Basic Research in Computer Science

Non-Committing Encryption is Too Easy in the Random Oracle Model

Jesper Buus Nielsen

BRICS Report Series **RS-01-47**

ISSN 0909-0878 **December 2001**

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/01/47/`

Non-Committing Encryption is Too Easy in the Random Oracle Model

Jesper Buus Nielsen

December 6, 2001

Abstract

The non-committing encryption problem arises in the setting of adaptively secure cryptographic protocols, as the task of implementing secure channels. We prove that in the random oracle model, where the parties have oracle access to a uniformly random function, non-committing encryption can be implemented efficiently using any trapdoor permutation.

We also prove that no matter how the oracle is instantiated in practice the resulting scheme will never be non-committing, and we give a short discussion of the random oracle model in light of this.

1 Introduction

1.1 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.

The motivation for such an approach has been, that only statically secure general multiparty computation 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, by using an adaptively secure protocol for encrypting all communication on the open channels.

It is obviously not enough to encrypt the messages using standard public key encryption. The reason is that this allows an adversary to observe a ciphertext from P_1 to P_2, say, and later send it as a ciphertext from some corrupt party P_i to P_2, thereby making P_i send the same message as P_2, which would be impossible for P_i if the communication had been over secure channels. If, however we use a chosen ciphertext attack (CCA) secure encryption scheme and include in the messages the identity of the sender to protect against copying, then we will have a statically secure implementation[Can01].

However, no CCA secure encryption scheme is known for which this protocol is *adaptively* secure in the non-erasure model, where the parties are not trusted to be able to erase parts of their state reliably. The problem is that after sending an encryption $E(m, r)$ of a message m the sending party is generally committed to m.

Assume that the protocol, where P_2 sends a uniformly random public key to P_1 and P_1 returns $c = E_{pk}(i\|m)$, were adaptively secure in the multiparty computation model of e.g. [Can01]. Consider the adversary that corrupts no party and the environment, which activates the sender with an arbitrary message m and waits for the protocol to generate output. We let the adversary output the observed public key pk and encryption c. Then the environment corrupts the sender and the adversary gives to the environment the observed random bits r such that $c = E_{pk}(m, r)$.

Now, by the definition of security there should exist a simulator such that the simulator executed in the ideal process with the same environment produces an output indistinguishable from that of the adversary. But in the ideal process for secure communication the parties basically share a secure channel, and thus the simulator does not see anything during the execution, and it must generate pk and c independently of m. Then on the corruption of the sender, the simulator sees m and computes r_m to give to the environment. Since the environment cannot tell the difference from the real execution it follows that pk is computationally indistinguishable from a real public key, that r_m is computationally indistinguishable from a uniformly random string, and that $c = E_{pk}(m, r_m)$. This means that the encryption scheme has the property that one can generate a 'fake' public key pk and a 'fake' encryption c such that c can later be claimed to contain any message m.

The problem with the above protocol is that the r_m, which the environment should be shown as the internal state of the sender, cannot be computed for typical encryption schemes. If however, we considered the

cryptographic model with erasure, we could specify as part of the protocol that r should be erased. Then the adversary and environment will not see r in the protocol execution and therefore the simulator should not compute r_m in the simulation either. This removes the problem and, as already observed, we can obtain an efficient adaptive security preserving reduction to the information theoretical model. This was first shown by Beaver and Haber in [BH92]. There, a semantically secure encryption scheme is used and all keys, random bits, and other values are deleted after the message is send.

However, in many settings trusting the parties to be able to erase parts of their state might be unrealistic, due to e.g. physical limitations on erasure and weak operating systems. The first adaptive security preserving reduction to the information theoretical model without using erasure is by Canetti et al.[CFGN96]. They define a non-committing encryption scheme to be a protocol which securely implements the encryption functionality in the cryptographic model in the presence of adversaries that might corrupt the sender and the receive adaptively. They construct a solution based on what they call common domain trapdoor systems and show how to build non-committing encryption based on the RSA and the DH assumption. Their scheme has expansion factor $\Omega(k^2)$, i.e. the communication complexity of their scheme is $\Omega(k^2)$ bits per plaintext bit to be communicated.

