Zero Knowledge in the Random Oracle Model, Revisited

Hoeteck Wee*

Queens College, CUNY
hoeteck@cs.qc.cuny.edu

Abstract. We revisit previous formulations of zero knowledge in the random oracle model due to Bellare and Rogaway (CCS '93) and Pass (Crypto '03), and present a hierarchy for zero knowledge that includes both of these formulations. The hierarchy relates to the programmability of the random oracle, previously studied by Nielsen (Crypto '02).

- We establish a subtle separation between the Bellare-Rogaway formulation and a weaker formulation, which yields a finer distinction than the separation in Nielsen's work.
- We show that zero-knowledge according to each of these formulations is not preserved under sequential composition. We introduce stronger definitions wherein the adversary may receive auxiliary input that depends on the random oracle (as in Unruh (Crypto '07)) and establish closure under sequential composition for these definitions. We also present round-optimal protocols for NP satisfying the stronger requirements.
- Motivated by our study of zero knowledge, we introduce a new definition of proof of knowledge in the random oracle model that accounts for oracle-dependent auxiliary input. We show that two rounds of interaction are necessary and sufficient to achieve zero-knowledge proofs of knowledge according to this new definition, whereas one round of interaction is sufficient in previous definitions.
- Extending our work on zero knowledge, we present a hierarchy for circuit obfuscation in the random oracle model, the weakest being that achieved in the work of Lynn, Prabhakaran and Sahai (Eurocrypt '04). We show that the stronger notions capture precisely the class of circuits that is efficiently and exactly learnable under membership queries.

Keywords. zero-knowledge, random oracle model, sequential composition, obfuscation

1 Introduction

The random oracle (RO) model, introduced by Fiat and Shamir [10] and refined by Bellare and Rogaway [3], was proposed as a framework for designing and analyzing

* Work done while visiting Tsinghua University, Beijing, China.

cryptographic schemes that offers a trade-off between provable security and practical efficiency. In this model, every party has oracle access to a truly random function. With this additional functionality, many cryptographic problems admit more efficient solutions than in the standard model, along with considerably simpler proofs of security [3, 4, 24, 11]. In practice, the idealized random function is instantiated using a "good" cryptographic hash function, like SHA-1 or a variation thereof. There are also cryptographic problems for which we have partial solutions in the random oracle model but not in the standard model, most notably that of circuit obfuscation [1, 19, 20]. In both cases, proofs in the random oracle model do not guarantee security or feasibility in the standard model (and in fact, there has been substantial evidence to the contrary [6, 15, 2]); nonetheless, the model provides a useful idealized test-bed for analyzing cryptographic schemes.

As a first step towards establishing security, it is necessary to define security in the random oracle model. A naive extension of a definition in the standard model may affect the semantics of the underlying notion of security. Consider the case of *zero-knowledge proofs*, namely proofs that yield no knowledge beyond the validity of the assertion proved [17]. Formally, an interactive protocol is zero-knowledge if there exists a simulator that can simulate the behavior of every, possibly cheating, verifier without access to the prover, such that its output is indistinguishable from the output of the verifier after having interacted with the honest prover. In the standard model, a zero-knowledge proof is necessarily *deniable*, in that the protocol's transcript does not constitute any evidence of the interaction, since any party could have generated the transcript by himself. However, the Bellare-Rogaway formulation of zero-knowledge in the random oracle model does not imply deniability, since the simulator can *choose* the random oracle [22, 21]. In particular, the formulation allows for (non-trivial) one-round zero-knowledge proof systems, and the transcript of such a protocol constitutes evidence of participating in the protocol, contradicting deniability.

In this work, we revisit two aspects of formulating zero-knowledge in the random oracle model. The first relates to defining security in the random oracle model and in particular, what it means to choose the random oracle, an issue first addressed by Nielsen [21]. The second relates to a different aspect of zero-knowledge proofs, namely, we want the zero-knowledge guarantee to hold even if the verifier may have some additional a priori information about the input. The need to account for such auxiliary input, which arises in typical applications such as sequential repetitions of a protocol, was articulated in the work of Goldreich and Oren [14] and again in that of Unruh [25]. While the Bellare-Rogaway formulation of zero-knowledge does take into account auxiliary input, it does not allow for dependencies between the auxiliary input and the random oracle, which arise for instance, when the auxiliary input is a transcript of a previous interaction using the oracle.

1.1 Programmability in the random oracle model

There are two reasons why, in the simulation-based paradigm, it is easier to achieve security in the random oracle model:

– the simulator can see the queries parties make to the random oracle;
– the simulator can choose the answers to these queries.

The second is what we refer to as *programming* the random oracle, and may be qualified in several different ways. Suppose our goal is to simulate a transcript $\text{RO}(s)$, namely the evaluation of the random oracle RO at some value s. Our intuition about the random oracle as a truly random function indicates that picking a truly random string τ should suffice (essentially choosing the evaluation of RO at s to be τ), and indeed, no distinguisher - even computationally unbounded ones - can distinguish a truly random string from $\text{RO}(s)$, provided the distinguisher does not get access to RO. On the other hand, if we give the distinguisher access to RO, then the only "good" simulation of the transcript is $\text{RO}(s)$, and the simulation must query RO at s. This is because the distinguisher may have s hardwired into it, then queries RO at s and checks whether the answer matches the transcript. In this setting, the simulator does not get to choose the answers to oracle queries. To distinguish between these two notions of security, we will refer to the former as the *fully programmable* random oracle (FPRO) model, and the latter as the *non-programmable* random oracle (NPRO) model (as coined by Nielsen [21]).

