Cryptography Using Captcha Puzzles

Abishek Kumarasubramanian[1], Rafail Ostrovsky[1], Omkant Pandey[2], and Akshay Wadia[1]

[1] University of California, Los Angeles
abishekk@cs.ucla.edu,rafail@cs.ucla.edu,awadia@cs.ucla.edu
[2] University of Texas, Austin
omkantp@gmail.com

Abstract. A Captcha is a puzzle that is easy for humans but hard to solve for computers. A formal framework, modelling Captcha puzzles (as hard AI problems), was introduced by Ahn, Blum, Hopper, and Langford ([ABHL03], Eurocrypt 2003). Despite their attractive features and wide adoption in practice, the use of Captcha puzzles for general cryptographic applications has been limited.

In this work, we explore various ways to formally model Captcha puzzles and their human component and explore new applications for Captcha. We show that by defining Captcha with additional (strong but realistic) properties, it is possible to broaden Captcha applicability, including using it to learning a machine's "secret internal state." To facilitate this, we introduce the notion of an human-extractable Captcha, which we believe may be of independent interest. We show that this type of Captcha yields a *constant round* protocol for *fully* concurrent non-malleable zero-knowledge. To enable this we also define and construct a Captcha-based commitment scheme which admits "straight line" extraction. We also explore Captcha definitions in the setting of Universal Composability (UC). We show that there are two (incomparable) ways to model Captcha within the UC framework that lead to different results. In particular, we show that in the so called *indirect access model*, for every polynomial time functionality $\mathcal{F}$ there exists a protocol that UC-realizes $\mathcal{F}$ using human-extractable Captcha, while for the so-called *direct access model*, UC is impossible, even with the help of human-extractable Captcha.

The security of our constructions using human-extractable Captcha is proven against the (standard) class of all polynomial time adversaries. In contrast, most previous works guarantee security only against a very limited class of adversaries, called the *conservative* adversaries.

Keywords: Captcha, concurrent non-malleable zero-knowledge, universal composability, human-extractable Captcha.

1 Introduction

Captcha is an acronym for *Completely Automated Public Turing test to tell Computers and Humans Apart*. These are puzzles that are easy for humans but hard to solve for automated computer programs. They are used to confirm the "presence of a human" in a communication channel. As an illustration of a scenario where such a confirmation is very important, consider the problem of spam. To carry out their nefarious activities, spammers need to create a large number of fake email accounts. Creating a new email account usually requires the filling-in of an online form. If the spammers were to manually fill-in all these forms, then the process would be too slow, and they would not be able to generate a number of fake addresses. However, it is relatively simple to write a script (or an automated *bot*) to quickly fill-in the forms automatically without human intervention. Thus, it is crucial for the email service provider to ensure that the party filling-in the form is an actual human, and not an automated script. This is achieved by asking the party to solve a CAPTCHA, which can only be sovled by a human[3]. A common example of a CAPTCHA puzzle involves the distorted image of a word, and the party is asked to identify the word in the image.

The definition of CAPTCHA stipulates certain limitations on the power of machines, in particular, that they cannot solve CAPTCHA puzzles efficiently. This gives rise to two distinct questions which are interesting from a cryptographic point of view. Firstly, what are the underlying hard problems upon which CAPTCHA puzzles can be based? Von Ahn, Blum, Hopper and Langford [ABHL03] study this question formally, and provide constructions based on the conjectured hardness of certain Artificial Intelligence problems.

The second direction of investigation, and the one which we are concerned with in this paper, is to use CAPTCHAs as a tool for achieving general cryptographic tasks. There have been only a few examples of use of CAPTCHAs in this regard. Von Ahn, Blum, Hopper and Langford [ABHL03] use CAPTCHAs for image-based steganography. Canetti, Halevi and Steiner construct a scheme to thwart off-line dictionary attacks on encrypted data using CAPTCHAs. And recently, Dziembowski [Dzi10] constructs a "human" key agreement protocol using only CAPTCHAs. We continue this line of work in the current paper, and investigate the use of CAPTCHAs in zero-knowledge and UC secure protocols. On the face of it, it is unclear how CAPTCHAs may be used for constructing such protocols, or even for constructing building blocks for these protocols, like commitment schemes. However, motivated by current CAPTCHA theory, we define a new *extraction* property of CAPTCHAs that allows us to use them for designing these protocols.

We now give an overview of our contributions. We formally define CAPTCHAs in Section 3, but give an informal overview of the model here to make the following discussion cogent. Firstly, modelling CAPTCHA puzzles invariably involves modelling humans who are the key tenets in distinguishing CAPTCHAs from just another one-way function. Following [CHS06] we model the presence of a human entity as an oracle H that is capable of solving CAPTCHA puzzles. A party generates a CAPTCHA puzzle by running a (standard) PPT generation algorithm denoted by G. This algorithm outputs a puzzle-solution pair (z, a). All parties

[3] For many more uses of CAPTCHA, see [Off]

have access to a "human" oracle denoted by H. To "solve" a Captcha puzzle, a party simply queries its oracle with the puzzle and obtains the solution in response. This allows us to distinguish between two classes of machines. Standard PPT machines for which solving Captchas is a hard problem and oracle PPT machines with oracle access to H which may solve Captchas efficiently.

The starting point of our work is the observation that if a machine must solve a given Captcha puzzle (called *challenge*), it *must* send one or more Captcha-queries to a human. These queries are likely to be correlated to the challenge puzzle since otherwise they would be of no help in solving the challenge puzzle. Access to these queries, with the help of another human, may therefore provide us with some knowledge about the internal state of a (potentially) malicious machine! This is formulated in our definition of an human extractable Captcha (Definition 32). Informally, we make the following assumption about Captcha puzzles. Consider two randomly chosen Captcha puzzles (p_0, p_1) of which an adversary obtains only one to solve, say p_b, where the value of b is not known to the challenger. Then by merely looking at his queries to a human oracle H, and with the help of a *human*, a challenger must be able to identify the value of b. More precisely, we augment the human oracle H to possess this added ability. We then model adversaries in our protocols as oracle PPT machines with access to a Captcha solving oracle, but whose internal state can be "extracted" by another oracle PPT machine.