The protocols based on the RSA and DH assumptions are the most efficient in terms of rounds. They are both two-round protocols, which is optimal when no secret information is shared in advance. In the first round the receiver send a public key and in the second round the sender sends an encryption of the message under the public key.

The protocol based on general common-domain trapdoor systems uses an interactive protocol to set up the public keys and uses three more rounds.

Later Beaver[Bea91] proposed a simpler and more efficient scheme based on the DDH assumption. His scheme has expansion factor $\Theta(k)$ and is a three-round protocol. In [DN00] Damgård and Nielsen generalized the ideas of Beaver to construct non-committing encryption with a similar complexity based on the RSA assumption and showed how to construct non-committing encryption from any collection of trapdoor permutations (with the technical condition, that the domain of the permutation can be sampled in an invertible manner).

1.1.1 Our Result

In this paper we show that in the random oracle model non-committing encryption can be implemented using any collection of trapdoor permutations. The protocol is non-interactive (one-round) and has a constant expansion factor. Our protocol is reminiscent of a construction of chosen ciphertext secure encryption in [BR93], only we extend the basic semantically secure system in a different manner. A message will be transmitted as $(mid, f(x), H(mid\|i\|j\|x\|m))$, where f is trapdoor permutation (where f^{-1} is only known by the receiver), H is a pseudo-random function, mid is a message id, i and j are the unique identities of the sender respectively the receiver, and x is a uniformly random element in the domain of f. If the function H is modeled as random oracle, i.e. a uniformly random function to which the parties and the adversary only have oracle access, then this scheme can be shown to be a non-committing encryption scheme. This scheme is as efficient as the protocol in [BH92] for the erasure model. This proves that in the random oracle model, secure channels can be implemented as efficiently as in the erasure model.

1.2 The Random Oracle Model

The idea behind the random oracle model is that by modeling primitives as DES, MD5 or SHA using the strong assumption that they (properly used/modified) behave like random oracles to model the properties that these primitives actually seems to have in practice, 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], the Fiat-Shamir heuristic for zero-knowledge proofs[FS86], and the efficient Byzantine agreement protocol of Cachin, Kursawe, and Shoup[CKS00].

In [BR93] it is said about the methodology of proving schemes secure in the random oracle model and then instantiating the oracle with a carefully chosen function in practice that *'It is our* thesis *that this method, when carried out, leads to secure and efficient protocols. Indeed, protocols constructed under this paradigm have so far proven "secure" in practice. But we stress that all claims of provable security are claims made within the random oracle model, and instantiating the oracle with h is only a heuristic whose success we trust from experience.'*

1.2.1 Our Result

The second result of our paper is a negative one. We show that no matter how the random oracle is instantiated in the standard model, our encryption protocol will not be non-committing. This shows that security in the random oracle model does not always allow a secure implementation in the standard model.

This is to some extend anticipated in [BR93], where it is said, that *'We stress that the protocol problem* Π *and protocol* P *must be "independent" of the hash function we are to use. It is easy to construct unnatural problems or protocols whose description and goals depend explicitely on* h *so that the protocol is secure in the random oracle model but fails when the random oracle is instantiated with the hash function. The notion of "independence" will not be formalized in this paper.'*

Our result is however of a different nature than this. First of all we do not find adaptively secure encryption in the non-erasure model, or our suggested protocol, unnatural, and second, the problem does not depend on H at all. Indeed we prove that any function will fail to be an "independent" function, no matter the definition of "independence".

Other examples of constructions which are secure in the random oracle model, and not in the standard model, were known prior to our work. We will compare our result to these, and will give a short discussion of the random oracle model based on this comparison.

1.3 Organization

In Section 2 we define collections of trapdoor permutations and state a lemma which will come in handy later. In Section 3 we give a short sketch of the multiparty computation model that we use and cast the problem of non-committing encryption in this framework. In Section 4 we describe our non-committing encryption protocol and prove it secure in the random oracle model. In Section 5 we prove that our protocol will never be non-committing in the standard model. Finally in Section 6 we compare our result to previous results, and give a short discussion the random oracle 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 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 algorithm, 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.

Key Generation 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 $\mathcal{G}$).