In the case where we allow the simulator to choose the answers to oracle queries, we may still impose an additional requirement, namely that the simulator must output its choices of these query/answer pairs. In the above example, whether the simulator chooses the output of RO at s to be some random string τ, its output will include the transcript τ, along with the list (s, τ), corresponding to the query s and answer τ. This is in fact the notion of programmability raised by Bellare and Rogaway [3] for zero-knowledge, and we will refer to this as the *explicitly programmable* random oracle (EPRO) model. We defer a precise definition to the body of the paper, but note at this point that security in the non-programmable random oracle model (strongest security guarantee) implies security in the explicitly programmable random oracle model, which in turn implies security in the fully programmable random oracle model (weakest security guarantee).

1.2 Our contributions and techniques

Hierarchy for zero knowledge. We begin with a simple and unified framework for defining zero knowledge in the three variants of the random oracle model, and then present a (perhaps surprising) separation for zero knowledge in the fully programmable and explicitly programmable random oracle models. This yields a finer separation than that in Nielsen's work [21], and complements Pass's separation for zero knowledge in the explicitly programmable and non-programmable random oracle models.

Auxiliary input and sequential composition. Following the work of Goldreich et al. [14, 13] for zero-knowledge in the standard model, we use closure under sequential composition as a yardstick for evaluating formulations of zero-knowledge. We show that zero-knowledge in all three variants of the random oracle model are not closed

under sequential composition[1]. This motivates a new formulation of zero-knowledge in the random oracle which allows for auxiliary inputs that depend on the oracle, as was done in [25] for one-way functions, encryption and other primitives.[2] We show that for efficient-prover protocols, zero-knowledge with oracle-dependent auxiliary input in the explicit-programmable and non-programmable random oracle models are preserved under a polynomial number of sequential repetitions. We also present round-optimal protocols for NP satisfying the new formulations of zero-knowledge.

Proofs of knowledge. Our constructions demonstrating that previous formulations of zero knowledge are not closed under sequential composition implicitly rely on a non-interactive zero-knowledge "proofs of knowledge" in the random oracle model. Specifically, non-interactive protocols are necessarily malleable (without unique identifiers), and the cheating verifier can generate a convincing proof of knowledge by copying one sent by the prover in a previous iteration of the protocol. This motivates a new formulation of proof of knowledge in the random oracle model that takes into account oracle-dependent auxiliary input. We show that two rounds of interaction are necessary and sufficient to achieve zero-knowledge proofs of knowledge according to this new definition.

Circuit obfuscation. We extend our framework for programmability to circuit obfuscation[3] in the random oracle model [19, 1], and note that the obfuscator constructions of Lynn et al. [19] achieve security in the fully programmable random oracle model. Next, we show circuit obfuscation in the explicit-programmable random oracle model can only be realized for classes of circuits that are efficiently and exactly learnable under membership queries, and for these classes, obfuscation may be (trivially) realized in the plain model, so the characterization is exact. We find it surprising that we can have non-trivial constructions in the explicitly programmable model for zero knowledge but not for circuit obfuscation.

---

[1] That this may be the case has been previously noted (e.g. [22]), but to our knowledge, there has been no formal (published) proof.

[2] For the primitives considered in [25], the random oracle is typically only used in the proof of security. Specifically, Unruh [25] does not explicitly address primitives with a simulation-based notion of security, which is the focus of this work and where the random oracle is also exploited in constructing a simulator. On the other hand, Unruh considers a stronger notion of oracle-dependent auxiliary input, where a polynomial bound is imposed only on the output length of the machine generating the auxiliary input and not its query complexity.

[3] We use the term obfuscation to refer to the stronger notion of obfuscation against general adversaries, instead of obfuscation against predicate adversaries [1, 26]. In the standard model, only classes of circuits that are efficiently and exactly learnable under membership queries are obfuscatable against general adversaries [26]. The result also extends to the fully-programmable RO model.

1.3 Discussion

Formulating zero-knowledge. A general framework for defining security in the random oracle model was presented by Nielsen [21], based on augmenting the universally composable (UC) framework [5] with a random oracle functionality. This guarantees composability. As pointed out by Pass [22], deniability is not guaranteed in this framework. Nielsen also defined security with a non-programmable random oracle, where the environment in the UC framework is also given access to the random oracle. This offers deniability, but may no longer guarantee universal composability.

Instead of adopting Nielsen's formulation, we consider a minimal framework (based on [3, 14]) for which we can provide the weaker guarantee of sequential composition. The simplicity allows us to focus on how the random oracle is incorporated differently in each of [21, 3, 22]. In addition, it offers several conceptual advantages: it offers modularity (which allows us to decouple the zero-knowledge property from the proof-of-knowledge property and other properties implied by UC zero-knowledge, and for impossibility results, these distinctions are particularly important) and reinforces the theme of this work, that of understanding how semantics can change between the standard model and the random oracle model. Furthermore, our framework is simple enough to be applied to circuit obfuscation, for which we have very few non-trivial positive results, let alone constructions that compose arbitrarily (which probably only exist for trivially obfuscatable families of circuits).

Sequential composition not the end-goal. We recall the arguments used in [14] to motivate the study of auxiliary-input zero-knowledge: first, it fully captures the intuitive meaning of the concept of zero-knowledge; and second, this stronger requirement is necessary when a zero-knowledge protocol is used as a sub-protocol within larger cryptographic protocols[4]. It is for these same reasons that we pursue a formulation of zero-knowledge in the random oracle model that incorporates auxiliary input (refer to [25] for additional arguments). Indeed, we regard our sequential composition lemma as evidence that we have properly accounted for auxiliary input in our formulation and not a goal in and of itself. Similarly, constructing protocols for NP that remain zero-knowledge under sequential composition should not be an objective in itself.[5] Neither should a generic method for transforming protocols that are zero-knowledge into another that remain zero-knowledge under sequential composition.

On "explicit programmability". From previous work [3, 22, 19, 26], we know that allowing the simulator to program the random oracle is necessary and sufficient for

[4] One may ask, why not aim for universal composability then? This is addressed in the previous paragraph, and as with previous work in the standard setting, we feel that zero-knowledge w.r.t. auxiliary input is indeed the right compromise.