It is clear that this idea, i.e.—the idea of learning something non-trivial about a machine's secret by looking at its Captcha-queries—connects Captcha puzzles with main-stream questions in cryptography much more than ever. This work uses this feature present in Captchas to construct building blocks for zero-knowledge protocols which admit "straight-line" simulation. It is then natural to investigate that if we can get "straight-line" simulation, then perhaps we can answer the following questions as well: construction of *plain-text aware* encryption schemes [BR94], "straight-line" extractable commitment schemes, constant-round fully concurrent zero-knowledge for **NP** [DNS98], fully concurrent two/multi-party computation [Lin03a,PR03,Pas04], universal composition *without* trusted setup assumptions [Can01,CLOS02], and so on.

Our Contribution. In section 4 (theorem 42), as the first main result of this work, we construct a commitment scheme which admits "straight-line" extraction. That is, the committed value can be extracted by looking at the Captcha-queries made by the committer to a human oracle.

The starting point (ignoring for a moment an important difficulty) behind our commitment protocol is the following. The receiver R chooses two independent Captcha puzzles (z_0, z_1). To commit to a bit b, the sender C will select z_b using the 1-2-OT protocol and commit to its solution a_b using an ordinary (perfectly-binding) commitment scheme. Since the committer cannot solve the puzzle itself, it must query a human to obtain the solution. By looking at the puzzles C queries to the human, an extractor (with the help of another human oracle) can detect the bit being committed. Since the other puzzle z_{1-b} is computationally hidden from C, this should indeed be possible.

As alluded above, the main difficulty with this approach is that a cheating sender may not query the human on any of the two puzzles, but might still be able to commit to a correct value by obtaining solutions to some related puzzles. This is the issue of malleability that we discuss shortly, and also in section 3.

We then use this commitment scheme as a tool to obtain new results in protocol composition. First off, it is straightforward to see that given such a scheme, one can obtain a constant-round concurrent zero-knowledge protocol for all of **NP**. In fact, by using our commitment scheme in place of the "PRS-preamble" [PRS02] in the protocol of Barak, Prabhakaran, and Sahai [BPS06], we obtain a constant-round protocol for *concurrent non-malleable zero-knowledge* [BPS06] (see appendix D).[4]

As a natural extension, we investigate the issue of incorporating CAPTCHA puzzles in the UC framework introduced by Canetti [Can01]. The situation turns out to be very sensitive to the modelling of CAPTCHA puzzles in the UC framework. We discuss two different ways of incorporating CAPTCHA puzzles in the UC framework: [5]

– INDIRECT ACCESS MODEL: In this model, the environment $\mathcal{Z}$ is *not* given direct access to a human H. Instead, the environment is given access to H *only through the adversary* $\mathcal{A}$. This model was proposed in the work of Canetti et. al. [CHS06], who constructed a UC-secure protocol for password-based key-generation functionality. We call this model the *indirect* access model.
– DIRECT ACCESS MODEL: In this model, the environment is given a direct access to H. In particular, the queries made by $\mathcal{Z}$ to H are not visible to the adversary $\mathcal{A}$, in this model.

In the indirect access model, we show how to construct UC-secure protocols for all functionalities. In section 5, as the second main result of this work, we construct a constant-round *UC-puzzle* protocol as defined by Lin, Pass, and Venkitasubramaniam [LPV09]. By the results of [LPV09], UC-puzzles are sufficient to obtain UC-secure protocols for general functionalities. Our protocol for UC-puzzles is obtained by combining our commitment scheme with a "cut-and-choose" protocol and (standard) zero-knowledge proofs for **NP** [GMW86,Blu87].

In contrast, in the direct access model, it is easy to show that UC-secure computation is impossible for most functionalities. A formal statement is obtained by essentially reproducing the Canetti-Fischlin impossibility result for UC-commitments [CF01] (details reproduced in appendix E.1). The situation turns out to be the same for concurrent self-composition of two-party protocols: by reproducing the steps of Lindell's impossibility results [Lin03b,Lin08],

[4] For readers familiar with concurrent non-malleability, our protocol admits "straight-line" simulation, but the extraction of witnesses from a man-in-the-middle is not straight-line. Also, another modification is needed to the protocol of [BPS06]: we need to use a constant round non-malleable commitment scheme and not that of [DDN00]. We can use any of the schemes presented in [PR05,PPV08,LP11,Goy11].

[5] We assume basic familiarity with the model of universal composition, and briefly recall it in appendix C.1.

concurrent self-composition in this model can be shown equivalent to universal composition. This means that secure computation of (most) functionalities in the concurrent self-composition model is impossible even with CAPTCHA puzzles.

On modelling CAPTCHA *puzzles in the UC framework.* The fact that UC-computation is possible in the indirect access model but concurrent self-composition is impossible raises the question whether indirect access model is the "right" model. What does a positive result in this model mean? To understand this, let us compare the indirect access model to the other "trusted setup" models such as the Common-Random-String (CRS) model [BSMP91]. In the CRS-model, the simulator $\mathcal{S}$ is in control of generating the CRS in the ideal world—this enables $\mathcal{S}$ to have a "trapdoor" to continue its actions without having to "rewind" the environment. We can view the indirect access model as some sort of a setup (i.e., access to H) controlled by the simulator in the ideal world. The fact that $\mathcal{S}$ can see the queries made by $\mathcal{Z}$ to H in the indirect-access-model, is then analogous to $\mathcal{S}$ controlling the CRS in the CRS-model. The only difference between these two settings is that the indirect-access-model does *not* require any trusted third party. viewed this way, the indirect-access-model can be seen as a "hybrid" model that stands somewhere between a trusted setup (such as the CRS model) and the plain model.

