Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case

Jesper Buus Nielsen*

BRICS** Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Arhus C, Denmark

Abstract. We show that there exists a natural protocol problem which has a simple solution in the random-oracle (RO) model and which has no solution in the complexity-theoretic (CT) model, namely the problem of constructing a non-interactive communication protocol secure against adaptive adversaries a.k.a. non-interactive non-committing encryption. This separation between the models is due to the so-called programability of the random oracle. We show this by providing a formulation of the RO model in which the oracle is not programmable, and showing that in this model, there does not exist non-interactive non-committing encryption.

1 Introduction

Before describing our separation result and the non-programmable random-oracle (NPRO) model, we introduce non-committing encryption (NCE) and the non-interactive NCE (NINCE) problem.

Non-committing Encryption. One way of constructing a secure protocol for the cryptographic model is to take a protocol which is secure in the information theoretical model, where secure channels are assumed, and then compile this protocol for the cryptographic model by adding encryption to the channels. A motivation for this approach has been, that only statically secure general multi-party computation (MPC) protocols have been constructed for the cryptographic model directly, whereas adaptively secure protocols for the information theoretical model were published already in [BGW88,CCD88]. The goal is therefore to replace the secure channels of the information theoretical model by open channels using an adaptively secure communication protocol a.k.a. NCE.

Before we can define NCE more formally we have to sketch our MPC model. We use the model of asynchronous MPC from [Can01]. The security of a protocol

* `buus@brics.dk`.
** Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

is defined by requiring that the real-life execution of the protocol can be simulated efficiently given only access to an ideal-world abstraction of the protocol problem that the protocol is to solve. The real-life execution is controlled by an adversary $\mathcal{A}$ (a probabilistic polynomial time (PPT) interactive Turing machine (ITM)) which can see all communication between the parties and schedules message delivery. By PPT we mean PPT in the security parameter k, which is given to all entities in the system. The adversary can furthermore adaptively corrupt parties to learn their current state (or entire execution history if we do not model erasures) and start controlling the corrupted party. The execution takes place in context of an environment $\mathcal{Z}$ (also a PPT ITM) which provides inputs to and receives outputs from the parties. The environment and adversary can communicate during the execution. We denote the output of the environment after such an execution by $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$, where π is the protocol. This execution is compared to an ideal-world execution where the parties have access to an ideal functionality with the desired input-output behavior of the protocol. Message delivery is controlled by an ideal-world adversary $\mathcal{S}$ which can again corrupt parties and learn their internal state (which is just the inputs from the environment), and the protocol is executed in context of an environment $\mathcal{Z}$ with the same role as in the real-life model. We denote the output of the environment after such an execution by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, where $\mathcal{F}$ is a PPT ITM specifying the desired input-output behavior of the protocol problem to be solved. These executions are then compared by saying that for each real-life adversary $\mathcal{A}$ there should exist an ideal-world adversary $\mathcal{S}$ such that for all environments $\mathcal{Z}$ the executions $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable, i.e. the environment cannot tell whether its observing a real-life execution or the simulator $\mathcal{S}$ running in the ideal-world. The role of $\mathcal{S}$ is similar to the role of the simulator in the definition of zero-knowledge. The role of the environment is that of an distinguisher between the real-life execution $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ and the simulation $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. An important part of the model is that the environment receives the identity of all corrupted parties. This guarantees that the simulator does not accomplish its goal by corrupting other parties than the real-life adversary.

For the NCE problem the ideal functionality $\mathcal{F}_{\mathtt{nce}}$ works as follows: On input (mid, j, m) from P_i deliver (mid, i, m) to P_j, and reveal $(mid, i, j, |m|)$ to the adversary. Here mid is a message identifier, m is the message, and $|m|$ is the length of m. For the specific task of secure communication the above definition of security basically says that whatever a real-life adversary can obtain from attacking the protocol an ideal adversary $\mathcal{S}$ could obtain (simulate) given just the length of the messages sent.

If we let each party P_i have a private key for a semantically secure public-key encryption scheme, where the public key pk_i is known by all other parties, and if we encrypt all communication to P_i under pk_i (including in the messages the identity of the sender to protect against copying), then we will have a statically secure implementation. However, no encryption scheme exists for which this protocol is *adaptively* secure. This follows from a general result that no non-interactive communication protocol is adaptively secure; Throughout the paper we will let non-interactive communication protocol denote a communication protocol with the property that after a pre-processing phase, which might involve interaction (e.g. the receiver sending a public key to the sender), the sender can send an unbounded number of bits to the receiver without there being any communication from the receiver to the sender.

We show that no non-interactive communication protocol can be adaptively secure in the asynchronous model. Assume for this sake that we have an adaptively secure communication protocol for the asynchronous model. Consider two parties P_R and P_S acting as receiver resp. sender. Consider the environment $\mathcal{Z}$ which activates P_S with an arbitrary message m of length $l_m(k)$, where $l_m(k)$ is some polynomial. Consider the adversary $\mathcal{A}$ that corrupts no party but just waits for P_R and P_S to finish the preprocessing phase and for P_S to send a ciphertext c to P_R. The adversary outputs c to the environment and then corrupts P_R before c arrives, and the adversary outputs to the environment the value sk of the internal state of P_R. By the security of the encryption scheme we have that if the environment runs the code of P_R from internal state sk and with input c, then c will decrypt to m, except possibly with negligible probability. Now, by the definition of security there should exist a simulator $\mathcal{S}$ such that the simulator executed in the ideal-world execution with the same environment $\mathcal{Z}$ produces an output indistinguishable from that of the adversary $\mathcal{A}$. But in the ideal-world abstraction of secure communication given by $\mathcal{F}_{\mathsf{nce}}$, the simulator does not see m during the execution as long as both parties are uncorrupted, and the simulator must therefore generate c given just $|m|$. Then on the corruption of P_R, the simulator sees m and computes sk to give to the environment. Since the definition of security requires that the environment cannot tell the difference between the real-life execution and the simulation it follows that running P_R from sk on input c will result in output m except with negligible probability, in particular with probability more than $\frac{1}{2}$. Since no internal state can make c decrypt to two different values, both with more than probability $\frac{1}{2}$, there exists an injective map from messages m to internal states $sk_{c,m}$ which make c decrypt to m. Intuitively this means that the length of sk must be at least l_m. If the protocol can send an unbounded number of bits, this holds for any polynomial l_m and thus the length of sk must be superpolynomial contradicting that P_R is a PPT ITM.