[5] All "natural" zero-knowledge protocols for NP in the RO model (in the [3] sense) remain zero-knowledge under sequential, even concurrent, composition, but this does not obliterate the need for the "right" definition. After all, when auxiliary-input zero-knowledge was introduced, all known zero-knowledge protocols were black-box and therefore remained zero-knowledge under sequential composition.

one-round zero-knowledge protocols for NP and obfuscating point functions in the random oracle model. However, while explicit programmability is sufficient for zero-knowledge, we show that full programmability is necessary for the latter. This means that the reason we are able to realize non-trivial circuit obfuscation in the random oracle model comes not only from programming the random oracle, but also from not having to specify explicitly how we program the random oracle.

The issue of explicit programmability also arises in the study of sequential composition. To obtain zero-knowledge that is closed under a polynomial number of sequential compositions, it appears that explicit programmability is necessary, in addition to properly accounting for auxiliary input.

Self-composition for circuit obfuscation. Lynn et al. introduce self-composition for circuit obfuscation [19], a notion of composition analogous to sequential composition. In addition, they give an obfuscator for point functions in the random oracle model that is not 2-self-composing. This is because the construction is not a valid obfuscator w.r.t. dependent auxiliary input. To obtain polynomial self-composition for obfuscation using techniques in this work, we will need a definition that incorporates both oracle-dependent auxiliary input and explicit programmability.

2 Preliminaries

A negligible function is a function of the form $n^{-\omega(1)}$, and is denoted $\mathrm{neg}(n)$. We use PPT as an abbreviation for a probabilistic (strict) polynomial-time Turing machine. We also consider the nonuniform and oracle analogues, which we denote by nonuniform PPT and oracle PPT respectively. In probability expressions that involve a probabilistic computation, the probability is also taken over the internal coin tosses of the underlying computation. We refer the reader to [12] for definitions of interactive proof systems, zero-knowledge, proofs of knowledge and witness-indistinguishability (WI) in the standard model. For a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, the *language associated with* R is $L_R = \{x : \exists y \ (x,y) \in R\}$.

3 Zero knowledge in the random oracle model

In this section, we present our hierarchy of formulations for zero knowledge in the RO model, along with those that account for oracle-dependent auxiliary input. We begin with several formalisms we will use in defining zero knowledge:

- We use RO to denote the random oracle and ε to denote an oracle that returns the empty string on all inputs.
- Given a function $f : \{0,1\}^* \to \{0,1\}^*$ and a list $\ell \subseteq \{0,1\}^* \times \{0,1\}^*$, we use $f[\ell]$ to denote a function that agrees with f everywhere except on inputs specified

by the set ℓ. Specifically,

$$f[\ell](x) = \begin{cases} y & \text{if } \exists! y \text{ such that } (x,y) \in \ell \\ f(x) & \text{otherwise} \end{cases}$$

Informally, we refer to $f[\ell]$ as the function obtained by programming f on the inputs in ℓ. In the definition of zero-knowledge, the simulator generates a pair (τ, ℓ): the simulator programs the random oracle on the inputs in ℓ, and τ corresponds to the view of the cheating verifier while interacting with the prover using the oracle $\text{RO}[\ell]$.

– We allow the auxiliary input to be generated by a nonuniform oracle PPT Z (the nonuniformity allows for auxiliary information that may depend on the input instance) which we refer as the *auxiliary input machine*. We will give Z oracle access to either ε or RO, depending on whether we allow the auxiliary input to depend on the random oracle.

Definition 1 (zero-knowledge [3, 22]). *Let (P, V) be an interactive protocol for a language $L = L_R$. Let V^*, S, Z, D be oracle PPTs. Given $(x, w) \in R$, we define $\mathsf{diff}^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3}_{V^*,S,Z,D}(x, w)$ to be the quantity*

$$\begin{aligned} &\Pr_{\text{RO}}\left[z \leftarrow Z^{\mathsf{O}_1}(1^{|x|});\ \tau \leftarrow (P^{\text{RO}}(w), V^{*\text{RO}}(z))(x);\ D^{\mathsf{O}_2}(x,\tau,z) = 1\right] \\ &\quad - \Pr_{\text{RO}}\left[z \leftarrow Z^{\mathsf{O}_1}(1^{|x|});\ (\tau,\ell) \leftarrow S^{\text{RO}}(x,z);\ D^{\mathsf{O}_3}(x,\tau,z) = 1\right] \end{aligned}$$

We say that (P, V) is zero-knowledge in the fully programmable random oracle model (FPRO) *if for every oracle PPT V^*, there exists an oracle PPT S such that for all $(x, w) \in R$ and for all nonuniform oracle PPTs Z and D, $\mathsf{diff}^{\varepsilon,\varepsilon,\varepsilon}_{V^*,S,Z,D}(x, w)$ is negligible (as a function of $|x|$). In addition, we obtain* zero-knowledge *in the:*

explicitly programmable RO (EPRO) model	*if* $\mathsf{O}_1 = \varepsilon, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}[\ell]$
non-programmable RO (NPRO) model	*if* $\mathsf{O}_1 = \varepsilon, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}$
FPRO model w.r.t dependent auxiliary input	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \varepsilon, \mathsf{O}_3 = \varepsilon$
EPRO model w.r.t dependent auxiliary input	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}[\ell]$
NPRO model w.r.t. dependent auxiliary input	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}$

For simplicity, we will also refer to the respective notions of zero-knowledge as FPRO zero-knowledge, EPRO zero-knowledge, NPRO zero-knowledge, auxiliary-input FPRO zero-knowledge, auxiliary-input EPRO zero-knowledge, and auxiliary-input NPRO zero-knowledge.

