simulates deception towards the corresponding adversary.

[6] Note that the Definition in [10] also changes the underlying model of computation, which makes it necessary to re-prove composition.

[7] Our protocol uses the CLOS protocol [11] in a black-box manner, and it remains secure even when CLOS is replaced by more efficient protocols, e.g., the IPS protocol [21] in the pre-processing model.

of efficient (e.g. poly-time) ITMs and by $[n]$ the set of integers $\{1, \ldots, n\}$. For simplicity, we use the notations "p_i" and "party i" interchangeably to refer to the party with identity i. For a set $\mathcal{J} \subseteq [n]$ if for each $i \in \mathcal{J}$ the ITM π_i is a protocol machine for party i then we use the shorthand $\pi_{\mathcal{J}}$ to refer to the $|\mathcal{J}|$-tuple $(\pi_{i_1}, \ldots, \pi_{i_{|\mathcal{J}|}})$. In particular we simply write π to denote $\pi_{[n]}$.

A protocol π UC emulates ρ if π can replace ρ in any environment in which it is executed; similarly, a protocol UC realizes a given functionality $\mathcal{F}$ if it UC emulated the the *dummy $\mathcal{F}$-hybrid protocol* ϕ, which simply relays inputs from the environment to $\mathcal{F}$ and vice versa. In [6] protocols come with their hybrids (so the hybrids are not written in the protocol notation); but for sake of clarity in order to make the hybrid-functionality explicit, we at times write it as a superscript of the protocol, e.g., we might denote a $\mathcal{G}$-hybrid protocol π as $\pi^{\mathcal{G}}$.

Finally, we use the following standard UC terminology: we say that a party (or functionality) P *issues a delayed message* x for another party P' (where x can be an input or an output for some functionality) to refer to the process in which P prepares x to be sent to P', but requests for the simulator's approval before actually sending it. Depending on whether or not this approval request includes the actual message, we refer to the delayed output as *public* or *private*, respectively. For details on delayed messages we refer to [6].

2 Our UC Incoercibility Definition

Our security notion aims to capture the intuition that deceiving one's coercer is as easy as the function we are computing allows it to be. Intuitively, this means that for any *(ideal-world) deception* strategy that the coerced party would follow in the ideal world—where the functionality takes care of the computation—there exists a corresponding *(real-world) deception* strategy that he can play in the real world which satisfies the following property:

> The distinguishing advantage of any set of coercers in distinguishing between executions in which parties deceive and ones where they do not deceive is the same in the ideal world (where coerced parties follow their ideal deception strategy `DI`) as it is in the real world (where parties follow their corresponding real-world deception strategy `DR`).

To capture worst-case incoercibility (and get composition) we let the environment play the role of the coercer. This makes the ability of coercers to collude explicit while capturing worst-case and on-line coercion strategies. However, in order to provide a flexible definition, which for example captures the standard notion of receipt-freeness, where coerced parties follow their protocol, we define the effect of a party's coercion as a transformation applied on its protocol, which specifies the control the environment/coercer has on a corrupted party. For example, in the case of receipt-freeness this transformation internally logs the state of the coerced party, and upon reception of a special message from $\mathcal{Z}$ requesting a "receipt" it hands $\mathcal{Z}$ this state. We refer to the next section for a detailed definition of different coercion types.