The NCE problem was first solved by Beaver and Haber in [BH92]. In their protocol P_R sends to P_S the public key pk. Then P_S generates a uniformly random message p and sends $c = E_{pk}(p)$ to P_R. Then P_R computes $p = D_{sk}(c)$ *and erases everything except* p. When m later becomes known to P_R he computes $c' = p \oplus m$, where $\oplus$ denotes bitwise xor, and sends c' to P_R who can then compute $m = p \oplus c'$. Since at no point P_R knows both sk and the encryption c' of m, the attacker cannot obtain both, which preempts the problem that for fixed c' there should be an injective map from sk to messages m. Since sk is deleted a new key-pair must be generated each time P_S has sent a total of $|p|$ bits. If further more synchronization between the parties are assumed, the protocol can be made non-interactive: Set aside a prefix of p to use as a seed s for a pseudo-random generator, and each time p is used up, expand s to obtain a new p as long as the original p and delete s. This method is then iterated each time p is used up. In this way no more than $|p|$ bits are communicated from P_S to P_R before P_R deletes its internal state and creates a new one.

The protocol from [BH92] depends essentially on the use of erasure. However, in many settings trusting the parties to be able to reliably erase parts of their state might be unrealistic, due to e.g. physical limitations on erasure and weak operating systems. The first solution to the NCE problem in the non-erasure model is presented in [CFGN96] by Canetti et al. The scheme is however inefficient: It can encrypt 1 bit using a public key of $\Theta(k^2)$ bits. Later Beaver[Bea97] and Damgård and Nielsen[DN00] proposed more efficient schemes communicating 1 plaintext bit using $\Theta(k)$ bits of communication. These protocols are all three-round protocols.

The Random-Oracle Model. The idea behind the random-oracle (RO) model is that by modeling primitives as DES, MD5 or SHA using the strong assumption that they (properly used/modified) behave like ROs, one can build efficient and secure protocols based on these primitives. The model has been used to argue the security of a number of constructions. Examples are the OAEP encryption mode for RSA[BR95,Sho01] and the Fiat-Shamir heuristic[FS86].

We define the RO model to be the real-life execution model described above where additionally the parties and the adversary has access to a uniformly random function $\{0,1\}^* \to \{0,1\}^k$. This can be modeled within the framework in [Can01] using a hybrid model. A hybrid model is the real-life model extended with an ideal functionality $\mathcal{F}$ (also called a trusted party) to which all parties have a secure channel. The calls to $\mathcal{F}$ works as in the ideal-world. We use $\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ to denote an execution of protocol π in the hybrid model with trusted party $\mathcal{F}$, and say that π realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model if for each hybrid adversary $\mathcal{A}$ there exist an ideal-world adversary $\mathcal{S}$ such that for all environments $\mathcal{Z}$ the executions $\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable. We let the RO model by the hybrid model with the trusted party $\mathcal{F}_{\mathtt{ro}}$ working as follows: On input $x \in \{0,1\}^*$ from any of the parties or the adversary it outputs a uniformly random value $r \in \{0,1\}^k$ to the calling party; If queried on the same x twice, the same r is returned. Thus $\mathcal{F}_{\mathtt{ro}}$ defines a uniformly random function $H : \{0,1\}^* \to \{0,1\}^k$.

Possibility of NINCE in the RO model. We prove that if trapdoor permutations exists, then NINCE exists in the RO model. Our protocol is reminiscent of a construction of chosen ciphertext secure encryption in [BR93]. In the pre-processing phase the receiver P_R sends a description f of a trapdoor permutation to the sender P_S. Each message from P_S to P_R is transmitted as $(f(x), H(x) \oplus m)$, where x is a uniformly random element in the domain of f and H is the uniformly random function defined by $\mathcal{F}_{\mathtt{ro}}$. To prove the scheme secure we construct a simulator $\mathcal{S}$. The simulator works by running internally a copy of the protocol and a copy of $\mathcal{A}$. It tries to make the internal protocol consistent with the values of m input to the ideal-world execution (knowing only $|m|$) and lets $\mathcal{A}$ attack the simulated execution and lets $\mathcal{A}$ do the interaction with the environment $\mathcal{Z}$. The simulator $\mathcal{S}$ simulates in such a way that $\mathcal{A}$ thinks that it observes a real-life execution, and such that in particular its interaction with the environment is distributed computationally indistinguishable from that observed by $\mathcal{Z}$ in the real-life execution, which in turn makes the output of $\mathcal{Z}$ computationally indistinguishable in the two worlds. The simulator $\mathcal{S}$ proceeds as follows: Distribute the public keys as in the real-life. Note that $\mathcal{A}$ and the parties of the protocol might request to evaluate the RO H on a value x, as they expect to run in the RO model. To simulate the RO, $\mathcal{S}$ returns a uniformly random element r; If queried on the same x twice, it returns the same r. To simulate the sending of m the simulator $\mathcal{S}$ generates random x and sends $(f(x), H(x) \oplus 0^l)$, where l is the length of m and 0^l is the all-zero string of length l. If the simulated oracle H was not defined on x the simulator sets it to be a uniformly random element r as above. Assume that after simulating the sending of an arbitrary number of messages $\mathcal{A}$ corrupts P_R. The simulator then corrupts P_R in the ideal evaluation, and for each $(f(x), H(x) \oplus 0^l)$ sent in the simulation it receives the real value m which should have been sent and must come up with an internal state of P_R consistent with m. Assume that the simulator defined $H(x) = r$, i.e. that $(f(x), r)$ was the message sent. The simulator then simulates by simply claiming that $H(x) = m \oplus r$. This is a perfect simulation as we get that $(f(x), r) = (f(x), (r \oplus m) \oplus m) = (f(x), H(x) \oplus m)$. However, there are two ways this simulation can fail. First of all, if the same x was used twice the simulator might be in the situation that it needs to define $H(x)$ to both $r \oplus m_1$ and $r \oplus m_2$ for $m_1 \neq m_2$. However this happens with negligible probability as the x's are chosen uniformly at random by the simulator. Second, it might be that $\mathcal{A}$ queried H on x and therefore knows that $H(x)$ was defined to r, which *commits* the simulator to this choice and makes the simulation fail. However, if the $\mathcal{A}$ queried H on x it intuitively had to invert the trapdoor function f on a uniformly random element: $\mathcal{A}$ returned x given only $f(x)$. This would contradict the hardness of inverting f on random elements, and thus except with negligible probability the simulation goes through.