Zero-knowledge in the FPRO *model.* This definition captures the weakest requirement, in that the simulator may choose the random oracle in the simulated transcript, as long as it "looks" random. We point out that we require simulating the output of the cheating verifier, but not the random oracle used in the simulated transcript. This is equivalent to a definition wherein the simulator S is given access to RO. Since

the distinguisher does not have access to RO, the simulator can simply generate a random oracle by itself, so giving the simulator access to RO does not give the simulator any extra power. Note that this definition also constitutes a relaxation of the UC framework augmented with a random oracle functionality (namely, that obtained by replacing the interactive environment with a non-interactive distinguisher) [21, 5].

Zero-knowledge in the EPRO *model.* The main qualitative difference between FPRO zero-knowledge and EPRO zero-knowledge[6] is that the simulator is required to completely specify a simulated random oracle (namely $\text{RO}[\ell]$) in the latter, which the distinguisher is given access to.[7] We require that S specifies ℓ explicitly, which implies a polynomial bound on the size of ℓ. On the other hand, the oracle $\text{RO}[\ell]$ is specified implicitly. EPRO zero-knowledge is equivalent to the Bellare-Rogaway formulation, except the latter does not give the simulator oracle access to RO. As with zero-knowledge in the FPRO model, this does not make any qualitative difference as the simulator can simply generate random answers to the RO queries and add these query-answer pairs to the list ℓ.

Zero-knowledge in the NPRO *model.* Here, the simulator is not allowed to choose the random oracle in the simulated transcript. This implies deniability, and is equivalent to Pass's formulation [22]. It is a special case of EPRO zero-knowledge with $\ell = \emptyset$. For efficient-prover protocols, the NPRO zero-knowledge requirement is equivalent to requiring that the following quantity be negligible [22, 8]:

$$\mathrm{E}_{\text{RO}}\Big[\big|\Pr[z \leftarrow Z^{\mathsf{O}_1}(1^{|x|});\ D^{\text{RO}}(x,(P^{\text{RO}}(w),V^{*\text{RO}}(z))(x),z)=1] \\ -\Pr[z \leftarrow Z^{\mathsf{O}_1}(1^{|x|});\ D^{\text{RO}}(x,S^{\text{RO}}(x,z),z)=1]\big|\Big]$$

This is also true w.r.t. dependent auxiliary input.

Incorporating dependent auxiliary input. Incorporating dependent auxiliary input provides some guarantee of "independence" between the queries made to the random oracle in the protocol and prior queries, even though we do not know what the prior queries are. To achieve this definition, we construct simulators that program the random oracle on inputs that have not been previously queried by Z (here, we exploit the polynomial bound on the query complexity of Z). Unlike the case without auxiliary input, it is essential that we provide the simulator for zero-knowledge and EPRO zero-knowledge with oracle access to RO so that the simulator may generate transcripts that are consistent with the output from Z.

Verifier's view. A common convention in defining zero-knowledge in the standard model is to use $(P^{\text{RO}}(w), V^{*\text{RO}}(z))(x)$ to denote the view of the verifier V^*, which

[6] Indeed, making this distinction in the UC framework would require clumsy modifications.

[7] For some secret value s and a random RO, we may easily simulate a view of $\text{RO}(s)$ with a random string. However, in order to simulate a view of $\text{RO}(s)$ along with an oracle that is consistent with this view, we will need to either query RO at s or program RO at s; either operation requires "knowing" s.

consists of the protocol's transcript and the verifier's random tape, instead of the output of the verifier. This is because we may incorporate the computation of the output from the view into the distinguisher. This argument does not necessarily apply to definitions in the RO model. In this case, the distinguisher does not have access to RO and may not be able to compute the output from the view.[8] Therefore, we reserve $(P^{\text{RO}}(w), V^{*\text{RO}}(z))(x)$ to denote the output of the verifier.

A note on black-box simulation. As with previous works on zero knowledge in the RO model, we will establish the zero-knowledge property via black-box simulation, except we will allow the simulator to see the oracle queries made by the cheating verifier. This is consistent with the definition of zero knowledge because the simulator can execute the code of the cheating verifier and observe the oracle queries made during the executions. This is a crucial advantage over mere black-box simulation of the cheating verifier in the standard model. On the other hand, we do not allow the simulator to see the oracle queries made by Z. Consider a typical application, namely that of sequential repetitions of the protocol. Here, the auxiliary input is a transcript from previous executions of the protocol and may therefore depend on the oracle RO. The cheating verifier receives the transcript, but does not gain access to the private coin tosses used to generate the transcript. The distinction arises from the fact that we allow the simulator to depend on the cheating verifier but not on Z.

4 Zero-knowledge protocols and separations

Several constructions of zero-knowledge protocols for NP in the RO model were given in [3, 22, 11]. It is straight-forward to verify that the zero-knowledge protocol in [3] is also auxiliary-input EPRO zero-knowledge. In an unpublished work [23], Pass determined the round-complexity of auxiliary-input NPRO zero-knowledge protocols for NP. We summarize these results below:

Theorem 1 (protocols [3, 22, 23]). *Assuming the existence of one-way functions, there exist:*

- *a one-round proof of knowledge protocol for* NP *that is auxiliary-input* EPRO *zero-knowledge (moreover, we may assume that the knowledge extractor is straight-line and runs in strict polynomial time* [22, 11]*);*
- *a two-round protocol for* NP *that is* NPRO *zero-knowledge; and*
- *a 3-round protocol for* NP *that is auxiliary-input* NPRO *zero-knowledge.*

Furthermore, each of these protocols has perfect completeness, negligible soundness, and an efficient prover.

[8] Simply requiring that the verifier's query/answer pairs be included in its view may not be sufficient as we may also need the prover's query/answer pairs.

Theorem 2 (triviality [22, 23]). *Only languages in* BPP *have a one-round* NPRO *zero-knowledge protocol or a 2-round auxiliary-input* NPRO *zero-knowledge protocol.*

We outline the proofs in [23]. The 3-round auxiliary-input NPRO zero-knowledge protocol for NP is based on the 2-round NPRO zero-knowledge protocol in [22] except we have the prover pick a random prefix α in the first round, and prepend α to all prover's and verifier's queries to the random oracle. The proof of Theorem 2 follows essentially from the fact that the proofs of the analogous statments in the standard model [14] relativizes.