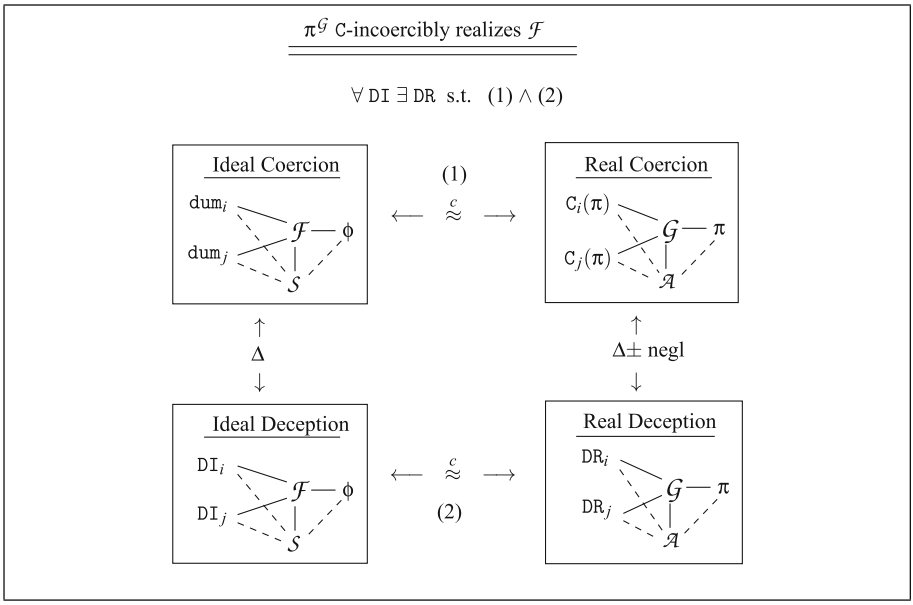

Fig. 1. The incoercibility definition. For clarity we explicitly write the hybrids $\mathcal{F}$ and $\mathcal{G}$. The interfaces on the right of $\mathcal{F}$ (resp. $\mathcal{G}$) correspond to interfaced of honest parties. Parties i and j are coerced according to the coercer C. Dashed lines denote communication tapes to the adversary implicitly modeling a network of insecure channels.

The above idea is demonstrated in Fig. 1, were the following four worlds are illustrated: the ideal world where coerced parties follow their coercer's instructions (top left), the ideal world where coerced parties attempt a deception (bottom left), the real world where coerced parties follow their coercer's instructions (top right), and the real world where the coerced parties attempt a deception (bottom right). As sketched above, incoercibility requires that if the advantage of the best environment (i.e., the one that maximizes its advantage) in distinguishing the top-left world from the bottom-left world is $0 \leq \Delta \leq 1$, then the advantage of the best environment in distinguishing the top-right world from the bottom-right world is also $\Delta' = \Delta$ (plus/minus some negligible quantity).

The above paradigm captures the intuition of incoercibility, but in order to get a more meaningful statement we need the incoercible protocol to also be secure, i.e., implements its specification. This means that when parties do follow their coercion instructions, the protocol should be a secure implementation of the given functionality. In the above terminology, there should be a simulator which makes the top-right world indistinguishable from the top-left world. This has two implications: First, together with the previous requirement, i.e., that $\Delta' = \Delta \pm negl$. it implies that the bottom-right world should also be indistinguishable from the bottom-left world for the same simulator.

Second, to ensure that the top two worlds are indistinguishable for natural coercions, e.g., for receipt-freeness, we need that when the environment sends a

coercion-related message—e.g., a receipt-request—to a coerced party, this message is actually answered wether in the real or in the ideal world. In the real world the coerced protocol will take care of this. Therefore, in the ideal world we assign this task to the simulator: any messages which is not for the functionality is re-routed to the simulator who can then reply with a (simulated) receipt; formally, this is done by applying a "dummy" ideal-coercion strategy which just performs the above rerouting. Importantly, to make sure that the receipt is independent of the actual protocol execution, and in particular independent of the ideal deception strategy, we do not allow the simulation knowledge of the inputs of coerced parties, or of the deception strategy (formally, the latter is guaranteed by ensuring that the ideal deception strategy is applied on messages that are not given to the simulator.) The detailed definition follows.

Coercions and Deceptions. For a given protocol machine π_i we define a *coercion* $\mathtt{C}$ to be a special a mapping from ITMs to ITMs with the same set of communication tapes. In particular the ITM $\mathtt{C}(\pi_i)$ has the same set of communication tapes as π_i and it models the behavior the coercer is attempting to enforce upon party p_i running protocol π. Different types of coercions from the literature can be captured by different types of mappings. In the following section we specify three examples corresponding to the most common coercion types in the literature.

To model the ideal-world behavior (intuitively the "effective" behavior) of a coerced party when obeying its coercer, we use the protocol ITM $\mathtt{dum}$ called the *dummy coercion* (we at times refer to $\mathtt{dum}$ as the *extended dummy* protocol). As sketched above, $\mathtt{dum}$ ensures that the simulator handles all messages that are not intended for the functionality. More concretely, the following describes the behaviour of $\mathtt{dum}$ upon receiving a message from various parties.