Give Up on *pk* On a give up request on pk, where pk was generated in a key generation request, A is given $r_{\mathcal{G}}$.

Element Generation for *pk* On an element generation request for pk, A receives a uniformly random element $y = f_{pk}(x)$, where x was generated as $x \leftarrow \mathcal{X}(pk, r_{\mathcal{X}})$.

Give Up on y On a give up request on y, where y was generated in an element generation request, A is given $r_{\mathcal{X}}$.

Wining If A manage to return an element x such that $y = f_{pk}(x)$, where pk is a key from a key generation request on which A has not given up and where y is from an element generation request on which A has not given up, then A wins the game.

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 Non-Committing Encryption

3.1 The General Framework

We will cast the non-committing encryption problem in the framework for universally composable asynchronous multiparty computation from [Can01]. In this framework the security of a protocol is defined in three steps.

First the real-life execution of the protocol is defined. Here the protocol π is modeled by n interactive Turing machines (ITMs) $P_1, \ldots, P_n$ called the parties of the protocols. Also present in the execution is an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ modeling the environment in which $\mathcal{A}$ is attacking the protocol. The environment gives inputs to honest parties, receives outputs from honest parties, and can communication with $\mathcal{A}$ at arbitrary points in the execution. Both $\mathcal{A}$ and $\mathcal{Z}$ are PPT ITMs.

Second an ideal process is defined. In the ideal process an ideal functionality $\mathcal{F}$ is present to which all the parties have a secure communication channel. The ideal functionality is an ITM defining the desired input-output behavior of the protocol. Also present is an ideal adversary $\mathcal{S}$, the environment $\mathcal{Z}$, and n so-called dummy parties $\tilde{P}_1, \ldots, \tilde{P}_n$ — all PPT ITMs. The only job of the dummy parties is to take inputs from the environment and send them to the ideal functionality and take messages from the ideal functionality and output them to the environment. This basically makes the ideal process a trivially secure protocol with the same input-output behavior as the ideal functionality.

The security of the protocol is then defined by requiring that the protocol emulates the ideal process. We say that the protocol securely realizes the ideal functionality.

The framework also defines the hybrid models, where the execution proceeds as in the real-life execution, but where the parties in addition have access to an ideal functionality. An important property of the framework is that an ideal functionality in a hybrid model can securely be replaced by a sub-protocol securely realizing that ideal functionality.

Below we add a few more details. For a more elaborate treatment of the general framework, see [Can01].

The framework as we will be using it models asynchronous authenticated communication over point-to-point channels, erasure free computation, and an active adaptive adversary. In the real-life execution all parties are assumed to share an open point-to-point channel. In the ideal process all parties are assumed to have a secure channel to the ideal functionality. These assumptions are modeled by the way the execution proceeds.

The environment $\mathcal{Z}$ is the driver of the execution. It can either provide a honest party, P_i or $\tilde{P}_i$, with an input or send a message to the adversary. If a party is given an input, that party is then activated. The party can then, in the real-life execution, send a message to another party or give an output to the environment. In the ideal process an activated party just copies its input to the ideal functionality and the ideal functionality is then activated, sending messages to the parties and the adversary according to it program. After the party and/or the ideal functionality stops, the environment is activated again.

If the adversary, $\mathcal{A}$ or $\mathcal{S}$, is activated it can do several things. It can corrupt a honest party, send a message on behalf of a corrupt party, deliver any message send from one party to another, or communicate with the environment. On corrupting a party the adversary sees the entire communication history of that party including the random bits used in the execution. After the corruption the adversary sends and receives messages on behalf of the corrupted party. The adversary controls the scheduling of the message delivery. In the real-life execution the adversary $\mathcal{A}$ can see the contents of all message and may decide which messages should be delivered and when — it can however not change messages or add messages to a channel. In the ideal process the adversary $\mathcal{S}$ cannot see the contents of the messages as the channels are assumed to be secure. It can only see that a message has been send and can then decide

when the message should be delivered, if ever.

If the adversary delivers a message to some party, then this party is activated and the environment resumes control when the party stops. At the beginning of the protocol all parties, the adversary, and the environment is given as input the security parameter k and random bits. Furthermore the environment is given an auxiliary input z. At some point the environment stops activating parties and outputs some bit. This bit is taken to be the output of the execution. We use $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ to denote the random variable describing the real-life execution and use $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ to denote the random variable describing the ideal process.