Impossibility of NINCE in the CT and NPRO Model. The simulator sketched above uses essentially that it is possible to program the RO, by defining the value of $H(x)$ to be some value appropriately chosen by the simulator: It sets $H(x)$ to $r \oplus m$ after H "should" have been defined on x. We can prove that the use of the programability of the RO is necessary for the simulator. We start by formalizing the NPRO model.

The NPRO model is the real-life model, where all ITMs are extended to be ITMs with oracle access to a random oracle. An ITM M with oracle access is an ITM which in addition to the usual tapes and states has an oracle query tape, an oracle input tape, and a classification of some of its states as oracle query states. We write $M^{(\cdot)}$ to denote an ITM with oracle access. We write $M^{\mathcal{O}}$ to denote running M with oracle $\mathcal{O}$. If M enters an oracle query state, then the contents of the oracle query tape is given as input to $\mathcal{O}$, and the output of $\mathcal{O}$

is written on the oracle input tape of M. Now let $\mathcal{O}$ denote an ITM defining a uniformly random function $H : \{0,1\}^* \to \{0,1\}^k$. The NPRO model defines the two distribution ensembles $\mathrm{REAL}_{\pi^{\mathcal{O}},\mathcal{A}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ and $\mathrm{IDEAL}_{\mathcal{F}^{\mathcal{O}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ and as above these are compared by requiring that for each real-life adversary $\mathcal{A}^{(\cdot)}$ there exists an ideal-world adversary $\mathcal{S}^{(\cdot)}$ such that for all environments $\mathcal{Z}^{(\cdot)}$ the executions $\mathrm{REAL}_{\pi^{\mathcal{O}},\mathcal{A}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ and $\mathrm{IDEAL}_{\mathcal{F}^{\mathcal{O}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ are computationally indistinguishable.

The main difference between the RO model and the NPRO model is that in the NPRO model also the environment has access to the RO $\mathcal{O}$. Intuitively this allows the environment to verify whether the values that it is shown is consistent with the RO that it has access to, which basically makes it impossible for $\mathcal{S}$ to program the random oracle according to its desires.

The impossibility of NINCE in the NPRO model follows the proof for the CT model sketched in the introduction to NCE. Because $\mathcal{Z}$ has access to the same RO as $P_R^{(\cdot)}$ it can run $P_R^{\mathcal{O}}$ from internal state sk with input c and it follows that there exists an injective mapping from the possible messages to the fixed set of possible internal states of $P_R^{(\cdot)}$ after the pre-processing phase. This argument fails in the programmable RO model as the environment does not have access to $\mathcal{O}$ and thus cannot run $P_R^{\mathcal{O}}$.

To obtain our separation it would be enough to prove NINCE impossible in the asynchronous model without erasure. However, to strengthen the separation result we show that NINCE is impossible in a number of weaker models too. We show the result for the asynchronous model with erasure, the synchronous model without erasure, and for the synchronous model with erasure we show that no NCE protocol can communicate an unbounded number of bits *per round*; By the result from [BH92] mentioned above we cannot hope to prove a stronger result than this for the synchronous model with erasure.

Previous Separation Results. Other examples of constructions secure in the RO model and not secure in the CT model were known prior to our work. Most prominently, in [CGH98] Canetti, Goldreich and Halevi construct an encryption scheme which is secure in the RO model, but is not secure in the CT model no matter the instantiation of the RO. The scheme is constructed as to try to "detect" whether it is in the RO model or not, and then reveal the secret key if it is not in the RO model. A strength of the result from [CGH98] is that it is the semantic security of the encryption scheme that is violated in the CT model, whereas it in our example it is the less standard non-committing property that is violated. Their result thus establishes that even standard security properties do not carry over from the RO model to the CT model. Another strength of the result from [CGH98] is that their encryption scheme can be proven secure in the NPRO model, as they do not use the programability of the RO. This means that their result separates the CT model and the NPRO model.[1]

Another well-known separation result is that the Fiat-Shamir[FS86] methodology can be proven secure in the RO model, and that not all non-interactive zero-knowledge proofs obtained by the methodology using a *fixed* function for

[1] Using CS proofs for the NPRO model.

implementing RO cannot be black-box zero-knowledge in the CT model unless $BPP \subset NP$.[GK90] This is however not a separation of the strength of the models: When the RO is implemented by a random function h drawn from some function family, say, by a trusted party, and handed to both the prover and the verifier, then f is a de facto common random string and the existence of one-round zero-knowledge proofs is no longer ruled out[BFM88]. It does in particular not follow that there does not exist in some preprocessing model some kind of non-interactive instantiation of the RO which makes the methodology secure.

Discussion and Future Work. We have shown that the programability of the RO in proofs in the RO model is a feature of the model which is so strong that there exist natural protocol problems which are trivially solvable in the RO model, but have no solution in a model without the programability.

We point out that the NPRO model is formulated in this paper primarily to pin-point a property of the RO model which allows for our separation result. It is not meant as a suggestion for 'the' formulation of a weaker RO model. Though it could be interesting to have a weaker formulation of the RO model, as to increase the trust that security in the model would imply a certain level of 'heuristic security' in the real world, our formulation has two shortcomings for this purpose: First of all, our formulation of the NPRO model only addresses security defined by simulation. Second, it is possibly to define even weaker versions of the RO model than the NPRO model and it is not clear which would be 'the appropriate' weak formulation.