Beyond Conservative Adversaries. An inherent difficulty when dealing with CAPTCHA puzzles, is that of *malleability.* Informally, this means that given a challenge puzzle z, it might be possible for an algorithm $\mathcal{A}$ to efficiently generate a new puzzle z' such that given the solution of z', $\mathcal{A}$ can efficiently solve z. Such a malleability attack makes it difficult to reduce the security of a cryptographic scheme to the "hardness" of solving CAPTCHA puzzles.

To overcome this, previous works [CHS06,Dzi10] only prove security against a very restricted class of adversaries called *conservative* adversaries. Such adversaries are essentially those who do not launch the 'malleability' attack: that is, they only query H on CAPTCHA instances that are provided to them by the system. In both of these works, it is possible that a PPT adversary, on input a puzzle z may produce a puzzle z' such that the solutions of z and z' are related. But both works consider only restricted adversaries which are prohibited from querying H with such a mauled puzzle z'. As noted in [CHS06,Dzi10], this an unreasonable restriction, especially knowing that CAPTCHA puzzles are in fact easily malleable.

In contrast, in this work, we prove the security of our schemes against the standard class of all probabilistic polynomial time (PPT) adversaries. The key-idea that enables us to go beyond the class of conservative adversaries is the formulation of the notion of an *human-extractable* CAPTCHA puzzle. Informally speaking, an human-extractable CAPTCHA puzzle, has the following property: suppose that a PPT algorithm $\mathcal{A}$ can solve a challenge puzzle z, and makes queries $\bar{q}$ to the human H during this process; then there is a PPT algorithm which on input the queries $\bar{q}$, can distinguish with the help of the human that $\bar{q}$ are correlated to z and not to some other randomly generated puzzle, say z''.

We discuss this notion at length in section 3, and many other issues related to formalizing CAPTCHA puzzles. This section essentially builds and improves upon previous works of [ABHL03,CHS06,Dzi10] to give a unified framework for working with CAPTCHA puzzles. We view the notion of human-extractable CAPTCHA puzzles as an important contribution to prove security beyond the class of conservative adversaries.

2 Preliminaries

In this work, to model "access to a human", we will provide some parties (modeled as interactive Turing machines–ITM) *oracle* access to a function H. An ITM M with oracle access to H is an ordinary ITM except that it has two special tapes: a write-only *query tape* and a read-only *answer tape*. When M writes a string q on its query tape, the value $H(q)$ is written on its answer tape. If q is not a valid query (i.e., not in the domain of H), a special symbol $\perp$ is written on the output tape. Such a query and answer step is counted as one step in the running time of M. We use the notation M^H to mean that M has oracle access to H. The reader is referred to [Gol01,AB09] for a detailed treatment of this notion.

Notation. The output of an oracle ITM M^H is denoted by a triplet $(\texttt{out}, \bar{q}, \bar{a})$ where $\texttt{out}, \bar{q}$, and $\bar{a}$ denote the contents of M's output tape, a vector of strings written to the query tape in the current execution, and the answer to the queries present in $\bar{q}$ respectively.

Let $k \in \mathbb{N}$ denote the security parameter, where $\mathbb{N}$ is the set of natural numbers. All parties are assumed to receive 1^k as an implicit input (even if not mentioned explicitly). When we say that an (I)TM M (perhaps with access to an oracle H) runs in polynomial time, we mean that there exists a polynomial $T(\cdot)$ such that for every input, the total number of steps taken by M are at most $T(k)$. For two strings a and b, their concatenation is denoted by $a \circ b$. The statistical distance between two distributions X, Y is denoted $\Delta(X, Y)$.

In all places, we only use standard notations (with their usual meaning) for describing algorithms, random variables, experiments, protocol transcripts and so on. We assume familiarity with standard concepts such as computational indistinguishability, negligible functions, and so on (see [Gol01]).

Statistically Secure Oblivious Transfer We now recall the notion of a statistically secure, two message oblivious transfer (OT) protocol, as defined by Halevi and Kalai [HK10].

Definition 21 (Statistically Secure Oblivious Transfer), [HK10] *Let $\ell(\cdot)$ be a polynomial and $k \in \mathbb{N}$ the security parameter. A two-message, two-party protocol $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}} \rangle$ is said to be a* statistically secure oblivious transfer *protocol for bit-strings of length $\ell(k)$ such that both the sender S_{OT} and the receiver R_{OT} are* PPT *ITMs receiving 1^k as common input; in addition, S_{OT} gets as input* *two strings* $(m_0, m_1) \in \{0,1\}^{\ell(k)} \times \{0,1\}^{\ell(k)}$ *and* R_{OT} *gets as input a choice bit* $b \in \{0,1\}$. *We require that the following conditions are satisfied:*

- Functionality: *If the sender and the receiver follow the protocol then for every* $k \in \mathbb{N}$, *every* $(m_0, m_1) \in \{0,1\}^{\ell(k)} \times \{0,1\}^{\ell(k)}$, *and every* $b \in \{0,1\}$, *the receiver outputs* m_b.
- Receiver security: *The ensembles* $\{R_{\mathrm{OT}}(1^k, 0)\}_{k\in\mathbb{N}}$ *and* $\{R_{\mathrm{OT}}(1^k, 1)\}_{k\in\mathbb{N}}$ *are computationally indistinguishable, where* $\{R_{\mathrm{OT}}(1^k, b)\}_{k\in\mathbb{N}}$ *denotes the (first and only) message sent by* R_{OT} *on input* $(1^k, b)$. *That is,*

$$\{R_{\mathrm{OT}}(1^k, 0)\}_{k\in\mathbb{N}} \overset{c}{\equiv} \{R_{\mathrm{OT}}(1^k, 1)\}_{k\in\mathbb{N}}$$