From $\mathcal{Z}$: If the message has the form (x, fid) intended for delivery to functionality $\mathcal{F}$, $\mathtt{dum}$ forwards x to $\mathcal{F}$ using a private delayed input (c.f. Page 7). All other messages from $\mathcal{Z}$ are forwarded to the simulator.
From $\mathcal{F}$: Any message from $\mathcal{F}$ is delivered to the simulator.
From $\mathcal{S}$: If the message has the form (x, fid) then $\mathtt{dum}$ forwards x to $\mathcal{F}$. Otherwise it forwards the message to $\mathcal{Z}$.

An ideal-world *deception strategy* corresponds to an attempt of a coerced party to lie to the environment about its interaction with the ideal functionality $\mathcal{F}$. Thus, it can be described as a mapping applied on the messages that the deceiving party exchanges with the functionality and with the environment. To keep our assumptions minimal, we require the real-world (protocol) deception strategy to also have the same structure, i.e., be described it as mappings applied on the messages that the deceiving party p_i running a protocol exchanges with its hybrids and with the environment.[8]

Thus, to capture deception by party p_i running ITM π_i, we define a *deception strategy*, denoted by $\mathtt{D}_i(\pi_i)$, to be an ITM which can be described via a triple

[8] A more liberal, but weaker, definition could allow the real-world deception strategy to be an arbitrary Turing machine with the same hybrids as p_i.

$(\mathtt{D}_i^1, \pi_i, \mathtt{D}_i^2)$ of interconnected ITMs behaving as follows: π_i's messages to/from the adversary are not changed, but we place $\mathtt{D}_i^1$ between π_i and $\mathcal{Z}$ while we place $\mathtt{D}_i^2$ between π_i and its hybrids. For notational simplicity we, at times, omit the argument from $\mathtt{D}_i(\cdot)$ and write $\mathtt{D}_i$ instead of $\mathtt{D}_i(\pi_i)$ when the argument is already clear from the context.

Using these concepts we can now somewhat sharpen the above intuition on our definition. Informally, a protocol π UC *incoercibly* realized a functionality $\mathcal{F}$ with respect to a coercion $\mathtt{C}$ (in short: π $\mathtt{C}$-IUC *realizes* $\mathcal{F}$) if the following two conditions are satisfied: (1) for any set $\mathcal{J} \subseteq [n]$ of coerced parties, when replacing the honest protocol π_i with the wrapped protocol $\widehat{\pi}_i = \mathtt{C}_i(\pi)$ for all $i \in \mathcal{J}$ the resulting network UC realizes the $\mathcal{F}$-dummy protocol ϕ, where the parties $i \in \mathcal{J}$ use $\mathtt{C}_i$ instead of ϕ; and (2) for any player and their ideal deception $\mathtt{DI}_i = \mathtt{D}_i(\mathtt{dum}_i)$ there exists a real deceiving strategy $\mathtt{DR}_i = \mathtt{D}'_i(\mathtt{C}_i(\pi))$ such that no environment can catch coerced parties lying with $\mathtt{DI}_i$ in ρ with probability better than catching them lying with $\mathtt{DR}_i$ in π.

To make the above intuition formal, we need the following notation. Let $\mathcal{J} \subseteq [n]$ denote the set of coerced parties. (To avoid unnecessarily complicated statements, we restrict to static coercion, so the set $\mathcal{J}$ is chosen by $\mathcal{Z}$ at the beginning of the protocol execution.) The execution of protocol π with coercion $\mathtt{C}$ corresponds to executing, *in the UC model of execution*, the protocol which results by replacing for each party $j \in \mathcal{J}$ it's protocol machine π_j with the above described $\mathtt{C}_j(\pi)$. Much like UC, we write $\{\textsc{Exec}_{\pi,\mathtt{C},\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$ to denote the ensemble of the outputs of the environment $\mathcal{Z}$ when executing protocol π with the above modifications, in the presence of adversary $\mathcal{A}$. Consistently with the UC literature, we often write $\textsc{Exec}_{\pi,\mathtt{C},\mathcal{A},\mathcal{Z}}$ instead of $\{\textsc{Exec}_{\pi,\mathtt{C},\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$. We also use the notation UC-$\textsc{Exec}_{\pi,\mathcal{A},\mathcal{Z}}$ to denote the analogous ensemble of outputs for a standard UC execution. For clarity, for the dummy $\mathcal{F}$-hybrid protocol ϕ we might write $\textsc{Exec}_{\mathcal{F},\mathtt{C},\mathcal{S},\mathcal{Z}}$ and UC-$\textsc{Exec}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ instead of $\textsc{Exec}_{\phi,\mathtt{C},\mathcal{A},\mathcal{Z}}$ and UC-$\textsc{Exec}_{\phi,\mathcal{A},\mathcal{Z}}$, respectively.