We are now ready to state the definition of securely realizing an ideal functionality. For this purpose let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble $\{\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ and let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$. We recall the definition of computationally indistinguishable distribution ensembles over $\{0,1\}$.

Definition 2 (indistinquishable ensembles) *We say distribution ensembles $X=\{X(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ and $Y=\{Y(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ over $\{0,1\}$ are indistinguishable (written $X\stackrel{c}{\approx}Y$) if for any $c\in\boldsymbol{N}$ there exists $k_0\in\boldsymbol{N}$ such that $|\Pr[X(k,z)=1]-\Pr[Y(k,z)=1]|<k^{-c}$ for all $k>k_0$ and all z.*

Definition 3 ([Can01]) *We say that π* securely realizes *$\mathcal{F}$ if for all real-life adversaries $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that for all environments $\mathcal{Z}$ we have that $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}\stackrel{c}{\approx}\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.*

3.2 The Random Oracle Model

The random oracle model is the real-life model extended with an ideal functionality H, called the random oracle, parameterized with two domains X and Y. On input $x\in X$ from any party (including the adversary) the random oracle outputs a uniformly random element $y\in Y$ independent of all other evaluations (except that if queried on the same x twice the same value y will be returned), to the calling party and the adversary.

In our protocol we will take $X=\{0,1\}^*$ and $Y=\{0,1\}^k$.

3.3 The Non-Committing Encryption Functionality

The non-committing-encryption (NCE) functionality is simply the oracle, which on input $(\texttt{send}, mid, j, m)$ from $\tilde{P}_i$ delivers $(\texttt{receive}, mid, i, m)$ to $\tilde{P}_j$ and delivers $(\texttt{receive}, mid, i, |m|)$ to $\mathcal{A}$, where $|m|$ denotes the bit-length of m.

4 The Protocol

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 $(\texttt{send}, mid, j, m)$ party P_i will generate a uniformly random element $x \leftarrow \mathcal{X}(pk_j)$, compute $(mid, f_{pk_j}(x), H(mid\|i\|j\|x) \oplus m)$, and send this value to P_j.[1]

Receive If P_j receives a message (mid, y, R) from P_i, where $y \in D_{pk_j}$,[2] then P_j computes $x = f^{-1}_{sk_j}(y)$ and $m = R \oplus H(mid\|i\|j\|x)$ and outputs $(\texttt{receive}, mid, i, m)$.

Theorem 1 *The above protocol securely realizes the NCE functionality in the random oracle 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 the real-life protocol to $\mathcal{A}$ and let $\mathcal{A}$ do the communication with $\mathcal{Z}$ to convince $\mathcal{Z}$ that it is viewing a real-life execution.

Since the protocol runs in the random oracle model, $\mathcal{S}$ will also have to simulate a random oracle H. It simply does this by defining $H(h)$ to be some uniformly random value $r \in \{0,1\}^k$, when $H(h)$ is needed. The definition of H is stored in a dictionary.

[1] We use $\|$ to denote some injective and easily parsable encoding $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$.

[2] We require from the collection of trapdoor permutations, that one can check $y \in D_{pk_j}$ given just the public key. Intuitively this is needed for security because a corrupt party P_i might use a honest P_j to determine whether $y \in D_{pk_j}$ by observing P_j's behavior on the message (mid, y, R) — remember that by the security preserving composition property, security in the framework of [Can01] implies that the protocol is secure in any context, in particular contexts where it can be observed whether P_j accepts the message or not. Therefore the information whether $y \in D_{pk_j}$ or not should already be available to P_j when sending the message.

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 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 indices of parties and $x \in D_{pk_j}$,[3] 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 $h = mid\|i\|j\|x$ and $(mid\|i\|j\|f_{pk_j}(x), r) \in$ `img` or h cannot be parsed as specified above and $(h, r) \in$ `raw`.

Because f_{pk_j} is a permutation, this representation is consistent — $H(h) = r$ will only become defined if explicitly recorded.