- Sender security: *There exists a negligible function* negl$(\cdot)$ *such that for every* $(m_0, m_1) \in \{0,1\}^{\ell(k)} \times \{0,1\}^{\ell(k)}$, *every first message* $\alpha \in \{0,1\}^*$ *(from an arbitrary and possibly unbounded malicious receiver), and every sufficiently large* $k \in \mathbb{N}$, *it holds that either*

$$\Delta_0(k) := \Delta(S_{\mathrm{OT}}(1^k, m_0, m_1, \alpha), S_{\mathrm{OT}}(1^k, m_0, 0^{\ell(k)}, \alpha)) \text{ or,}$$

$$\Delta_1(k) := \Delta(S_{\mathrm{OT}}(1^k, m_0, m_1, \alpha), S_{\mathrm{OT}}(1^k, 0^{\ell(k)}, m_1, \alpha))$$

is negligible, where $S_{\mathrm{OT}}(1^k, m_0, m_1, \alpha)$ *denotes the (only) response of the honest sender* S_{OT} *with input* $(1^k, m_0, m_1)$ *when the receiver's first message is* α.

Statistically secure OT can be constructed from a vareity of cryptographic assumptions. In [HK10], Halevi and Kalai construct protocols satisfying the above definition under the assumption that *verifiable smooth projective hash families with hard subset membership problem* exist (which in turn, can be constructed from a variety of standard assumptions such as the quadratic-residue problem). [HO09] show the equivalence of 2-message statistically secure oblivious transfer and lossy encryption.

3 Modeling Captcha Puzzles

As said earlier, Captcha puzzles are problem instances that are easy for "humans" but hard for computers to solve. Let us first consider the "hardness" of such puzzles for computers. To model "hardness," one approach is to consider an asymptotic formulation. That is, we envision a randomized generation algorithm G which on input a security parameter 1^k, outputs a puzzle from a (discrete and finite) set $\mathcal{P}_k$ called the *puzzle-space*. Indeed, this is the formulation that previous works [ABHL03,Dzi10,CHS06] as well as our work here follow. assume that there is a fixed polynomial $\ell(\cdot)$ such that every puzzle instance $z \in \mathcal{P}_k$ is a bit string of length at most $\ell(k)$.

Of course, not all Captcha puzzle systems satisfy such an asymptotic formulation. It is possible to have a (natural) non-asymptotic formulation to define

Captcha puzzles which takes into consideration this issue and defines hardness in terms of a "human population" [ABHL03]. However, a non-asymptotic formulation will be insufficient for cryptographic purposes. For many puzzles, typically hardness can be amplified by sequential or parallel repetition[CHS04].

Usually, Captcha puzzles have a unique and well defined solution associated with every puzzle instance. We capture this by introducing a discrete and finite set $\mathcal{S}_k$, called the *solution-space*, and a corresponding *solution function* $H_k : \mathcal{P}_k \to \mathcal{S}_k$ which maps a puzzle instance $z \in \mathcal{P}_k$ to its corresponding solution. Without loss of generality we assume that every element of $\mathcal{S}_k$ is a bit string of length k. We will require that G generates puzzles together with their solutions. This restriction is also required in previous works [ABHL03,Dzi10]. To facilitate the idea that the puzzle-generation is a completely *automated* process, G will not be given "access to a human."

With this formulation, we can view "humans" as computational devices which can "efficiently" compute the solution function H_k. Therefore, to capture "access to a human", the algorithms can simply be provided with *oracle* access to the family of solution functions $H := \{H_k\}_{k \in \mathbb{N}}$. Recall that by definition, oracle-access to H means that algorithms can only provide an input z to some function $H_{k'}$ in the family H, and then read its output $H_{k'}(z)$; if z is not in the domain $\mathcal{P}_{k'}$, the response to the query is set to a special symbol, denoted $\perp$. Every query to $H_{k'}$ will be assumed to contribute one step to the running time of the querying algorithm. The discussion so far leads to the following definition for Captcha puzzles.

Definition 31 (Captcha Puzzles) *Let $\ell(\cdot)$ be a polynomial, and $\mathcal{S} := \{\mathcal{S}_k\}_{k \in \mathbb{N}}$ and $\mathcal{P} := \{\mathcal{P}_k\}_{k \in \mathbb{N}}$ be such that $\mathcal{P}_k \subseteq \{0,1\}^{\ell(k)}$ and $\mathcal{S}_k \subseteq \{0,1\}^k$. A* Captcha *puzzle system $\mathcal{C} := (G, H)$ over $(\mathcal{P}, \mathcal{S})$ is a pair such that G is a randomized polynomial time turing machine, called the* generation algorithm*, and $H := \{H_k\}_{k \in \mathbb{N}}$ is a collection of* solution functions *such that $H_k : \mathcal{P}_k \to \mathcal{S}_k$. Algorithm G, on input a security parameter $k \in \mathbb{N}$, outputs a tuple $(z, a) \in \mathcal{P}_k \times \mathcal{S}_k$ such that $H_k(z) = a$. We require that there exists a negligible function* negl$(\cdot)$ *such that for every* PPT *algorithm $\mathcal{A}$, and every sufficiently large $k \in \mathbb{N}$, we have that:*

$$p_{\mathsf{inv}}(k) := \Pr\left[(z, a) \leftarrow G(1^k); \mathcal{A}(1^k, z) = a\right] \leq \mathrm{negl}(k)$$

where the probability is taken over the randomness of both G and $\mathcal{A}$.