Definition 1 (UC Incoercibility). *Let π be an n-party protocol and for an n-party functionality $\mathcal{F}$ let ϕ 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 $\mathtt{DI}_i$ there exists an real deception strategy $\mathtt{DR}_i$ with the following property. For every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any set $\mathtt{DI}_\mathcal{J} = \{\mathtt{DI}_i : i \in \mathcal{J}\}$ and every environments $\mathcal{Z}$:*

$$\textsc{Exec}_{\phi,\mathtt{dum}_\mathcal{J},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \textsc{Exec}_{\pi,\mathtt{C}_\mathcal{J},\mathcal{A},\mathcal{Z}} \tag{1}$$

$$\textsc{Exec}_{\phi,\mathtt{DI}_\mathcal{J},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \textsc{Exec}_{\pi,\mathtt{DR}_\mathcal{J},\mathcal{A},\mathcal{Z}} \tag{2}$$

where $\mathtt{dum}$ *denotes the dummy coercer described above.*

We observe that when no party is coerced, i.e., $\mathcal{J} = \emptyset$, then the definition coincides with UC security which shows that incoercibility with respect to any type of coercion also implies standard UC security.

3 Types of Coercion

Using our definition we can capture the types of coercion previously considered (mainly in the e-voting) literature. These types are specified in this section, where we also prove the composability of the corresponding definitions.

I/O Coercion. As a sanity check we look at a particularly weak form of coercion called *input/output (I/O) coercion*. Intuitively, this corresponds to a setting where a party is being coerced to use a particular input and must return the output of the protocol to the coercer as evidence of it's actions. We capture this formally by defining the I/O coercion $\mathtt{C}^{\mathrm{io}}$ to be identical to the dummy coercion; that is for any protocol machine π_i $\mathtt{C}^{\mathrm{io}}(\pi_i) = \mathtt{dum}(\pi_i) = \pi_i$. In particular it faithfully uses the input to π_i supplied by $\mathcal{Z}$ and follows the code of π_i during the entire execution, and eventually returns the output back to $\mathcal{Z}$.

Not surprisingly, we already have I/O-incoercible protocols for a wide variety of functionalities since standard UC realization is equivalent to I/O-incoercible realization.

Theorem 1. *In the static corruption model protocol π UC realizes functionality $\mathcal{F}$ with static corruptions if and only if π $\mathtt{C}^{io}$-IUC realizes $\mathcal{F}$.*

An immediate consequence of Theorem 1 and the UC composition theorem in [6] is that I/O-incoercibility is a composable notion.

Semi-honest Coercion (Receipt-Freeness). The type of incoercibility that has been mostly considered in the literature is the so-called receipt-freeness. The idea there is that the coercer expects to be provided with additional evidence of that a specific input was used. In the most severe case such a proof could, for example, be the entire view of a coerced party in the protocol execution.

In the following, we define the semi-honest coercion $\mathtt{C}^{\mathrm{sh}}$, which captures receipt freeness: at a high-level, for a given protocol machine π_i the ITM $\mathtt{C}^{\mathrm{sh}}(\pi_i)$ behaves identically to π_i with the only difference that upon being asked by $\mathcal{Z}$, ITM $\mathtt{C}^{\mathrm{sh}}(\pi_i)$ outputs all messages it has received from the adversary and it hybrids as well as it's random coins (i.e., the contents of his random tape). Note that, as $\mathcal{Z}$ already knows the messages it previously sent to $\mathtt{C}^{\mathrm{sh}}(\pi_i)$, it can now reconstruct the entire view of p_i in the protocol.