Equally important, this representation allows to define and evaluate H on $h = mid\|i\|j\|x$ given just (mid, i, j, y), where $y = f_{pk_j}(x)$. To look up H on $mid\|i\|j\|x$, simply look up $mid\|i\|j\|y$ in `img`, and to define H on $mid\|i\|j\|x$, simply add an entry with key $mid\|i\|j\|y$ to `img`. We call these manipulations oblivious.

The simulation proceeds as follows.

Random Oracle Evaluation If $\mathcal{A}$ asks for an evaluation of the random oracle 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 $(\texttt{send}, mid, j, |m|)$ from the NCE functionality we know that $\tilde{P}_i$ has input $(\texttt{send}, mid, j, m)$ for some $m \in \{0,1\}^{|m|}$ to the NCE functionality, which has then send $(\texttt{receive}, mid, i, m)$ to $\tilde{P}_j$.

[3] This is the point in the proof, where we use that $x \in D_{pk_j}$ can be checked given just the public key.

- If $\tilde{P}_j$ is corrupt, 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 $y = f_{pk_j}(x)$ and $x = \mathcal{X}(pk, r)$. 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 defines $H(mid\|i\|j\|x) := R \oplus m$.
  - 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 defines $H(mid\|i\|j\|x) := R \oplus m$.

  If $H(mid\|i\|j\|x)$ was already defined (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 $(\texttt{receive}, mid, i, m)$ to $\tilde{P}_j$, where $m = R \oplus H(mid\|i\|j\|f_{sk_j}^{-1}(y))$.

- If P_i is honest, then (mid, y, R) was send by $\mathcal{S}$ itself and in that case the message $(\texttt{receive}, mid, i, m)$ has already been send to $\tilde{P}_j$ in the ideal process. The simulator then simply delivers this message to $\tilde{P}_j$.
- If P_i is corrupt, then decrypt as follows. If H is not defined on $mid\|i\|j\|f_{sk_j}^{-1}(y)$, then obliviously define it to a uniformly random value. Then obliviously look up $H(mid\|i\|j\|f_{sk_j}^{-1}(y))$ and let $m = R \oplus H(mid\|i\|j\|f_{sk_j}^{-1}(y))$.

  Then input $(\texttt{send}, mid, j, m)$ to $\tilde{P}_i$ in the ideal process and deliver the message $(\texttt{receive}, mid, i, m)$ to $\tilde{P}_j$.

It is easy to see that if the simulation is not given up, then it is distributed exactly as a real-life execution. So, if $\mathcal{S}$ does not give up the simulation, then the final output of $\mathcal{Z}$ will be identically distributed in the real-life execution with adversary $\mathcal{A}$ and in the ideal process with adversary $\mathcal{S}$.

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[4] probability. This means that with significant probability

1. 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.

2. 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 exclude 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 Random Oracle Evaluation. The first definition of H on $mid\|i\|j\|x$ was therefore not oblivious, and thus $x = f^{-1}_{sk_j}(y)$ 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 Instantiating the Random Oracle

We have proven our construction secure in the random oracle model. The pending question is then whether we can construct a proper function family H, such that using a random function from H instead of a random oracle is secure — still yields a non-committing encryption scheme. Unfortunately we cannot. We can even prove that this is not due to our inability to construct a proper family H or our inability to prove that it works for a proper family H.

[4]We call a quantity significant if it is not negligible.

Put shortly, the reason is that the value

$$(f(x), H(mid\|i\|j\|x) \oplus m)$$

determines m uniquely for anyone who sees the value and knows H, as f is a permutation. For this particular instantiation, there might be other more basic reasons though for the scheme to be insecure — the scheme is not even guaranteed to semantically secure. We can remedy this by using a perfect one-way probabilistic hash function[Can97, CMR98]. This is a function H, where x is 'hashed' as $y = H(x, r)$, where r is a uniformly random string. A security notion for these hash functions guarantees that $H(x, r)$ looks uniformly random even if r is known and partial knowledge of x is known (which might be the case if e.g. $f(x)$ is known.) Examples of perfect one-way probabilistic hash functions are constructed in [Can97, CMR98], and a result from [Can97] allow us to conclude that if we encrypt as

$$(f(x), r, H(mid\|i\|j\|x, r) \oplus m) \ ,$$

for a particular type of perfect one-way probabilistic hash function, then the scheme is semantically secure.