Turing Machines vs Oracle Turing Machines. We emphasize that the Captcha puzzle generation algorithm G is an ordinary turing machine with no access to any oracles. Furthermore, the security of a Captcha system holds *only* against PPT adversaries $\mathcal{A}$ who are turing machines. It *does not* hold against oracle turing machines with oracle access to H. However, we use Captcha systems defined as above in protocols which guarantee security against adversaries who may even have access to the oracle H. This distinction between machines which have access to an (human) oracle and machines which don't occurs throughout the text.

The Issue of Malleability. As noted earlier, CAPTCHA puzzles are usually easily *malleable* [DDN00]. That is, given a challenge puzzle z, it might be possible for an algorithm $\mathcal{A}$ to efficiently generate a new puzzle $z' \neq z$ such that given the solution of z', $\mathcal{A}$ can efficiently solve z. It turns out that in all previous works this creates several difficulties in the security proofs. In particular, in reducing the "security" of a cryptographic scheme to the "hardness" of the CAPTCHA puzzle, it becomes unclear how to handle such an adversary.

Due to this, previous works [Dzi10,CHS06] only prove security against a very restricted class of adversaries called the *conservative* adversaries. Such adversaries are essentially those who do not query H_k on any CAPTCHA instances other than the ones that are provided to them by the system. To facilitate a proof against all PPT adversaries, we develop the notion of human-extractable CAPTCHA puzzles below.

Human-Extractable CAPTCHA *Puzzles.* The notion of human-extractable CAPTCHA puzzles stems from the intuition that if a PPT algorithm $\mathcal{A}$ can solve a random instance z produced by G, then it must make queries $\bar{q} = (q_1, q_2, \dots)$ to (functions in) H that contain sufficient information about z. More formally, suppose that z_1 and z_2 are generated by two random and independent executions of G. If $\mathcal{A}$ is given z_1 as input and it produces the correct solution, then the queries $\bar{q}$ will contain sufficient information about z_1 and no information about z_2 (since z_2 is independent of z_1 and never seen by $\mathcal{A}$). Therefore, by looking at the queries $\bar{q}$, it should be possible with the help of the human to deduce which of the two instances is solved by $\mathcal{A}$. We say that a CAPTCHA puzzle system is human-extractable if there exists a PPT algorithm Extr which, by looking at the queries $\bar{q}$, can tell with the help of the human which of the two instances was solved by $\mathcal{A}$. The formal definition follows; recall the convention that output of oracle Turing machines includes the queries $\bar{q}$ they make to H and corresponding answers $\bar{a}$ received.

Definition 32 (Human-extractable Captcha) *A* CAPTCHA *puzzle system* $\mathcal{C} := (G, H)$ *is said to be* human-extractable *if there exists an oracle* PPT *algorithm* Extr^H*, called the* extractor*, and a negligible function* $\text{negl}(\cdot)$*, such that for every oracle* PPT *algorithm* $\mathcal{A}^H$*, and every sufficiently large* $k \in \mathbb{N}$*, we have that:*

$$p_{\mathsf{fail}}(k) := \Pr\left[\begin{array}{l}(z_0, s_0) \leftarrow G(1^k); (z_1, s_1) \leftarrow G(1^k); b \overset{\$}{\leftarrow} \{0,1\}\,; \\ (s, \bar{q}, \bar{a}) \leftarrow \mathcal{A}^H(1^k, z_b); b' \leftarrow \mathsf{Extr}^H(1^k, (z_0, z_1), \bar{q}); \\ s = s_b \wedge b' \neq b\end{array}\right] \leq \text{negl}(k)$$

where the probability is taken over the randomness of $G, \mathcal{A}$*, and* Extr*.*

Observe that except with negligible probability, $s_0 \neq s_1$, since otherwise one can break the hardness of $\mathcal{C}$(definition 31).

We believe that the notion of human-extractable CAPTCHA puzzles is a very natural notion; it may be of independent interest and find applications elsewhere.

We note that while assuming the existence of human-extractable CAPTCHA puzzles may be a strong assumption, it is very different from the usual extractability assumptions in the literature such as the Knowledge-Of-Exponent (KOE) assumption [HT98,BP04]. In particular, often it might be possible to empirically test whether a given CAPTCHA system is human-extractable. For example, one approach for such a test is to just ask sufficiently many humans to correlate the queries $\bar{q}$ to one of the puzzles z_0 or z_1. If sufficiently many humans can correctly correlate $\bar{q}$ to z_b with probability noticeably better than $1/2$, one can already conclude some form of weak extraction. Such weak extractability can then be amplified by using techniques from parallel repetition. In contrast, there is no such hope for KOE assumption (and other problems with similar "non-black-box" flavor) since they are not falsifiable [Nao03].

In this work, we only concern ourselves with human-extractable CAPTCHA puzzles. Thus we drop the adjective human-extractable as convenient.

Drawbacks of Our Approach and Other Considerations. While our framework significantly improves upon previous works [Dzi10,CHS06], it still has certain drawbacks which are impossible to eliminate in an asymptotic formulation such as ours.

The first drawback is that as the value of k increases, the solution becomes larger. It is not clear if the humans can consistently answer such a long solution. Therefore, such a formulation can become unsuitable for even very small values of k. The second drawback is that the current formulation enforces strict "rules" on how a human and a Turing machine communicate via oracle access to H. This does not capture "malicious" humans who can communicate with their computers in arbitrary ways. It is not even clear how to formally define such "malicious" humans for our purpose.