Intuitively, the output of $\mathtt{C}^{\mathrm{sh}}(\pi_i)$ can be used as a receipt that p_i is running π_i on the inputs chosen by $\mathcal{Z}$ as follows. On the one hand, any message p_i claims to have received over the insecure channels can be confirmed to $\mathcal{Z}$ by the informant. On the other hand, for any prefix of receipt causing π_i to send a message over the insecure channel, $\mathcal{Z}$ can check with it's informant if indeed exactly that message was sent by p_i at that point.

Theorem 2. *Let $\mathtt{C}$ be a semi-honest coercer, i.e., $\mathtt{C} = \mathtt{C}^{sh}$. If protocol π $\mathtt{C}$-IUC-realizes functionality $\mathcal{F}$, and protocol σ $\mathtt{C}$-IUC-realizes functionality $\mathcal{H}$ in the $\mathcal{F}$-hybrid world, then the composed protocol σ^{π} $\mathtt{C}$-IUC-realizes functionality $\mathcal{H}$.*

Active Coercion. We next turn to defining active coercion. Here, instead of simply requiring a receipt, the coercer takes complete control over the actions of coerced parties. We capture this by introducing the fully-invasive—also referred to as *active* coercion $\mathtt{C}^{\mathrm{A}}$ which allows the environment full control over the coerced party's interfaces. Formally, for any (set of) functionalities $\mathcal{G}$ and any $\mathcal{G}$-hybrid protocol π $\mathtt{C}(\pi_i) = \bar{\phi}_i$ where $\bar{\phi}_i$ is the $\mathcal{G}$-hybrid dummy coercer's protocol, i.e., $\bar{\phi}_i = \mathtt{dum}_i^{\mathcal{G}}$. A universal composition theorem for incoercibility against active coercion can be proved along the lines of Theorem 2.

4 Receipt-Free-Incoercible Multi-party Computation

In this section we describe a protocol for IUC realizing any (well-formed [11]) n-party functionalities $\mathcal{F}$ in the presence of semi-honest (i.e. receipt-free) coercions. Our construction makes black-box use of the UC secure protocol by Canetti et al. [11] but it can be instantiated also with other (faster) UC secure protocols. In fact, our construction can be seen as a compiler of UC secure protocols to IUC secure protocols in the honestly generated hardware-token setting. Thus, by replacing the call to the protocol in [11] with a call to a faster UC secure protocol we obtain a reasonably efficient candidate for universally composable receipt-free voting.

Our protocol (compiler) assumes access to *honestly-generated* tamper resistant hardware tokens that perform encryption under a key which is secret shared among the parties.

Intuitively, the receipt-freeness of the protocol $\Pi_{\mathcal{F}}$ can be argued as follows: because the token does not reveal the encryption keys to anyone, the CPA security of the encryption scheme ensures that the adversary cannot distinguish encryptions of some x_i from encryption of an $x_i' \neq x_i$. Thus the real deceiver $\mathtt{DR}_i$ for a coerced p_i can simply change the input it provides the token according to the ideal deceiver $\mathtt{DI}_i$ and report back to $\mathcal{Z}$ (as part of the receipt) the actual reply to the token. Since we assume $t + t' < n$ there is at least one share of the decryption key unknown to $\mathcal{Z}$ and so it can not immediately detect that the ciphertext given in the receipt doesn't encrypt x_i. At this point $\mathtt{DR}_i$ can follow the rest of the protocol honestly and can report the remainder of it's view honestly in the receipt. A formal theorem and proof follow.

The Construction. For simplicity we restrict ourselves to non-reactive functionalities, also known as *secure function evaluation (SFE)*. (The general case can be reduced to this case by using a suitable form of secret sharing for maintaining the secret state of the reactive functionality.) Moreover, we describe our protocols as *synchronous protocols*, i.e., round-based protocols where messages sent in some round are delivered by the beginning of the next round; such protocols can be executed in UC as demonstrated in [24,25]. We point out that the protocols in [24] assume a global synchronizing clock; however, as noted in [24,25], when we do not require guaranteed termination, e.g., in fully asynchronous environments, the clock can be emulated by the parties exchanging dummy synchronization messages. We further assume that the parties have access to a broadcast channel.