As for the first shortcoming, the definition can to some extend be applied to different types of definitions of security as semantic security of encryption schemes and non-forgeability of signature schemes by giving an equivalent definition of the security notion in the MPC framework. As an example we describe how to define NPRO semantic security of public-key encryption: Let $\mathcal{E}$ be a public-key cryptosystem, and let $\pi_{\mathcal{E}}$ be the following protocol for two parties P_S and P_R: First P_R generates a random key pair (pk, sk) and sends pk to P_S. Each time P_S receives input m from the environment, it computes $c = E_{pk}(m)$ and sends c to P_R who computes and outputs to the environment the value $m' = D_{sk}(c)$. It can be proven that a public-key cryptosystem $\mathcal{E}$ is semantic secure in the CT model (resp. in the RO model of [BR93]) iff $\pi_{\mathcal{E}}$ is statically secure in the CT model (resp. in the RO model). Generalizing this, we can say that a public-key cryptosystem $\mathcal{E}$ is semantic secure in the NPRO model iff $\pi_{\mathcal{E}}$ is statically secure in the NPRO model.

As for existence of even weaker RO models, note that another strong property of the RO model which was used by our simulator was that the simulator learns on which points the simulated adversary evaluates the RO. This was what allowed us to make the reduction to the one-wayness of the trapdoor permutation f, as the simulator could obtain x from $f(x)$ if the adversary could evaluate H on x given $f(x)$. We call this property evaluation point knowledge (EPK). One interpretation of what is modeled by EPK is that it isn't possible to learn the value of $H(x)$ without knowing all of x. The fact that the simulator learns all points on which the adversary evaluates the oracle can then be viewed as a knowledge extraction of the adversary's EPK. The NPRO model still has the EPK property. We could formulate a RO model without EPK by requiring that $\mathcal{S}$ must simulate given only oracle access to $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{O}$. However, we find that this is far from a satisfactory formulation of the model, as it has the serious restriction that as it only applies to black-box proofs.

We find giving a simple and general formulating of the NPRO model and a RO model without EPK an interesting open problem.

The Rest of the Paper. The purpose of the rest of the paper is to give a formalization of the NPRO model and the separation between the RO and the NPRO model.

2 Trapdoor Permutations

Definition 1 (Collection of trapdoor permutations). *We call a tuple $(\mathcal{K}, F, \mathcal{G}, \mathcal{X})$ a collection of trapdoor permutations with security parameter k, if $\mathcal{K}$ is an infinite index set, $F = \{f_{pk} : D_{pk} \to D_{pk}\}_{pk \in \mathcal{K}}$ is a set of permutations, the key/trapdoor-generator $\mathcal{G}$ and the domain-generator $\mathcal{X}$ are PPT (in k) algorithms, and the following hold:*

Easy to generate and compute *$\mathcal{G}$ generates pairs of keys and trapdoors, $(pk, sk) \leftarrow \mathcal{G}(k)$, where $pk \in \mathcal{K} \cap \{0,1\}^{p(k)}$ for some fixed polynomial $p(k)$. Furthermore, there is a polynomial time algorithm which on input pk and $x \in D_{pk}$ computes $f_{pk}(x)$.*

Easy to sample domain *$\mathcal{X}$ samples elements in the domains of the permutations, we write $x \leftarrow \mathcal{X}(pk)$, where x is uniformly random in D_{pk}.*

Hard to invert *For $(pk, sk) \leftarrow \mathcal{G}(k)$, $x \leftarrow \mathcal{X}(pk)$, and for any PPT algorithm A the probability that $A(pk, f_{pk}(x)) = x$ is negligible in k.*

But easy with trapdoor *There is a polynomial time algorithm which on input pk, sk, $f_{pk}(x)$ computes x, for all $(pk, sk) \in \mathcal{G}(k)$ and $x \in D_{pk}$.*

Let A be any PPT ITM and consider the following game, which we will call the trapdoor game. The game is between A and the tuple $(\mathcal{K}, F, \mathcal{G}, \mathcal{X})$. The algorithm A can ask for a number of public key generations and element generations, and the goal of A is to invert a permutation for which it does not know the trapdoor information, on an element it did not generate itself.

- On a key generation request, A is given pk for a uniformly random key $(pk, sk) \leftarrow \mathcal{G}(k, r_{\mathcal{G}})$ (here $r_{\mathcal{G}}$ denotes the random bits used by $\mathcal{G}$).
- On a give up request on pk, where pk was generated in a key generation request, A is given $r_{\mathcal{G}}$.
- On an element generation request for pk, A receives $y = f_{pk}(x)$, where x was generated as $x \leftarrow \mathcal{X}(pk, r_{\mathcal{X}})$.
- On a give up request on y, where y was generated in an element generation request, A is given $r_{\mathcal{X}}$.

– The ITM A wins the game, if it manages to return an element x such that $y = f_{pk}(x)$, where pk is a key from a key generation request on which it has not given up and where y is from an element generation request on which it has not given up.

It is straightforward to prove the following lemma.

Lemma 1. *The tuple* $(\mathcal{K}, F, \mathcal{G}, \mathcal{X})$ *is a collection of trapdoor permutations iff for all PPT algorithms A, the probability that A wins over* $(\mathcal{K}, F, \mathcal{G}, \mathcal{X})$ *in the trapdoor permutation game is negligible.*

3 The Multiparty Computation Model

We will use the framework for universally composable asynchronous MPC from [Can01].

The General Framework. A protocol $\pi = (P_1, \ldots, P_n)$ consists of n PPT ITMs. The most general computation model considered in [Can01] is the hybrid model with ideal functionality $\mathcal{F}$. The execution in the hybrid model involves the parties, the ideal functionality $\mathcal{F}$, the adversary $\mathcal{A}$, and the environment $\mathcal{Z}$. The ideal functionality, the adversary and the environment are PPT ITMs. All parties are connected by point-to-point channels. These channels are modeled as insecure authenticated asynchronous channels by letting the adversary $\mathcal{A}$ see all messages sent and schedule message delivery (without being able to introduce messages). Besides controlling message delivery the adversary can corrupt parties. When a party is corrupted the adversary learns the current internal state or the entire execution history of the party (depending on whether we allow erasures or not) and from the point of corruption the adversary sends messages on behalf of the corrupted party. Besides the communication channels all parties are connected to $\mathcal{F}$ with secure channels ($\mathcal{A}$ does not see the messages, but still schedules the delivery). When a party P_i or the ideal functionality $\mathcal{F}$ receives a message it runs it code and sends messages accordingly. The ideal functionality can also receive messages from $\mathcal{A}$ and send messages to $\mathcal{A}$. Finally, the role of the environment is to deliver input to the parties and receive outputs from the parties. The environment can also input to the adversary and the adversary can send messages to the environment. The environment is the driver of the execution. At the beginning of the protocol it receives an auxiliary input $z \in \{0,1\}^*$, and it then activates the adversary and the parties of the protocol by giving them input. At some point the environment stops activating parties and halts by outputting some bit b. Let $\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}(k, z)$ be a random variable describing the output of $\mathcal{Z}$.