Finally, definition 31 enforces the condition that $|\mathcal{S}_k|$ is super-polynomial in k. For many CAPTCHA puzzle systems in use today, $|\mathcal{S}_k|$ may be small (e.g., polynomial in k or even a constant). Such CAPTCHA puzzles are not directly usable in our setting. Observe that if $|\mathcal{S}_k|$ is small, clearly $\mathcal{A}$ can solve a given challenge puzzle with noticeable probability. Therefore, it makes sense to consider the following weaker variant in definition 31: instead of requiring $\mathrm{p}_{\mathsf{inv}}$ to be negligible, we can consider it to be a small constant ϵ. Likewise, we can also consider weakening the extractability condition by in definition 32 by requiring $\mathrm{p}_{\mathsf{fail}}$ to be only noticeably better than $1/2$.

A subtle point to observe here is that while it might be possible to *individually* amplify $\mathrm{p}_{\mathsf{inv}}$ and $\mathrm{p}_{\mathsf{fail}}$ by using parallel or sequential repetitions, it may not be possible to amplify *both at the same time.* Indeed, when $|\mathcal{S}_k|$ is small, the adversary $\mathcal{A}$ can simply ask one CAPTCHA puzzle for every solution $a \in \mathcal{S}_k$ multiple times and "hide" the challenge puzzle z_b (in some mauled form z'_b) somewhere in this large list of queries. Such a list of queries might have sufficient correlation with *both* z_0 and z_1 simply because the solutions of these both are in $\mathcal{S}_k$ and $\mathcal{A}$ has asked at least one puzzle for each solution in the whole space. In this case, even though parallel repetition may amplify $\mathrm{p}_{\mathsf{inv}}$, extraction might completely fail because the correlation corresponding to the challenge puzzle is not easy to observe in $\mathcal{A}$'s queries and answers.

As a consequence of this, our formulation essentially rules out the possibility of using such "weak" Captcha puzzles for which both $\mathrm{p}_{\mathsf{inv}}$ and $\mathrm{p}_{\mathsf{fail}}$ are not suitable. This is admittedly a strong limitation, which seems to come at the cost of proving security beyond the class of conservative adversaries.

4 A Straight-line Extractable Commitment Scheme

In this section we present a straight-line extractable commitment scheme which uses human-extractable Captcha puzzles. The hiding and binding properties of this commitment scheme rely on standard cryptographic assumptions, and the straight-line extraction property relies on the extraction property of Captcha puzzles.

We briefly recall the notion of secure commitment schemes, with emphasis on the changes from the standard definition and then define the notion of straight-line extractable commitments.

Commitment Schemes. First, we present a definition of commitment schemes augmented with Captcha puzzles. Let $\mathcal{C} := (G, H)$ be a Captcha puzzle system, and let $\mathrm{Com}_{\mathcal{C}} := \langle \mathsf{C}^H, \mathsf{R} \rangle$ be a two-party interactive protocol where (only) C has oracle access to the solution function family H[6]. We say that $\mathrm{Com}_{\mathcal{C}}$ is a commitment scheme if: both C and R are PPT (interactive) TM receiving 1^k as the common input; in addition, C receives a string $m \in \{0,1\}^k$. Further, we require C to *privately* output a *decommitment* string d, and R to *privately* output an auxiliary string aux. The transcript of the interaction is called the *commitment* string, denoted by c. During the course of the interaction, let $\bar{q}$ and $\bar{a}$ be the queries and answers obtained by C via queries to the Captcha oracle H. To denote the sampling of an honest execution of $\mathrm{Com}_{\mathcal{C}}$, we use the following notation: $(c, (d, \bar{q}, \bar{a}), \mathsf{aux}) \leftarrow \langle \mathsf{C}^H(1^k, m), \mathsf{R}(1^k) \rangle$.

Notice that $(d, \bar{q}, \bar{a})$ is the output of oracle ITM C^H as defined in section 2. For convenience, we associate a polynomial time algorithm DCom which on input (c, d, aux) either outputs a message m, or $\perp$. It is required that for all honest executions where C commits to m, DCom always outputs m. We say that $\mathrm{Com}_{\mathcal{C}}$ is an *ordinary* commitment scheme if $\bar{q}$ (and hence $\bar{a}$) is an empty string.

Furthermore, our definition of a commitment scheme allows for stateful commitments. In particular the output aux might be necessary for a successful decommitment of the committed message.

We assume that the reader is familiar with perfect/statistical binding and computational hiding properties of a commitment scheme. Informally, straight-line extraction property means that there exists an extractor $\mathsf{ComExtr}^H$ which

[6] The reason we do not provide R with access to H, is because our construction does not need it, and therefore we would like to avoid cluttering the notation. In general, however, both parties can have access to H. Also, in our adversarial model, we consider all malicious receivers to have access to the oracle H

on input the commitment string c (possibly from an interaction with a malicious committer), aux (from an honest receiver), and $\bar{q}$, outputs the committed message m (if one exists), except with negligible probability. If m is not well defined, there is no guarantee about the output of $\mathsf{ComExtr}$.

For any commitment, we use $\mathcal{M} = \mathcal{M}(c, \mathsf{aux})$ to denote a possible decommitment message defined by the commitment string c and the receiver state aux. If such a message is not well defined (say there could be multiple such messages or none at all) for a particular (c, aux), then define $\mathcal{M}(c, \mathsf{aux}) = \bot$.

Definition 41 (Straight-line Extractable Commitment) *A statistically-binding computationally-hiding commitment scheme* $\mathrm{Com}_{\mathcal{C}} := \langle \mathsf{C}^H, \mathsf{R} \rangle$ *defined over a human-extractable* CAPTCHA *puzzle system* $\mathcal{C} := (G, H)$ *is said to admit* straight-line extraction *if there exists a* PPT *algorithm* $\mathsf{ComExtr}^H$ *(the* extractor*) and a negligible function* $\mathrm{negl}(\cdot)$*, such that for every* PPT *algorithm* $\widehat{\mathsf{C}}$ *(a* malicious *committer whose input could be arbitrary), and every sufficiently large* $k \in \mathbb{N}$*, we have that:*

$$\Pr\left[\begin{matrix}(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux}) \leftarrow \langle \widehat{\mathsf{C}}^H(1^k, \cdot), \mathsf{R}(1^k) \rangle; \mathcal{M} = \mathcal{M}(c^*, aux); \\ m \leftarrow \mathsf{ComExtr}^H(1^k, \bar{q}, (c^*, \mathsf{aux})) : (\mathcal{M} \neq \bot) \wedge (m \neq \mathcal{M})\end{matrix}\right] \leq \mathrm{negl}(k)$$

where the probability is taken over the randomness of $\widehat{\mathsf{C}}, \mathsf{R}$*, and* $\mathsf{ComExtr}$.