Without loss of generality, we assume that the functionality $\mathcal{F}$ being computed has a global output obtained by evaluating the function f on the vector of inputs. The case of local (a.k.a. private) and/or randomized functionalities can be dealt with by using standard techniques (c.f. [29].) Furthermore, as is usual with UC functionalities, we assume that $\mathcal{F}$ delivers its outputs in a delayed manner—whenever an output is ready for some party the simulator $\mathcal{S}$ is notified and $\mathcal{F}$ waits for $\mathcal{S}$'s permission to deliver the output.[9] Finally, to ensure properly synchronized simulation, we need to allow $\mathcal{S}$ to know when honest parties hand their input to the functionality. Thus we assume that the functionality $\mathcal{F}$ informs the simulator upon reception of any input x_i from an honest party p_i. We point out that as we allow a dishonest majority, we are restricted to security with abort, i.e., upon receiving a special message (`abort`) from the simulator, the functionality $\mathcal{F}$ sets all outputs of honest parties to a special symbol $\perp$.

Finally, our protocols makes use of an *authenticated additive n-out-of-n secret sharing*. Informally, this is an additive secret sharing where each share is authenticated by a digital signature for which every party knows the verification key, buy no party knows the signing key. We refer to the full version for a formal specification of our scheme.

In the remainder of this section we present our protocol and prove its security. We start by describing the hardware token that our protocol needs. The token functionality $\mathcal{T}_{\texttt{ThEnc}}$ captures a threshold authenticated encryption token and is described in Fig. 2. The token is parameterized by an IND-CPA secure symmetric key encryption scheme $(\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ and an existentially unforgeable signature scheme $(\texttt{Gen}', \texttt{Sign}, \texttt{Ver})$. Initially the token generates signature key pair $(\texttt{sk}, \texttt{vk})$. Then upon request from any party i (or the adversary when p_i is corrupt) it generates a random encryption key k_i for p_i and uses $\texttt{vk}$ to compute an n-out-of-n authenticated sharing k_i. Each party $j \in [n]$ requests it's share $\langle k_i \rangle_j$. Subsequently, whenever p_i requests an encryption of some message m from the token, $\mathcal{T}_{\texttt{ThEnc}}$ computes a *fresh* encryption of m under key k_i and hands the result to p_i.

Given hybrid access to $\mathcal{T}_{\texttt{ThEnc}}$ $(\mathcal{P})$, our protocol $\pi_{\mathcal{F}}$ for $\mathtt{C}^{\text{sh}}$-incoercibly (UC) securely realizing any given functionality $\mathcal{F}$ proceeds in three sequential phases:

1. In the setup phase, for each player (at their behest) an encryption key is generated and shared with an n-out-of-n authenticated secret sharing. Formally, for each $i \in [n]$ a message $(\texttt{keygen}, i)$ is sent by p_i to the token.[10] Shares are then delivered to parties when they send a `keyShare` to $\mathcal{T}_{\texttt{ThEnc}}$.
2. In the second phase, each p_i asks the token to encrypt its inputs x_i under key k_i, i.e., inputs $(\texttt{encrypt}, x_i, i)$ to the token $\mathcal{T}_{\texttt{ThEnc}}$ $(\mathcal{P})$.
3. Finally, in a third phase, the parties invoke a UC secure SFE protocol, e.g., the one from [11] denoted by Π_{CLOS}, to implement the functionality $\widehat{\mathcal{F}}$. Roughly

[9] Because we restrict to public-output functions, we can wlog assume that the output is issued in a *public* delayed manner (c.f., Sect. 1).

[10] Presumably in a real world setting this phase will be executed on behalf of the players by the authority in charge of running the election. Then the tokens with an initialized state can be distributed to the players.

speaking, $\widehat{\mathcal{F}}$ receives from each player as input a ciphertext and one key-share for each of the decryption keys $k_1, \ldots, k_n$, reconstructs the decryption keys from the shares, and uses them to decrypt the ciphertexts to obtain plaintexts $\{x_i\}_{i\in[n]}$. If either reconstruction (i.e. signature verification) or decryption fails then $\mathcal{F}$ outputs $\perp$. Otherwise it computes and outputs a fresh n-out-of-n authenticated sharing of the value $f(x_1, \ldots, x_n)$. We refer to the full version for a formal description of protocol $\Pi_{\mathcal{F}}$ and functionality $\widehat{\mathcal{F}}$.