Next, we state our first result, separating FPRO and EPRO zero-knowledge.

Theorem 3. *Assuming the existence of one-way permutations, there exists a protocol that is auxiliary-input* FPRO *zero-knowledge but not* EPRO *zero-knowledge.*

Proof. Let π be a one-way permutation, and consider the following protocol for the relation $R = \{(x, w) \mid x = \pi(w)\}$, where $L_R = \{0,1\}^*$ (note that soundness holds vacuously):

Common input: An instance $x \in \{0,1\}^n$.
Prover's private input: A witness $w \in \{0,1\}^n$.
$P \to V$: Sends $\alpha \xleftarrow{\mathrm{R}} \{0,1\}^n$.
$V \to P$: Sends $\tau \xleftarrow{\mathrm{R}} \{0,1\}^n$.
$P \to V$: If $\tau = \mathrm{RO}(\alpha \circ w)$, send w; else, send $\mathrm{RO}(\alpha \circ w)$.
verification: V always accepts.

To see that this protocol is auxiliary-input FPRO zero-knowledge[9], fix a cheating verifier V^* (along with its random tape and an auxiliary input z from Z), pick a random α, and simulate the execution of V^*, forwarding the oracle queries made by V^* to RO, until we obtain its first message τ. During the simulation, we also check if any of V^*'s queries matches $\alpha \circ w$ (which we can check efficiently given x). If so, we would have recovered w, and may successfully compute the output of V^*. If we do not manage to recover w, we simulate the prover's response with a random string $\tau' \in \{0,1\}^n$ and continue to simulate the execution of V^*, forwarding all oracle queries to RO, unless the query matches $\alpha \circ w$, in which case we respond with τ'. This is ok because with probability $1 - \mathrm{neg}(n)$ over α, none of the queries made by Z has prefix α. This completes the description of the zero-knowledge simulator.

Suppose on the contrary that the protocol is EPRO zero-knowledge, and consider the simulator S that outputs the view of the honest verifier. Fix $x \in L$, and consider a distinguisher with $w = \pi^{-1}(x)$ hardwired into it. Then, S must output a transcript that contains $\mathrm{RO}[\ell](\alpha \circ w)$ with probability $1 - \mathrm{neg}(n)$. For the latter, S must with high probability, either query RO at $\alpha \circ w$ or output a list ℓ that contains the string $\alpha \circ w$. In

[9] Informally, the prover uses $(\alpha, \mathrm{RO}(\alpha \circ w))$ to check whether the verifier already "knows" w.

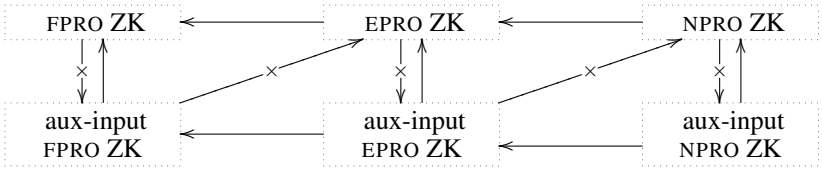

Fig. 1. Relations between different variants of zero knowledge in the RO model, assuming the existence of one-way permutations. An arrow is an implication, and a crossed arrow indicate separation. We stress that the relations refer to protocols satisfying the respective notion of zero-knowledge.

both cases, we may derive a PPT that on input x, outputs $\pi^{-1}(x)$ with high probability, which contradicts π being one-way. □

5 Sequential composition fails without dependent auxiliary input

In this section, we present zero-knowledge protocols which are no longer zero-knowledge when executed twice sequentially. The protocols are similar in spirit to that in [9], a zero-knowledge protocol in the standard model that is no longer zero-knowledge when executed twice in parallel. The protocols exploit zero-knowledge proofs of knowledge (which may be realized non-interactively in the random oracle model), using these proofs as the auxiliary input which a cheating verifier could use to "gain knowledge". Specifically, the prover will send the verifier a zero-knowledge proof of knowledge of the witness, and the cheating verifier will copy this proof to "claim" knowledge of the witness. The apparent contradiction arises from a problem in the definition of proofs of knowledge in the random oracle model, an issue we will address in Section 7.

Theorem 4. *Assuming the existence of one-way functions,* FPRO *zero knowledge,* EPRO *zero knowledge, and* NPRO *zero knowledge are not closed under sequential composition.*

Proof (sketch). We begin by constructing an EPRO zero-knowledge protocol that is no longer zero-knowledge when composed twice. The protocol is for the language L corresponding to the relation $R = \{(x, w) \mid x = f(w)\}$, where f is a one-way function, and we use as an underlying protocol a one-round EPRO zero-knowledge proof of knowledge protocol (from Theorem 1).

Common input: An instance $x \in \{0,1\}^n$.
Prover's private input: A witness $w \in \{0,1\}^n$.
$V \rightarrow P$: Send a random string τ.
$P \rightarrow V$: If τ is an EPRO zero-knowledge proof of knowledge that $x \in L$, send w; else, send an EPRO zero-knowledge proof of knowledge that $x \in L$.
verification: V accepts if it receives either w such that $f(w) = x$ or an accepting proof of knowledge that $x \in L$.

To prove EPRO zero-knowledge, the simulator runs the cheating verifier to obtain the first message τ. If τ is an accepting proof of knowledge for $x \in L$, the simulator runs the knowledge extractor to obtain a valid witness w. Otherwise, the simulator runs the zero-knowledge simulator for the underlying zero-knowledge protocol to generate the second-round message. We would actually require that the underlying zero-knowledge protocol be auxiliary-input EPRO zero-knowledge, which is ok.