We define the security of a protocol by *comparing* the input-output behavior (as seen be the environment) of its execution to an *ideally secure protocol* with the *desired input-output* behavior. We specify the desired input-output of the protocol by giving an ideal-functionality $\mathcal{F}$ defining the desired input-output behavior of the protocol. The ideally secure protocol implementing this desired input-output behavior is then defined to be $\mathrm{HYB}^{\mathcal{F}}_{\tilde{\pi},\mathcal{A},\mathcal{Z}}(k,z)$, where $\tilde{\pi}$ is the dummy protocol where the parties just send their input from the environment to $\mathcal{F}$ and send the response from $\mathcal{F}$ to the environment. Since the parties are connected to $\mathcal{F}$ via secure channels and $\mathcal{F}$ cannot be corrupted this protocol is trivially secure. We call $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z) = \mathrm{HYB}^{\mathcal{F}}_{\tilde{\pi},\mathcal{S},\mathcal{Z}}(k,z)$ the ideal-world execution. We then say that a protocol π securely realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model if for all adversaries $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for all environments $\mathcal{Z}$ and all $c \in \boldsymbol{N}$ there exists $k_c \in \boldsymbol{N}$ such that for all $z \in \{0,1\}^*$ it holds that that $|\Pr[\mathrm{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}}(k,z) = 1] - \Pr[\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z) = 1]| < k^{-c}$.

Non-committing Encryption. We specify the desired input-output behavior of secure communication by the functionality $\mathcal{F}_{\mathtt{nce}}$, which on input $(\mathtt{send}, mid, j, m)$ from P_i delivers $(\mathtt{receive}, mid, i, m)$ to P_j and delivers $(\mathtt{receive}, mid, i, j, |m|)$ to $\mathcal{A}$. The value mid is a message identifier. We say that π is a an NCE protocol for some model if π securely realizes $\mathcal{F}_{\mathtt{nce}}$ in that model.

Consider any communication protocol for two parties, sender P_S and receiver P_R, of the following form: First the parties execute a pre-processing phase. The protocol is executed independently of the messages to be send later, and in particular the length of the internal state of P_R after the pre-processing, which we denote by sk, is independent of the messages to be send. Then each time a message m becomes known to P_S he computes an encryption c of m and sends c to P_R who outputs a value m'. We allow access to ideal functionalities during the pre-processing phase. This means that in principle the keys could be distributed entirely by a trusted party. We only require that P_R receives no messages from P_S or ideal functionalities during decryption! We call such a protocol a non-interactive communication protocol. Since no messages are send from P_R to P_S between the encryptions sent to P_R from P_S and the computation is asynchronous we can assume that the protocol can handle arbitrary long messages, possibly by blockwise encryption using unique message identifiers.

The Random-Oracle Model. The random-oracle model is the hybrid model with access to an ideal functionality $\mathcal{O}$ specified as follows: On input $x \in \{0,1\}^*$ from any party (including the adversary) the functionality outputs to the calling party a uniformly random element $y \in \{0,1\}^k$ independent of all other evaluations (except that if queried on the same x twice the same value y will be returned). We say that a protocol π securely realizes $\mathcal{G}$ in the random-oracle model if π securely realizes $\mathcal{G}$ in the $\mathcal{O}$-hybrid model.

The Non-Programmable Random-Oracle Model. We say that a protocol $\pi^{(\cdot)} = (P_1^{(\cdot)}, \ldots, P_n^{(\cdot)})$ securely realizes $\mathcal{G}$ in the non-programmable random-oracle model if for all adversaries $\mathcal{A}^{(\cdot)}$ there exists an adversary $\mathcal{S}^{(\cdot)}$ such that for all environments $\mathcal{Z}^{(\cdot)}$ we have that $\mathrm{IDEAL}_{\mathcal{G}^{\mathcal{O}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ and $\mathrm{REAL}_{\pi^{\mathcal{O}},\mathcal{A}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}$ are computationally indistinguishable, where $\mathcal{O}$ is the RO functionality.

4 Possibility of NINCE in the RO Model

Let F be a family of trapdoor permutations, where one can verify $y \in D_{pk}$ given just pk, and consider the following protocol $\pi_{F,\mathtt{nce}}$: On initialization of the protocol each party P_i generates $(pk_i, sk_i) \leftarrow \mathcal{G}(k)$ and sends pk_i to all other parties. After the key distribution phase the protocol proceeds as follows:

Send On input $(\mathtt{send}, mid, j, m)$ party P_i generates a uniformly random element $x \leftarrow \mathcal{X}(pk_j)$, computes $(mid, f_{pk_j}(x), H(mid\|i\|j\|x) \oplus m)$, and sends this value to P_j.[2]

Receive If P_j receives (mid, y, R) from P_i, where $y \in D_{pk_j}$, then P_j computes $x = f^{-1}_{sk_j}(y)$ and $m = R \oplus H(mid\|i\|j\|x)$ and outputs $(\mathtt{receive}, mid, i, m)$.

Theorem 1. *If F is a family of trapdoor permutations, then $\pi_{F,\mathtt{nce}}$ is a NINCE protocol for the RO model.*

Proof. Let $\mathcal{A}$ be any PPT adversary. We construct an ideal process adversary $\mathcal{S}$, which running in the ideal process will simulate an execution of $\pi_{F,\mathtt{nce}}$ to $\mathcal{A}$ and let $\mathcal{A}$ do the communication with any $\mathcal{Z}$ to convince $\mathcal{Z}$ that it is viewing a real-life execution. Since the protocol runs in the RO model, $\mathcal{S}$ will also have to simulate a RO H. It does this by defining $H(h)$ to be some uniformly random value $r \in \{0,1\}^k$, when $H(h)$ is needed. The simulator $\mathcal{S}$ will simulate the key-distribution phase by generating random keys as in the protocol. In fact, to make the proof of security easier we will assume that $\mathcal{S}$, besides running in the ideal process, participates in a trapdoor game. The public keys pk_i for the parties will then be obtained from the trapdoor game using n key generation requests. The trapdoors will therefore not be known to $\mathcal{S}$.