Since, as detailed above, it might potentially by stronger to use probabilistic function families, we will do so. Still however, m is uniquely determined by $(f(x), r, H(mid\|i\|j\|x, r) \oplus m)$ and there is not much hope that the scheme should be non-committing.

One attempt to try to escape this is to use $n(n-1)$ functions $H_{i,j}$ instead, where $H_{i,j}$ is only known to P_i and P_j. These functions are then drawn from a probabilistic function family at the beginning of the protocol and distributed by a trusted party. However:

Theorem 2 *Let H be any probabilistic function family. Then the random oracle non-committing encryption scheme, using a secret random function from H for each pair of parties in place of the oracle, is not a non-committing encryption scheme.*

Proof: Assume for simplicity that for a fixed value of the security parameter, the description of functions from H all have some fixed length l (polynomial in k). The analysis generalizes to the case where one considers instead the expected length of a random function from H.

Consider the protocol with two parties P_1 and P_2 and consider the environment $\mathcal{Z}$, which activates P_1 on a uniformly random message $m \in \{0,1\}^{l+1}$. Consider the adversary $\mathcal{A}$, which first delivers all messages of

the protocol such that P_2 will output m. During this the adversary sees a message (y, r, R). The adversary will output (y, r, R) along with pk_2 to the environment. Then the adversary corrupts P_1 and P_2 and sends the internal view of P_1 and P_2 to the environment. This view includes x such that $y = f_{pk_2}(x)$, $H_{1,2}$ such that $m = H_{1,2}(x, r) \oplus R$,[5] $r_{\mathcal{G}}$ such that $(pk_2, sk_2) \leftarrow \mathcal{G}(k, r_{\mathcal{G}})$, and $r_{\mathcal{X}}$ such that $x \leftarrow \mathcal{X}(k, r_{\mathcal{X}})$. The environment outputs a bit e, where $e = 1$ iff it sees these values.

Note that

$$\Pr[\mathrm{REAL}_{\mathcal{A},\mathcal{Z}}(k, z) = 1] = 1 \ .$$

It is therefore (by far) enough to prove that for all[6] ideal process adversaries $\mathcal{S}$,

$$\Pr[\mathrm{IDEAL}_{\mathcal{S},\mathcal{Z}}(k, z) = 1] \leq \frac{1}{2} \ .$$

For this purpose, let $\mathcal{S}$ be any ideal process adversary. Let E denote the event that in the ideal process $\mathrm{IDEAL}_{\mathcal{S},\mathcal{Z}}(k, z)$, the ideal process adversary $\mathcal{S}$ returns to $\mathcal{Z}$ values $(y, r, R, pk_2, sk_2, r_{\mathcal{G}}, x, r_{\mathcal{X}}, H_{1,2})$ such that these values are on the expected syntactic form and $(pk_2, sk_2) = \mathcal{G}(k, r_{\mathcal{G}})$, $x = \mathcal{X}(k, r_{\mathcal{X}})$, and $y = f_{pk_2}(x)$. By definition of $\mathcal{Z}$ it is enough to prove that

$$\Pr[\mathrm{IDEAL}_{\mathcal{S},\mathcal{Z}}(k, z) = 1 | E] \leq \frac{1}{2} \ .$$

To prove this consider the ideal process $\mathrm{IDEAL}_{\mathcal{S},\mathcal{Z}}(k, z)$ conditioned on the event E. At some point in the simulation $\mathcal{S}$ will send a value (y, r, R, pk_2) to $\mathcal{Z}$. Then P_1 and P_2 are corrupted. This means that $\mathcal{S}$ is given m and delivers values $sk_2, r_{\mathcal{G}}, x, r_{\mathcal{X}} H_{1,2}$ to $\mathcal{Z}$, where $(pk_2, sk_2) = \mathcal{G}(k, r_{\mathcal{G}})$, $x = \mathcal{X}(k, r_{\mathcal{X}})$, and $y = f_{pk_2}(x)$. In particular this means that f_{pk_2} is guaranteed to be a permutation (with inverse $f^{-1}_{sk_2}$) and that y is in its image. This implies that the value of x became uniquely defined when $\mathcal{S}$ handed (y, r, R, pk_2) to $\mathcal{Z}$. Since (y, r, R, pk_2) was handed to $\mathcal{Z}$ before the parties were corrupted, the value is independent of m. Since the possible values of m is at least twice the possible values of H, this means that with probability at least $\frac{1}{2}$ the value of m is such that there exists no H for which $m = H(x, r) \oplus R$. This proves the theorem.□