To see that this protocol is not zero-knowledge when composed twice, consider the cheating verifier V^* that sends a random string in the first execution, and sends the prover's response as its first message in the second execution. For all $x \in L$, the transcript between the honest prover and V^* (for two sequential repetitions) will contain $f^{-1}(x)$ with probability 1. That f is one-way implies that there is no PPT simulator for two sequential repetitions of this protocol.

A similar modification to Pass's 2-round NPRO zero-knowledge protocol for NP yields a 2-round NPRO zero-knowledge protocol that is no longer NPRO zero-knowledge when composed twice. □

Remark 1. Our counter-example are efficient-prover protocols (looking ahead, our sequential composition theorem only holds for efficient-prover protocols). This mean that a cheating verifier (which is allowed to be nonuniform) can in fact simulate an interaction between the honest prover and the honest verifier. This is different from the counter-example in [13] wherein the cheating verifier cannot simulate such an interaction. There, the honest prover is allowed nonuniformity, whereas the cheating verifier is not, and the counter-example exploits the fact that the honest prover is "more powerful" than the class of cheating verifiers in an essential manner. This distinction was previously made in [9].

6 Sequential composition with dependent auxiliary input

Next, we prove a sequential composition lemma for auxiliary-input EPRO zero-knowledge and auxiliary-input NPRO zero-knowledge, which confirms that these are in some sense indeed the "right" definitions.

Theorem 5 (sequential composition). *Let (P, V) be an efficient-prover protocol in the RO model. Let $Q(\cdot)$ be a polynomial, and let (P_Q, V_Q) be an interactive protocol that*

on common input $x \in \{0,1\}^n$, proceeds in $Q(n)$ phases, each of them consisting of executing the interactive protocol (P,V) on common input x (with independent coin tosses for P). If (P,V) is auxiliary-input EPRO *(resp.* NPRO*) zero-knowledge, then (P_Q, V_Q) is also auxiliary-input* EPRO *(resp.* NPRO*) zero-knowledge.*

Proof (sketch). We begin with the proof for EPRO zero-knowledge. The high-level structure is similar to that in [14] for establishing a sequential composition lemma for zero-knowledge proofs in the standard model. We start by partitioning the cheating verifier V_Q^* for (P_Q, V_Q) into $Q(n)$ phases, each of which is the execution of a verifier V^* for a stand-alone protocol (P,V). V^* takes as input the common input x and an auxiliary string encoding the state[10] for V_Q^* at the end of some phase i (the string also encodes i) of an interaction with P_Q, and upon interacting with P produces as output another string encoding the state for V^* at the end of phase $i+1$. The zero-knowledge property of (P,V) then guarantees a simulator S for V^*.

We generalize the earlier notation for programming a function by recursively defining $\text{RO}[\ell_1, \ldots, \ell_{i+1}]$ as $(\text{RO}[\ell_1, \ldots, \ell_i])[\ell_{i+1}]$. We may now specify the simulator for V_Q^* as follows: on input (x, z),

- Set $\tau_0 = z$
- Run the simulator S for Q phases using the simulated oracle generated in the previous phase; that is, for $i = 1, 2, \ldots, Q$,

$$S^{\text{RO}[\ell_1,\ldots,\ell_{i-1}]}(x, \tau_{i-1}) \to (\tau_i, \ell_i)$$

- Output $(\ell_1 \cup \ldots \cup \ell_Q, \tau_Q)$.[11]

We define $Q(n)+1$ hybrids, $H_0, \ldots, H_Q$. The j'th hybrid is defined as the output of the following experiment:

- Run $Z^{\text{RO}}(1^n) \to z$, $\tau_0 = z$.
- Let the honest prover interact with the cheating verifier for j phases; that is, for $i = 1, 2, \ldots, j$,

$$(P^{\text{RO}}, V^{*\text{RO}}(\tau_{i-1}))(x) \to \tau_i$$

- For the remaining $Q - j$ phases, run the simulator S using the simulated oracle generated in the previous phase; that is, for $i = j+1, \ldots, Q$,

$$S^{\text{RO}[\ell_j,\ldots,\ell_{i-1}]}(x, \tau_{i-1}) \to (\tau_i, \ell_i)$$

- H_j is $(\text{RO}[\ell_{j+1}, \ldots, \ell_Q], \tau_Q, z)$.

[10] For simplicity, we may think of the string as encoding the transcript for the first i phases of interaction with P_Q along with the random tape and auxiliary input for V_Q^*.

[11] We abuse $\cup$ slightly here; we want $\ell_1 \cup \ldots \cup \ell_Q$ to denote the set satisfying $\text{RO}[\ell_1 \cup \ldots \cup \ell_Q] = \text{RO}[\ell_1, \ldots, \ell_Q]$.

Note that H_0 and H_Q correspond to simulated transcript and the actual transcript respectively. We need to show that H_0 and H_Q are computationally indistinguishable. Suppose on the contrary that this is not the case. Therefore, we have a nonuniform oracle PPT D that distinguishes two consecutive hybrid distributions H_j and H_{j+1}. We define an auxiliary input machine Z_j that computes the interaction between P and V^* for the first j phases:

- Run $Z^{\text{RO}}(1^n) \to z, \tau_0 = z$.
- For $i = 1, 2, \ldots, j$, $(P^{\text{RO}}, V^{*\text{RO}}(\tau_{i-1}))(x) \to \tau_i$
- Output (z, τ_j).

This allows us to rewrite H_j and H_{j+1} as follows:

- Run $Z_j^{\text{RO}}(1^n) \to (z, \tau_j)$.
- $S^{\text{RO}}(x, \tau_j) \to (\tau_{j+1}, \ell_{j+1})$
- for $i = j+2, \ldots, Q$, $(S^{\text{RO}[\ell_{j+1}, \ldots, \ell_{i-1}]}(\tau_{i-1}))(x) \to (\tau_i, \ell_i)$
- Output $(\text{RO}[\ell_{j+1}, \ldots, \ell_Q], \tau_Q, z)$.