To be able to simulate without the trapdoors we represent H in a particular way using two dictionaries `raw` and `img`. At the beginning of the simulation both dictionaries are empty, and H is undefined on all values. We record a new definition $H(h) := r$ as follows.

- If h can be parsed as $mid\|i\|j\|x$, where i and j are indicies of parties and $x \in D_{pk_j}$, then the entry $(mid\|i\|j\|y, r)$, where $y = f_{pk_j}(x)$, is added to `img`.
- If h cannot be parsed as described above, then (h, r) is added to `raw`.

We say that $H(h)$ is defined and $H(h) = r$ iff $h = mid\|i\|j\|x$ (for $x \in D_{pk_j}$) and $(mid\|i\|j\|f_{pk_j}(x), r) \in$ `img`, or h cannot be parsed as specified and $(h, r) \in$ `raw`. Because f_{pk_j} is a permutation, this representation is consistent: $H(h) = r$ will become defined iff recorded. Equally important, this representation allows to define and evaluate H on $h = mid\|i\|j\|x$ given just (mid, i, j, y), where $y = f_{pk_j}(x)$. We call these manipulations oblivious.

Remember that $\mathcal{S}$ has access to the ideal-world execution. We name the parties in the ideal world $\tilde{P}_1, \ldots, \tilde{P}_n$ — remember that these dummy parties just pass messages between the environment and the ideal functionality $\mathcal{F}_{\mathtt{nce}}$. The parties of the simulated execution run by $\mathcal{S}$ we call $P_1, \ldots, P_n$. The simulation proceeds as follows:

[2] We let $\|$ denote an injective and easily parsable encoding $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.

RO Evaluation If $\mathcal{A}$ asks for an evaluation of the RO on some string h, then if $H(h)$ is defined, return $H(h)$, otherwise generate uniformly random $r \in \{0,1\}^k$, set $H(h) := r$, and return r.

Send On input $(\mathtt{send}, mid, i, j, |m|)$ from the NCE functionality we know that $\tilde{P}_i$ has input $(\mathtt{send}, mid, j, m)$ for some $m \in \{0,1\}^{|m|}$ to the NCE functionality, which has then sent $(\mathtt{receive}, mid, i, m)$ to $\tilde{P}_j$.

- If $\tilde{P}_j$ is corrupted, then $\mathcal{S}$ will deliver the message to $\tilde{P}_j$ to learn m and will then simulate by following the protocol using m as the message.
- If $\tilde{P}_j$ is honest, then $\mathcal{S}$ simulates the protocol to $\mathcal{A}$ by sending the message (mid, y, R), where y is obtained as a uniformly random element in the image of f_{pk_j} from the trapdoor game and $R \in \{0,1\}^k$ is chosen uniformly at random. If at a later point P_i or P_j is corrupted then:
  - If P_i was corrupted, then $\mathcal{S}$ corrupts $\tilde{P}_i$ in the ideal process and learns m. The simulator then gives up on y and learns x, r such that $x = \mathcal{X}(pk_j, r)$ and $y = f_{pk_j}(x)$. The simulator then gives up on pk_i and learns sk_i, r such that $(pk_i, sk_i) = \mathcal{G}(k, r)$. Then the simulator gives this internal view of P_i to $\mathcal{A}$ and records $H(mid\|i\|j\|x) := R \oplus m$.
  - If P_j was corrupted, then $\mathcal{S}$ corrupts $\tilde{P}_j$ in the ideal process and learns m. The simulator then gives up on pk_j and learns sk_j, r such that $(pk_j, sk_j) = \mathcal{G}(k, r)$. Then the simulator gives this internal view of P_j to $\mathcal{A}$ and records $H(mid\|i\|j\|x) := R \oplus m$.

  If $H(mid\|i\|j\|x)$ was already defined in either of the above cases (to a value different from $R \oplus m$), then the simulator gives up the simulation.

Receive On the message (mid, y, R) from P_i to P_j the simulator $\mathcal{S}$ needs to make the ideal functionality output $(\mathtt{receive}, mid, i, m)$ to $\tilde{P}_j$, where $m = R \oplus H(mid\|i\|j\|f^{-1}_{sk_j}(y))$.

- If P_i is honest, then (mid, y, R) was sent by $\mathcal{S}$ itself and in that case the message $(\mathtt{receive}, mid, i, m)$ has already been sent to $\tilde{P}_j$ in the ideal process. The simulator then delivers this message to $\tilde{P}_j$.
- If P_i is corrupted, then decrypt as follows: If $H(mid\|i\|j\|f^{-1}_{sk_j}(y))$ is not defined then obliviously define it to a uniformly random value. Obliviously look up $H(mid\|i\|j\|f^{-1}_{sk_j}(y))$, and let $m = R \oplus H(mid\|i\|j\|f^{-1}_{sk_j}(y))$. Then input $(\mathtt{send}, mid, j, m)$ to $\tilde{P}_i$ in the ideal process and make $\mathcal{F}_{\mathtt{nce}}$ deliver the message $(\mathtt{receive}, mid, i, m)$ to $\tilde{P}_j$.