The Commitment Protocol. At a high level, the receiver R of our protocol will choose two CAPTCHA puzzles (z^0, z^1) (along with their solutions s^0, s^1). To commit to bit b, the sender C will select z^b using the OT protocol and commit to its solution s^b using an ordinary (perfectly-binding) commitment scheme $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$. The solution to the puzzle is obtained by querying H on z^b. To decommit, first decommit to s^b which the receiver verifies; and then the receiver accepts b as the decommitted bit if the solution it received is equal to s^b. To facilitate this task, the receiver outputs an auxiliary string aux which contains (z^0, z^1, s^0, s^1). To commit to a k-bit string $m \in \{0,1\}^k$, this atomic protocol is repeated in *parallel* k-times (with some minor modifications as in Figure 4)

For convenience we assume that $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ is *non-interactive* (i.e., C sends only one message to R) for committing strings of length k^2. The decommitment string then consists of the committed messages and the randomness of C_{PB}. The formal description of our protocol appears in figure 1.

Theorem 42 *Assume that* $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ *is an ordinary, non-interactive, perfectly-binding and computationally-hiding commitment scheme,* $\mathcal{C} = (G, H)$ *is a human-extractable* CAPTCHA *puzzle system, and* $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}} \rangle$ *is a two-round statistically-secure oblivious transfer protocol. Then, protocol* $\mathrm{Com}_{\mathcal{C}} = \langle \mathsf{C}^H, \mathsf{R} \rangle$ *described in figure 1 is a 3-round perfectly-binding and computationally-hiding commitment scheme which admits straight-line extraction.*

Proof. [Sketch] Statistical binding of our scheme follows from perfect binding of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ and the fact that except with negligible probability over the randomness of G, $s_i^0 \neq s_i^1$ for every $i \in [k]$ (since otherwise $\mathrm{p}_{\mathsf{inv}}(k)$ will not be negligible).

Let k be the security parameter, $\mathcal{C} := (G, H)$ a human-extractable Captcha puzzle system, $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ a non-interactive perfectly-binding commitment scheme for strings of length k^2, and $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}} \rangle$ a two-message two-party OT protocol.

Commitment. Let $m = (m_1, \ldots, m_k) \in \{0,1\}^k$ be the message to be committed.

1. Captcha Generation: For every $i \in [k]$, R generates a pair of independent Captcha puzzles: $(z_i^0, s_i^0) \leftarrow G(1^k)$ and $(z_i^1, s_i^1) \leftarrow G(1^k)$.
2. Parallel OT: C and R perform k parallel executions of OT, where the i^{th} execution proceeds as follows. Party R acts as the OT-sender S_{OT} on input (z_i^0, z_i^1) and party C acts the OT-receiver R_{OT} on input the bit m_i. At the end of the execution, let the puzzle instances obtained by C be $z_1^{m_1}, \ldots, z_k^{m_k}$.
3. Commit to Captcha Solutions: For every $i \in [k]$, C queries H_k on $z_i^{m_i}$ to obtain the solution $s_i^{m_i}$. Let $\bar{s} := s_1^{m_1} \circ \ldots \circ s_k^{m_k}$, which is of length k^2. C commits to $\bar{s}$ using protocol $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$. Let r be the randomness used and c be the message sent by C in this step.
4. Outputs: R sets $\mathsf{aux} = \{(z_i^0, z_i^1, s_i^0, s_i^1)\}_{i=1}^k$, and C sets $d = (\bar{s}, r)$.

Decommitment. On input the commitment transcript, and strings $d = (\bar{s}, r)$ and $\mathsf{aux} = \{(z_i^0, z_i^1, s_i^0, s_i^1)\}_{i=1}^k$ do the following: parse the transcript to obtain string c from the last step, and verify that $(\bar{s}, r)$ is a valid decommitment for c. If yes, parse $\bar{s} = a_1 \circ \ldots \circ a_k$ and test that for every $i \in [k]$, there exists a *unique* bit b_i such that $a_i = s_i^{b_i}$. If any test fails, output $\perp$; otherwise output $m = (b_1, \ldots, b_k)$.

Fig. 1. Straightline Extractable Commitment Protocol $\langle \mathsf{C}^H, \mathsf{R} \rangle$

In addition, the computational-hiding of this scheme follows by a standard hybrid argument which uses the following two conditions: the *receiver security* property of $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}} \rangle$, and computational-hiding of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$. The proof of this part is standard, and omitted.

We now show that the scheme admits straight-line extraction. Suppose that string $(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux})$ represents the output of an execution of our commitment protocol; then by statistical binding of our commitment scheme, except with negligible probability there is a unique message defined by this string. In fact, this unique message is completely defined by only the strings $(c^*, \{z_i^0, z_i^1\}_{i=1}^k)$, where $\{z_i^0, z_i^1\}_{i=1}^k$ are Captcha puzzles of the honest receiver (included in aux). Recall that to refer to this message, we use the variable $\mathcal{M}$, and write $\mathcal{M}(c^*, \mathsf{aux})$ to explicitly mention a transcript.