- Run $Z_j^{\text{RO}}(1^n) \to (z, \tau_j)$.
- $(P^{\text{RO}}, V^{*\text{RO}}(\tau_j))(x) \to \tau_{j+1}$
- for $i = j+2, \ldots, Q$, $(S^{\text{RO}[\ell_{j+2}, \ldots, \ell_{i-1}]}(\tau_{i-1}))(x) \to (\tau_i, \ell_i)$
- Output $(\text{RO}[\ell_{j+2}, \ldots, \ell_Q], \tau_Q, z)$.

This is sufficient to contradict zero-knowledge of (P, V) for the $j+1$'th phase.[12] The result for NPRO zero-knowledge follows as a special case corresponding to $\ell_1 = \ldots = \ell_Q = \emptyset$. □

Remark 2. One might expect a priori that the zero knowledge is preserved under a polynomial number of repetitions once we take into account oracle-dependent auxiliary information. However, we are only able to establish such a statement in the EPRO and NPRO models. Technically, the proof breaks down for FPRO zero-knowledge because the simulator is not required to specify the simulated random oracle. In particular, this shows that sequential composition is more subtle than merely accounting for auxiliary input. A natural question that arises is whether prepending a random prefix to all oracle queries allows us to transform any protocol that is FPRO zero-knowledge into one that remains zero-knowledge under a polynomial number of sequential compositions. We note that using a random prefix only guarantees "independence" of the prover's messages across different iterations; a cheating verifier is not limited to queries with the given prefix.[13]

[12] Unlike in the standard model, we cannot use an averaging argument to fix the output (z, τ_j) from Z_j. This is because the output depends on RO. We may eliminate the efficient-prover constraint in the lemma by allowing the auxiliary input machine Z in the definition of zero-knowledge to be unbounded, but we do not know how to achieve zero-knowledge without a bound on the query complexity of Z.

[13] Specifically, consider the trivial protocol for the language $\{0,1\}^*$ wherein the prover sends nothing and the (honest) verifier always accepts. Note that using a random prefix does not affect this protocol in any way. Now, consider a cheating verifier that after each iteration outputs $\text{RO}(0^n)$. The zero-knowledge simulator for a single iteration (without dependent auxiliary input) may simply output a random string, but simply concatenating the output of

7 Proofs of knowledge with dependent auxiliary input

Several constructions of zero-knowledge protocols begin with the verifier sending a proof of knowledge, for instance, that used in our counter-example, and the NPRO zero-knowledge protocol in [22]. If we allow the cheating verifier to receive an auxiliary input that depends on the random oracle, we would need to also extend the definition of proof of knowledge to incorporate auxiliary inputs that depend on the random oracle.

Definition 2. *Let (P, V) be an interactive protocol for a language $L = L_R$. We say that (P, V) is a* proof of knowledge w.r.t. dependent auxiliary input in the RO model *(or auxiliary-input proof of knowledge) if for every oracle PPT P^*, there exists an oracle PPT E such that for all nonuniform oracle PPT Z and for all x:*

$$\Pr_{\text{RO}}\left[Z^{\text{RO}}(1^{|x|}) \to z;\ E^{\text{RO}}(x, z) \to w; (x, w) \in R\right]$$
$$> \Pr_{\text{RO}}\left[Z^{\text{RO}}(1^{|x|}) \to z;\ (P^{*\text{RO}}(z), V^{\text{RO}})(x) \text{ accepts}\right] - \text{neg}(|x|)$$

The next result implies a separation between zero-knowledge proofs of knowledge and zero-knowledge auxiliary-input proofs of knowledge. Specifically, we rule out non-interactive protocols that are simultaneously zero-knowledge and a proof of knowledge (in the above sense); otherwise, we can simply apply the knowledge extractor to the simulated proof to obtain a candidate witness. Some care is needed in arguing that the knowledge extractor works on the simulated proof, which uses a simulated random oracle and not the actual one. The reason why this approach only works for the new definition of proofs of knowledge is that we allow a cheating prover to receive oracle-dependent auxiliary input. In particular, the cheating prover may receive a convincing proof as auxiliary input, and the knowledge extractor can neither rewind the auxiliary input machine nor observe the oracle queries it makes. The proof is deferred to the full version.

Theorem 6. *Assuming the existence of one-way functions, there is a 2-round public-coin argument system for* NP *that is auxiliary-input* EPRO *zero-knowledge, and also an auxiliary-input proof of knowledge. On the other hand, only languages in* BPP *have a non-interactive argument system that is* EPRO *zero-knowledge and an auxiliary-input proof of knowledge.*

---

this simulator for a polynomial number of times does not yield a correct simulation of the view of the cheating verifier for a polynomial number of iterations. This highlights the difference between simulating the transcript vs the output of the verifier, and the difficulty in ensuring "independence" of the random oracles amongst different iterations.

8 Circuit obfuscation in the random oracle model

Let $\mathcal{O}$ be a probabilistic polynomial-time algorithm and let $\mathcal{C}$ be a family of circuits. Let A, S, Z, D be oracle PPTs. Given $C \in \mathcal{C}$, we define $\mathsf{diff}^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3}_{A,S,Z,D}(C)$ to be the quantity

$$\Pr_{\text{RO}}\big[z \leftarrow Z^{\mathsf{O}_1}(1^{|C|});\ \tau \leftarrow A^{\text{RO}}(\mathcal{O}^{\text{RO}}(C));\ D^{\mathsf{O}_2}(\tau, z)) = 1\big]$$
$$-\Pr_{\text{RO}}\big[z \leftarrow Z^{\mathsf{O}_1}(1^{|C|});\ (\tau, \ell) \leftarrow S^{\text{RO},C}(z);\ D^{\mathsf{O}_3}(\tau, z) = 1\big]$$