If the simulation is not given up, then it is distributed exactly as a real-life execution. It is therefore enough to prove that the probability that the simulation is given up is negligible. Assume for the sake of contradiction that the simulation is given up with significant (i.e. not negligible) probability. This means that with significant probability the simulator obtained $y = f_{pk_j}(x)$ from the trapdoor game and send (mid, y, R) from honest P_i to honest P_j, and the dictionary was defined on the value $mid\|i\|j\|x$ before $\mathcal{S}$ needed to define it on that value. Since P_i is guaranteed to be honest up to the point in the simulation where $\mathcal{S}$ needs to define the dictionary on the value $mid\|i\|j\|x$, we can neglect the probability that the simulator has defined H on $mid\|i\|j\|x$ twice, as it would involve choosing the same value y in the image of f_{pk_j} twice under the uniform distribution, which happens with negligible probability. Therefore the other definition of H on $mid\|i\|j\|x$ was made by the adversary in a RO Evaluation. The first definition of H on $mid\|i\|j\|x$ was therefore not oblivious, and thus $x = f_{sk_j}^{-1}(y)$ is known. Since both P_i and P_j are honest up to the point where the simulation is given up, the simulator has not given up on y or pk_j. This allows the simulator to win the trapdoor game with significant probability, a contradiction to Lemma 1. □

5 Impossibility of NINCE in the NPRO Model

We start by proving a lemma. Let $S^{(\cdot)}$ be a probabilistic ITM with oracle access, let $D^{(\cdot)}$ be a probabilistic TM with oracle access, and let $l_m, l_{sk} : \boldsymbol{N} \to \boldsymbol{N}$. We say that $S^{(\cdot)}$ is a NPRO non-committing cryptosystem simulator with private key-length l_{sk}, message length l_m, and decryption algorithm $D^{(\cdot)}$ if the following holds: On input $k \in \boldsymbol{N}$ and access to a RO $\mathcal{O}$ the ITM $S^{(\cdot)}$ outputs a string $c \in \{0,1\}^*$. Then on input $m \in \{0,1\}^{l_m(k)}$ the ITM $S^{(\cdot)}$ outputs a string $sk \in \{0,1\}^{\leq l_{sk}(k)}$, where $\{0,1\}^{\leq l_{sk}(k)}$ is the set of strings of length at most $l_{sk}(k)$. For all $m \in \{0,1\}^{l_m(k)}$, let $\Pr_{S,m}[c, sk, r]$ denote the joint probability distribution on the values (c, sk, r) when c, sk are generated by $S^{\mathcal{O}}$ on inputs (k, m) and r is a uniformly random string. Let $\Pr_S[c, r]$ be the distribution of the values (c, r), which are independent of m. We require that there exists k_0 such that for all $k > k_0$ and all $m \in \{0,1\}^{l_m(k)}$,

$$\Pr_{S,m}[D^{\mathcal{O}}(sk, c; r) = m] \geq \frac{3}{4} \ . \tag{1}$$

Lemma 2. *If $S^{(\cdot)}$ is a NPRO non-committing cryptosystem simulator with private key-length l_{sk}, message length l_m, and decryption algorithm $D^{(\cdot)}$, then for all $k > k_0$: $l_{sk}(k) + 2 \geq l_m(k)$.*

Proof. Assume for the sake of contradiction that there exists $k > k_0$ such that $l_{sk}(k) + 2 < l_m(k)$. For all $c \in \{0,1\}^*$ and all $m \in \{0,1\}^{l_m(k)}$ define

$$\begin{aligned} SK(c, m) &= \{sk \in \{0,1\}^{\leq l_{sk}(k)} | \Pr_S[D^{\mathcal{O}}(sk, c; r) = m] > \frac{1}{2}\} \\ X_m &= \sum_{c \in \{0,1\}^*} \Pr_S[c] |SK(c, m)| \\ X &= \sum_{m \in \{0,1\}^{l_m(k)}} X_m \ . \end{aligned}$$

If $sk \in SK(c, m_1) \cap SK(c, m_2)$ and $m_1 \neq m_2$, then $\Pr_S[D^{\mathcal{O}}(sk, c; r) \in \{m_1, m_2\}] > 1$. So, for fixed c each $sk \in \{0,1\}^{\leq l_{sk}(k)}$ belongs to only one of the sets $SK(c, m)$. Therefore

$$\begin{aligned} X &= \sum_{c \in \{0,1\}^*} \Pr_S[c] \sum_{m \in \{0,1\}^{l_m(k)}} |SK(c, m)| \\ &\leq \sum_{c \in \{0,1\}^*} \Pr_S[c](2^{l_{sk}(k)+1} - 1) = 2^{l_{sk}(k)+1} - 1 \ . \end{aligned} \tag{2}$$

If $X_m \geq \frac{1}{4}$ for all $m \in \{0,1\}^{l_m(k)}$, then by (2) we have that $2^{l_{sk}(k)+1} - 1 \geq X \geq 2^{l_m(k)}\frac{1}{4}$ contradicting $l_{sk}(k) + 2 < l_m(k)$. So, there exists $m \in \{0,1\}^{l_m(k)}$ s.t. $X_m < \frac{1}{4}$. In particular, by the Markov inequality $\Pr_S[|SK(c,m)| \geq 1] < \frac{1}{4}$ and

$$\begin{aligned}
&\Pr_{S,m}[D^{\mathcal{O}}(sk,c;r) = m] \\
=&\Pr_{S,m}[|SK(c,m)| \geq 1] \cdot \Pr_{S,m}[D^{\mathcal{O}}(sk,c;r) = m||SK(c,m)| \geq 1] + \\
&\Pr_{S,m}[|SK(c,m)| = 0] \cdot \Pr_{S,m}[D^{\mathcal{O}}(sk,c;r) = m||SK(c,m)| = 0] \\
<&\frac{1}{4} \cdot 1 + 1 \cdot \frac{1}{2} = \frac{3}{4}
\end{aligned}$$

contradicting (1). □

Theorem 2. *There exists no NINCE protocol for the NPRO model with erasure.*

Proof. Consider the real-life execution of a NINCE protocol with two parties P_S and P_R (we drop the $(\cdot)$ superfix as all ITMs in the proof are ITMs with oracle access). Let $sk(k)$ denote the random variable describing the internal state of P_R after the pre-processing phase and before receiving the first encryption. Since this state is independent of the inputs to be sent, there exists a polynomial $l_{sk}(k)$ s.t. the expected value of $|sk(k)|$ is bounded by $\frac{l_{sk}(k)}{6}$ for large enough k, and by the Markov inequality $\Pr[|sk(k)| \geq l(k)] < \frac{1}{6}$ for large enough k.