Put shortly, the entropy of $H_{i,j}$ should be larger than the entropy of the communication between P_1 and P_2. In that case it would however be more efficient for the parties to just use Vernam's one-time pad encryption with the description of $H_{i,j}$ as the pad.

[5]For brevity we drop *mid*, i, and j as input to $H_{1,2}$.

[6]Though irrelevant, the result even holds for computationally unbounded ideal adversaries.

Note, that if such a protocol should be able to handle an unbounded number of bits of communication, then the pad should be generated during the evaluation using an adaptively secure coin-flip protocol. This is exactly the form of the non-committing encryption protocols in [Bea91, DN00], where a bit is communicated from P_i to P_j by P_i and P_j first generating a shared secret random bit R and P_i then sending $R \oplus m$ to P_j. This therefore yields no new view on the non-committing encryption problem.

6 A Short Discussion of the Random Oracle Model

6.1 Comparison to Previous Negative Results

Our result is primarily a negative one. We show that there exists protocols which are secure in the random oracle model and are not secure in the standard model, no matter how the random oracle is instantiated. Other examples of primitives secure in the random oracle model and not in the standard model were known prior to our work.

First of all, the Fiat-Shamir heuristic[FS86] for transferring tree-move public-coin zero-knowledge proofs into non-interactive ones is provably secure in the random oracle model, but by the result in [GK90] these protocols cannot be zero-knowledge in the standard model unless NP is in BPP.

Yet another negative result is that of [CGH98]. In [CGH98] an encryption scheme and a signature scheme are constructed, which are secure in the random oracle model, but is not secure in the standard model no matter the instantiation. Their schemes are however highly unnatural. They are constructed as to try to "detect" whether they are in the random oracle model or not, and then behave insecurely if they are in the random oracle model.

One strength of the result from [CGH98] over ours is that in [CGH98] it is the semantic security and the security against forgery of the encryption scheme respectively the signature scheme that are violated in the standard 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 does not carry over from the random oracle model to the standard model.

Another strength of the result from [CGH98] is that it uses the random oracle in an essentially weaker way than it is used in [FS86] and in our proof. Specifically, in [FS86] and in our simulator, it is used essentially that one can control the random oracle, by defining the value of $H(h)$ to be some value appropriately chosen by the simulator. E.g. we set it to $R \oplus m$ to be consistent with some plaintext learned by the simulator after H "should" have been defined.

Since, the property that the random oracle can be lazily defined by the simulator is a property that a fixed function is guaranteed *not* to have, one could therefore get the impression that the negative results are due to this very strong use of the control over the oracle in the proof.

Indeed the proof in [CGH98] also exploits this control over random oracle, but only indirectly through the use of the CS-proofs of [Mic00] for the random oracle model. It is however easy to see that the result from [CGH98] still stands if these "non-interactive" proofs for the random oracle model are replaced by interactive CS-proofs (which can be constructed in the standard model under the assumption that collision resistant hash functions exits[Mic00]).

After this modification, the proof in [CGH98] does not exploit the control over the oracle. Indeed the oracle could as well be an oracle external to the simulator/proof, i.e. a uniformly random function to which also the simulator/proof only has oracle access.

True, the use of interactive CS-proofs makes the schemes in [CGH98] even more unnatural (now sending a ciphertext/signature to the decrypter/verifier involves a four-move protocol.) However, their result shows that even if the simulator/proof also uses the oracle in a black-box manner natural security properties still does not carry over to the standard model.