Functionality $\mathcal{T}_{\mathtt{ThEnc}}$ ($\mathcal{P}$)

The token functionality is parameterized by a symmetric-key CPA encryption scheme $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$, an existentially unforgeable signature scheme $(\mathtt{Gen}', \mathtt{Sign}, \mathtt{Ver})$, and a security parameter λ.

- Upon receiving message $(\mathtt{keygen}, i)$ from party $i \in \mathcal{P}$ (or the adversary) if p_i is honest and a message $(\mathtt{keygen}, i)$ has already been received from p_i then ignore it; otherwise execute the following steps:
  1. Sample $k_i \leftarrow \mathtt{Gen}(1^\lambda)$.
  2. Sample an n-out-of-n authenticated sharing $\{\langle k_i \rangle_j\}_{j\in[n]}$ of k_i.
- Upon receiving message $(\mathtt{keyShare}, i, j)$ from party $p_i \in \mathcal{P}$, if share $\langle k_i \rangle_j$ has not already been sampled then send $\perp$ to p_i. Otherwise send it to party i.
- Upon receiving message $(\mathtt{encrypt}, m, i)$ from party p_i for some $m \in \{0,1\}^*$, if a key k_i has not already been sampled then send $\perp$ to p_i. Otherwise compute a fresh encryption $e_i = \mathtt{Enc}_{k_i}(m)$ and send it to p_i.

Fig. 2. Threshold encryption token functionality

The security of protocol $\Pi_{\mathcal{F}}$ is argued as follows: As long as there is at least one honest party, the adversary will not get information about any of the encryption keys k_i. This follows from the security of the encryption scheme $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ used by the token and the privacy of the protocol Π_{CLOS}. Thus the simulator can simulate the entire protocol execution by simply using encryptions of random messages to simulate the tokens responses and storing local (simulated) copied of coerced parties. The unforgeability of the signatures used by the token to authenticate the shares will guarantee that the adversary cannot alter the input of honest or coerced parties by giving faulty inputs to the execution of Π_{CLOS}.

Theorem 3. *Let $\mathcal{F}$ be a n-party well-formed functionality as above. Further let $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme secure against chosen plaintext attacks (IND-CPA) and $(\mathtt{Gen}', \mathtt{Sign}, \mathtt{Ver})$ be an existentially unforgeable signature scheme. Then the protocol $\Pi_{\mathcal{F}}$ $\mathtt{C}^{sh}$-incoercibly (UC) securely realizes the functionality $\mathcal{F}$ in the static corruption model in the presence of any t corrupted and t' coerced parties where $t + t' < n$.*

5 Active-Incoercible Multi-party Computation

In this section we consider the strongest form of coercion, namely active coercions. Recall that these essentially turn a coerced party into a dummy party with all interaction driven by $\mathcal{Z}$. It turns out that the protocol from the previous section achieving semi-honest-incoercibility can also be shown to achieve full active-incoercibility. There are two key differences between the two security notions which must be addressed in the proof.

1. In a simulation for a semi-honest coerced party p_i, the simulator $\mathcal{S}$ must maintain a simulated internal state of p_i so that it can always respond to a receipt request from $\mathcal{Z}$. However no such requirement is placed on $\mathcal{S}$ for active coercions making the job of $\mathcal{S}$ easier in this respect.
2. On the other hand, say $\mathcal{Z}$ instructs a coerced (non-deceiving) party p_i to give input x to $\mathcal{F}$. In both the semi-honest and active case in the ideal worlds p_i will forward x to $\mathcal{F}$. Moreover in the semi-honest case p_i would use x as input to the honest protocol. However case of an active coercion $\mathcal{Z}$ is essentially running the protocol on behalf of p_i as it wishes. Thus their is no guarantee that x will be the effective input of p_i in such a protocol execution. So $\mathcal{S}$ must now extract the effective input of p_i during the protocol execution and force p_i to submit that as input to $\mathcal{F}$ in place of x. (Indeed, this is where $\mathcal{S}$ uses the property that parties have *delayed* input to $\mathcal{F}$.) Otherwise the two world would, in general, be distinguishable.