Consider the following environment $\mathcal{Z}$: It activates P_S on a message $m \in \{0,1\}^{l_m(k)}$, where $l_m(k) = l_{sk}(k) + 3$ and m is a prefix of z. Let E_1 denote the event that after activating P_S with input m the adversary outputs a value $c \in \{0,1\}^*$. If E_1 does not occur, the environment terminates with output 0. If E_1 occurs, the environment activates the adversary with input "corrupt the receiver". Let E_2 be the event that $\mathcal{Z}$ as response to this activation observes that P_R is corrupted and that the adversary outputs a value $sk \in \{0,1\}^{\leq l_{sk}(k)}$. If E_2 does not occur, $\mathcal{Z}$ outputs 0. If E_2 does occur, $\mathcal{Z}$ generates uniformly random bits r and computes m' by running the code of P_R from internal state sk with input c and random bits r and using the RO $\mathcal{O}$; Since P_R does not access any ideal functionalities during decryption and $\mathcal{Z}$ has access to the same oracle as P_R, the environment can actually carry out this computation. If $m' = m$ then $\mathcal{Z}$ outputs 1, otherwise $\mathcal{Z}$ outputs 0.

Consider the following real-life adversary $\mathcal{A}$: It activates P_S and P_R until P_S sends c. Then $\mathcal{A}$ outputs c, where c is the value sent from P_S to P_R but not delivered. Then on the next activation, it corrupts P_R and outputs to the environment the internal state of P_R.

Let E be the event that E_1 and E_2 occurs. Note that in the real-life execution $\mathrm{REAL}_{\pi^{\mathcal{O}},\mathcal{A}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z)$ the event E occurs with probability at least $\frac{5}{6}$ for large enough k. Furthermore, by the security (which implies correctness) of the protocol, the probability that $m' \neq m$ for uniformly random r is bounded by a negligible function $\delta(k)$. Therefore $\Pr[\mathrm{REAL}_{\pi^{\mathcal{O}},\mathcal{A}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z) = 1] \geq \frac{5}{6} - \delta(k) > \frac{4}{5}$ for large enough k. Since the protocol is secure and $\frac{3}{4} < \frac{4}{5}$ there exists $\mathcal{S}$ and k_0

such that for all $k > k_0$ we have that $\Pr[\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z) = 1] \geq \frac{3}{4}$. We use this to construct a NPRO non-committing cryptosystem simulator S as follows: On input k and access to oracle $\mathcal{O}$ run $\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z)$ on a message m of length $l_m(k)$. Note that as long as no party is corrupted the execution does not require that we know the value of m. If during the execution the event E does not occur, S uses arbitrary values for c and sk. If E occurs, S proceeds as follows: Run $\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z)$ until $\mathcal{S}$ outputs c and then output c. Then on input $m \in \{0,1\}^{l_m(k)}$ run $\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z)$ until $\mathcal{S}$ corrupts P_R, give $\mathcal{S}$ the value m, and run $\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z)$ until $\mathcal{S}$ outputs sk. Then output sk. Let D be the TM which on input c, sk and access to oracle $\mathcal{O}$ runs P_R from internal state sk and input c using uniformly random bits and the RO $\mathcal{O}$. By $\Pr[\text{IDEAL}_{\mathcal{F}_{\text{nce}},\mathcal{S}^{\mathcal{O}},\mathcal{Z}^{\mathcal{O}}}(k,z) = 1] \geq \frac{3}{4}$ we have that S is a non-committing cryptosystem simulator with private key-length l_{sk}, message length l_m, and decryption algorithm D, contradicting Lemma 2. □

The proof of Theorem 2 was done in the model from [Can01]. Since Theorem 2 states a negative result, the result would be stronger if we could prove it in a weaker model. In [CFGN96] the NCE problem was formulated in the MPC model of [Can00], which is a considerably weaker model as it models synchronous computation and only guarantees security preservation under non-concurrent composition of protocols. The negative result however still holds in this model. We can prove that no NINCE protocol exists for the synchronous model without erasure, and we can prove that no NINCE protocol which can communicate an unbounded number of bits in one round (i.e. without synchronizing the sender and the receiver) exists for the synchronous model with erasure. The proof of the first claim follows the proof of Theorem 2, except that c is communicated between the execution and the so-called post-execution phase and P_R is corrupted in the post-execution phase. We sketch the proof of the second claim.

The main difference between the models in [CFGN96] and [Can00] is that the model in [Can00] does not have an explicit mechanism for the adversary to output values to the environment during the execution. Since it is essential to the proof with erasure that P_R is corrupted before c arrives and that it is essential that c is output before P_R is corrupted, this becomes an issue. However in [Can00] some information can indeed flow from the adversary to the environment as the environment learns the identity of the parties corrupted by $\mathcal{A}$. The NCE problem is cast as a multiparty problem, see [CFGN96], where a polynomial number of parties participate. So the adversary can use its corruption pattern to communicate c; To guarantee that the value output has a fixed polynomial length, a Markov inequality can be used as in the proof of Theorem 2.

6 Acknowledgments

I would like to thank the following persons for valuable discussions of the paper and for suggestions on how to improve the presentation: Ran Canetti, Ronald Cramer, Ivan Damgård, Yehuda Lindell, Moti Yung, and a number of anonymous referees.

References

[ACM88] *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, 2–4 May 1988.

[Bea97] D. Beaver. Plug and play encryption. In *Crypto '97*, pages 75–89, Berlin, 1997. Springer. LNCS Vol. 1294.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In [ACM88], pages 103–112.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In [ACM88], pages 1–10.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *EuroCrypt '92*, pages 307–323, Berlin, 1992. Springer. LNCS Vol. 658.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computing and Communications Security*, pages 62–73. ACM, 1993.

[BR95] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EuroCrypt '94*, pages 92–111, Berlin, 1995. Springer. LNCS Vol. 950.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, winter 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science*. IEEE, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In [ACM88], pages 11–19.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 639–648, Philadelphia, Pennsylvania, 22–24 May 1996.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 209–218, Dallas, TX, USA, 24–26 May 1998.

[DN00] Ivan Damgård and Jesper B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Crypto 2000*, pages 432–450, Berlin, 2000. Springer. LNCS Vol. 1880.

[FS86] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Crypto '86*, pages 186–194, Berlin, 1986. Springer. LNCS Vol. 263.

[GK90] O. Goldreich and H. Krawczyk. On the composition of zero knowledge proof systems. In *Proceedings of ICALP 90*, Berlin, 1990. Springer. LNCS Vol. 443.

[Sho01] Victor Shoup. OAEP reconsidered. In *Crypto 2001*, pages 239–259, Berlin, 2001. Springer. LNCS Vol. 2139.