6.2 Conclusion

It seems that the thesis of [BR93], even though not phrased like that, is that *standard* security properties of *natural* (encryption and signature) schemes carry over from the random oracle model to the standard model by a careful instantiation of the random oracle, at least for all practical reasons — i.e. even though the schemes may not facilitate a formal proof of security, they are "secure" to use in practice. Indeed this thesis is what e.g. allows us to conjecture the semantic security-in-practice of the OAEP-RSA scheme[BR95].

We do not think that any of the negative results above contradict this thesis, either the schemes are highly unnatural or the properties lost in the instantiation are properties not initially intended to be considered in the framework.

An interesting open question is therefore whether we can identify "standard security properties" of some subset of "natural schemes" (all this maybe being defined by a restriction on how the oracle is used in the scheme and/or in the simulator,) which carry over to the standard model by a "careful instantiation".

References

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

[ACM98] *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, Dallas, TX, USA, 24–26 May 1998.

[Bea91] D. Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *Advances in Cryptology - Crypto '91*, pages 377–391, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 576.

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

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology - EuroCrypt '92*, pages 307–323, Berlin, 1992. Springer-Verlag. Lecture Notes in Computer Science Volume 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 Alfredo De Santis, editor, *Advances in Cryptology -*

EuroCrypt '94, pages 92–111, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science Volume 950.

[Can97] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burt Kaliski, editor, *Advances in Cryptology - Crypto '97*, pages 455–469, Berlin, 1997. Springer-Verlag. Lecture Notes in Computer Science Volume 1294.

[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 ACM [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 ACM [ACM98], pages 209–218.

[CKS00] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC 2000)*, pages 123–132. ACM, July 2000.

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In ACM [ACM98], pages 131–140.

[DN00] Ivan Damgård and Jesper B. Nielsen. Improved noncommitting encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology - Crypto 2000*, pages 432–450, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1880.

[FS86] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - Crypto '86*, pages 186–194, Berlin, 1986. Springer-Verlag. Lecture Notes in Computer Science Volume 263.

[GK90] O. Goldreich and H. Krawczyk. On the composition of zero knowledge proof systems. In *Proceedings of ICALP 90*, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science Volume 443.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

Recent BRICS Report Series Publications

RS-01-47 **Jesper Buus Nielsen.** ***Non-Committing Encryption is Too Easy in the Random Oracle Model.*** **December 2001. 20 pp.**

RS-01-46 **Lars Kristiansen.** ***The Implicit Computational Complexity of Imperative Programming Languages.*** **November 2001. 46 pp.**

RS-01-45 **Ivan B. Damgård and Gudmund Skovbjerg Frandsen.** ***An Extended Quadratic Frobenius Primality Test with Average Case Error Estimates.*** **November 2001. 43 pp.**

RS-01-44 **M. Oliver Möller, Harald Rueß, and Maria Sorea.** ***Predicate Abstraction for Dense Real-Time Systems.*** **November 2001. 27 pp.**

RS-01-43 **Ivan B. Damgård and Jesper Buus Nielsen.** ***From Known-Plaintext Security to Chosen-Plaintext Security.*** **November 2001. 18 pp.**

RS-01-42 **Zoltán Ésik and Werner Kuich.** ***Rationally Additive Semirings.*** **November 2001. 11 pp.**

RS-01-41 **Ivan B. Damgård and Jesper Buus Nielsen.** ***Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor.*** **October 2001. 43 pp.**

RS-01-40 **Daniel Damian and Olivier Danvy.** ***CPS Transformation of Flow Information, Part II: Administrative Reductions.*** **October 2001. 9 pp.**

RS-01-39 **Olivier Danvy and Mayer Goldberg.** ***There and Back Again.*** **October 2001. 14 pp.**

RS-01-38 **Zoltán Ésik.** ***Free De Morgan Bisemigroups and Bisemilattices.*** **October 2001. 13 pp.**

RS-01-37 **Ronald Cramer and Victor Shoup.** ***Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption.*** **October 2001. 34 pp.**

RS-01-36 **Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob.** ***Cache Oblivious Search Trees via Binary Trees of Small Height.*** **October 2001.**

RS-01-35 **Mayer Goldberg.** ***A General Schema for Constructing One-Point Bases in the Lambda Calculus.*** **September 2001. 6 pp.**