Theorem 4 (Active-Incoercibility). *Let $\mathcal{F}$ be a n-party well-formed functionality as above. Further let* $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme secure against chosen plaintext attacks (IND-CPA) and* $(\mathtt{Gen}', \mathtt{Sign}, \mathtt{Ver})$ *be an existentially unforgeable signature scheme. Then the protocol $\Pi_{\mathcal{F}}$ $\mathtt{C}^A$-incoercibly (UC) securely realizes the functionality $\mathcal{F}$ in the static corruption model in the presence of any t corrupted and t' coerced parties where $t + t' < n$.*

Acknowledgements. Joël Alwen was supported by the ERC starting grant (259668-PSPC). Rafail Ostrovsky was supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, Lockheed-Martin Corporation Research Award, and the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014 -11 -1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Vassilis Zikas was supported in part by the Swiss National Science Foundation (SNF) via the Ambizione grant PZ00P-2142549.

References

1. Backes, M., Hritcu, C., Maffei, M.: Automated verification of remote electronic voting protocols in the applied pi-calculus. In: CSF, pp. 195–209. IEEE Computer Society (2008)

2. Backes, M., Pfitzmann, B., Waidner, M.: The reactive simulatability (RSIM) framework for asynchronous systems. Inf. Comput. **205**(12), 1685–1720 (2007)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, pp. 1–10. ACM Press, May 1988
4. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: 26th ACM STOC, pp. 544–553. ACM Press, May 1994
5. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology **13**(1), 143–202 (2000)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
8. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
9. Canetti, R., Gennaro, R.: Incoercible multiparty computation (extended abstract). In: FOCS, pp. 504–513. IEEE Computer Society (1996)
10. Canetti, R., Goldwasser, S., Poburinnaya, O.: Adaptively secure two-party computation from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 557–585. Springer, Heidelberg (2015)
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press, May 2002
12. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: 20th ACM STOC, pp. 11–19. ACM Press, May 1988
13. Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.): Towards Trustworthy Elections. LNCS, vol. 6000. Springer, Heidelberg (2010)
14. Commision, E.E.: Internet voting in estonia, October 2013
15. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: CSFW, pp. 28–42. IEEE Computer Society (2006)
16. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols: a taster. In: Chaum et al. [13], pp. 289–309
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press, October 2013
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC, pp. 218–229. ACM Press, May 1987
19. Heather, J., Schneider, S.: A formal framework for modelling coercion resistance and receipt freeness. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 217–231. Springer, Heidelberg (2012)
20. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, p. 539. Springer, Heidelberg (2000)
21. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)

22. Jonker, H.L., de Vink, E.P.: Formalising receipt-freeness. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 476–488. Springer, Heidelberg (2006)
23. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Chaum et al. [13], pp. 37–63
24. Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 477–498. Springer, Heidelberg (2013)
25. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. In: Kleinberg, J.M. (ed.) 38th ACM STOC, pp. 109–118. ACM Press, May 2006
26. Küsters, R., Truderung, T.: An epistemic approach to coercion-resistance for electronic voting protocols. In: 2009 IEEE Symposium on Security and Privacy, pp. 251–266. IEEE Computer Society Press, May 2009
27. Küsters, R., Truderung, T., Vogt, A.: Verifiability, privacy, and coercion-resistance: New insights from a case study. In: IEEE Symposium on Security and Privacy, pp. 538–553. IEEE Computer Society (2011)
28. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. J. Comput. Secur. (special issue of selected CSF 2010 papers) **20**(6/2012), 709–764 (2012)
29. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptology **22**(2), 161–188 (2009)
30. Maurer, U., Renner, R.: Abstract cryptography. In: Chazelle, B. (ed.) ICS 2011, pp. 1–21. Tsinghua University Press, January 2011
31. Michels, M., Horster, P.: Some remarks on a receipt-free and universally verifiable mix-typevoting scheme. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 125–132. Springer, Heidelberg (1996)
32. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
33. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Lomas, M., Crispo, B., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
34. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
35. Unruh, D., Müller-Quade, J.: Universally composable incoercibility. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 411–428. Springer, Heidelberg (2010)
36. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press, November 1982