Definition 3 (circuit obfuscation [19, 1, 16]). *A probabilistic polynomial-time algorithm $\mathcal{O}$ is an* obfuscator for the family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ in the FPRO model *(where $\mathcal{C}_n$ is the subset of circuits in $\mathcal{C}$ that take inputs of length n) if the following three conditions hold:*

- *(approximate functionality) There exists a negligible function α such that for all n, for all $C \in \mathcal{C}_n$, with probability $1 - \alpha(n)$ over the internal coin tosses of the obfuscator and over* RO*, $\mathcal{O}^{\text{RO}}(C)$ describes a circuit with* RO*-gates that computes the same function as C.*
- *(polynomial slowdown) There is a polynomial p such that for every circuit $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq p(|C|)$.*
- *(virtual black-box property) For every oracle PPT A, there exists an oracle PPT S such that for all $C \in \mathcal{C}$ and for all nonuniform oracle PPTs Z and D, $\mathsf{diff}^{\varepsilon,\varepsilon,\varepsilon}_{A,S,Z,D}(C)$ is negligible (as a function of $|C|$).*

We say that $\mathcal{C}$ is FPRO obfuscatable *if there exists an obfuscator for $\mathcal{C}$. In addition, we obtain:*

EPRO obfuscatable	*if* $\mathsf{O}_1 = \varepsilon, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}[\ell]$
NPRO obfuscatable	*if* $\mathsf{O}_1 = \varepsilon, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}$
auxiliary-input FPRO obfuscatable	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \varepsilon, \mathsf{O}_3 = \varepsilon$
auxiliary-input EPRO obfuscatable	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}[\ell]$
auxiliary-input NPRO obfuscatable	*if* $\mathsf{O}_1 = \text{RO}, \mathsf{O}_2 = \text{RO}, \mathsf{O}_3 = \text{RO}$

A point function I_w is a boolean function that evaluates to 1 on input w and 0 everywhere else. As observed in [19], to obfuscate I_w in the RO model, we may simply pick a random $\alpha \in \{0,1\}^{|w|}$ and store $\alpha, \text{RO}(\alpha \circ w)$ in the obfuscated circuit, which on input x, outputs 1 iff $\text{RO}(\alpha \circ x) = \text{RO}(\alpha \circ w)$.

Theorem 7 (obfuscating point functions [19]). *There exists an auxiliary-input* FPRO *obfuscator for the class of point functions.*

One may expect some modification of the previous construction to yield an EPRO obfuscator for point functions, but this turns out to be impossible. The next result follows from a similar characterization in [26] for NPRO obfuscation:

Theorem 8 (triviality). *A family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ is* EPRO *obfuscatable iff $\mathcal{C}$ is efficiently and exactly learnable using membership queries.*

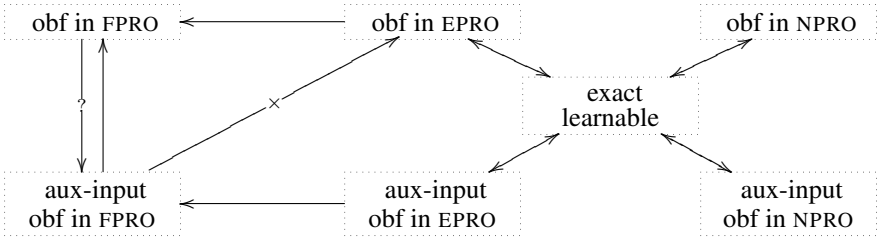

Fig. 2. Relations between different variants of obfuscation in the RO model. An arrow is an implication, a double-tailed arrow is an equivalence, and a crossed arrow indicate separation. We stress that the relations refer to families of circuits that are obfuscatable according to the respective notions.

Proof (sketch). Suppose $\mathcal{C}$ is efficiently and exactly learnable using membership queries. Consider an obfuscator that simply takes the input circuit C and outputs the circuit produced by the learning algorithm given oracle access to C; the simulator does essentially the same thing.

The learning algorithm for an EPRO obfuscatable family of circuits $\mathcal{C}$ is very simple. To evaluate $C \in \mathcal{C}$ on input x (given oracle access to C and input x), run the simulator S for the trivial adversary A that merely outputs the obfuscated circuit to obtain (τ, ℓ), answering queries to RO with random coin tosses, and then evaluate τ on the input x using the simulated oracle $\text{RO}[\ell]$. This may be modified via standard techniques (specifically, we will need to amplify the soundness error via repetition and then take a union bound over all inputs) to yield a learning algorithm that on oracle access to C output w.h.p. a (standard) circuit that agrees with C on all inputs. □

Acknowledgements. I am very grateful towards Rafael Pass for insightful exchanges on the subject and for allowing me to include his observations on deniable zero-knowledge [23], and the anonymous referees for meticulous and thought-provoking feedback. I also thank Andy Yao for hosting my visit at Tsinghua University in the 2005/2006 academic year (thereby funding this research) and for helpful discussions.

References

1. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.
2. M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT*, pages 171–188, 2004.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993.

4. M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In *EUROCRYPT*, pages 92–111, 1994.
5. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
6. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *JACM*, 51(4):557–594, 2004.
7. R. Canetti and T. Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003.
8. U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SICOMP*, 29(1):1–28, 1999.
9. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.
10. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
11. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO*, pages 152–168, 2005.
12. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
13. O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.
14. O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.
15. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–111, 2003.
16. S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.
17. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
18. D. Hofheinz and J. Müller-Quade. Universally composable commitments using random oracles. In *TCC*, pages 58–76, 2004.
19. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT*, pages 20–39, 2004.
20. A. Narayanan and V. Shmatikov. Obfuscated databases and group privacy. In *ACM CCS*, pages 102–111, 2005.
21. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002.
22. R. Pass. On deniabililty in the common reference string and random oracle models. In *CRYPTO*, pages 316–337, 2003.
23. R. Pass. private communication, 2005.
24. C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
25. D. Unruh. Random oracles and auxiliary input. In *CRYPTO*, pages 205–223, 2007.
26. H. Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005.