Now suppose that for a commitment scheme, it holds that $\Pr[\mathcal{M} \neq \perp]$ is negligible, then straight-line extraction property as in Definition 41 holds trivially. Therefore, in our analysis, it suffices to analyze malicious committers who satisfy the condition that the event $\mathcal{M} \neq \perp$ happens with non-negligible probability. The formal description of our commitment-extractor, $\mathsf{ComExtr}$, follows. It uses the (extractor-)machine Extr guaranteed for the Captcha system $\mathcal{C}$ (by definition 32), to extract m bit-by-bit.

```
Algorithm $\mathsf{ComExtr}^H(1^k, \bar{q}, (c^*, \mathsf{aux}))$:
  1. Parse aux to obtain the puzzles $\{z_i^0, z_i^1\}_{i=1}^k$.
  2. For every $i \in [k]$, set $b_i \leftarrow \mathsf{Extr}^H(1^k, (z_i^0, z_i^1), \bar{q})$. If $\mathsf{Extr}^H$ outputs $\perp$, then
     set $b_i = \perp$.
  3. Output the string $b_1 \circ \ldots \circ b_k$.
```

Observe that the extraction algorithm *does not* use the solutions $\{s_i^0, s_i^1\}_{i=1}^k$ included in the string aux, and this information is redundant. Also, the output of the algorithm above, may include the $\perp$ symbols in some places.

We say that $\mathsf{ComExtr}^H(1^k, \bar{q}, (c^*, \mathsf{aux}))$ *fails at step* i if $b_i = \perp$ or if $b_i \neq \mathcal{M}_i$, where $\mathcal{M}_i$ denotes the i^{th} bit of the unique message $\mathcal{M}$ if it exists. Further, let $p_i(k)$ denote the probability that this happens. Define the following probability $p_i(k)$ over the randomness of $\widehat{\mathsf{C}}$ and R and Extr:

$$p_i(k) := \Pr\left[\begin{matrix}(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux}) \leftarrow \langle \widehat{\mathsf{C}}^H(1^k, \cdot), \mathsf{R}(1^k)\rangle; \mathcal{M} := \mathcal{M}(c^*, \mathsf{aux}); \\ b_i = \mathsf{Extr}^H(1^k, (z_i^0, z_i^1), \bar{q}) : \mathcal{M}_i \neq \perp \wedge b_i \neq \mathcal{M}_i\end{matrix}\right] \tag{4.1}$$

We remark again, that $p_i(k)$ is not affected by the actual solutions $\{s_i^0, s_i^1\}_{i=1}^k$ included in aux in the equation above. This is because $\mathcal{M}$ is completely defined by c^* and the puzzles $\{z_i^0, z_i^1\}_{i=1}^k$, and every other expression in the equation does not depend on aux.

If we prove that $p_i(k)$ is negligible for every $i \in [k]$, then by union bound, it follows that, except with negligible probability, the output of $\mathsf{ComExtr}$ is always equal to $\mathcal{M}$ (or $\mathcal{M}$ equals $\perp$). This will complete the proof of the theorem. To show $p_i(k)$ is negligible, we construct an adversary for the extraction game in Definition 32(we will call such an adversary an Captcha *extraction adversary*), and then show that $p_i(k)$ and $\mathrm{p}_{\mathsf{fail}}$ are negligibly close.

Informally, the proof follows from the fact that when the commitment adversary solves a particular Captcha puzzle, say z_i^b, and commits to its solution s_i^b, the "other" Captcha puzzle z_i^{1-b} is statistically hidden from his view due to the statistical security of the OT protocol. Thus, this commitment adversary can be converted to a Captcha extraction adversary by setting $z_b^i = \tilde{z}$, where $\tilde{z}$ is the Captcha puzzle that the Captcha extraction adversary gets to see in the extraction game (Definition 32). Due to space limitations, we defer the formal proof to Appendix A.

■

5 Constructing UC-Puzzles using Captcha

We provided a basic background in the section 1 to our results on protocol composition, and mentioned that there are two ways in which we can incorporate

Captcha puzzles in the UC-framework: the indirect access model, and the direct access model. This section is about constructing *UC puzzles* [LPV09] in the indirect access model. Recall that in the indirect access model, the environment $\mathcal{Z}$ is not given direct access to a human (or the solution function family of the Captcha system) H; instead, $\mathcal{Z}$ must access H exclusively through the adversary $\mathcal{A}$. This allows the simulator to look at the queries of $\mathcal{Z}$, which in turn allows for a positive result. Due to space constraints, we shall assume basic familiarity with the UC-framework [Can01], and directly work with the notion of UC-puzzles. A more detailed review of the UC framework, and concurrent composition, is given in appendix C.

Lin, Pass and Venkitasubramaniam [LPV09] defined the notion of a *UC puzzle*, and demonstrated that to obtain universal-composition in a particular model (e.g., the CRS model), it suffices to construct a UC puzzle in that model. We will adopt this approach, and construct a UC puzzle using Captcha. We recall the notion of a UC-puzzle with necessary details, and refer the reader to [LPV09] for an extensive exposition. Our formulation directly incorporates Captcha puzzles in the definition and hence does not refer to any setup $\mathcal{T}$; other than this semantic change, the description here is essentially identical to that of [LPV09].

The UC-puzzle is a protocol which consists of two parties—a sender S, and a receiver R, and a PPT-relation $\mathcal{R}$. Let $\mathcal{C} := (G, H)$ be a Captcha puzzle system. Only the sender will be given oracle access to H, and the resulting protocol will be denoted by $\langle S^H, R\rangle$. Informally, we want that the protocol be *sound*: no efficient receiver R^* can successfully complete an interaction with S and also obtain a "trapdoor" y such that $\mathcal{R}(\mathsf{TRANS}, y) = 1$, where TRANS is the transcript of that execution. We also require *statistical UC-simulation*: for every efficient adversary $\mathcal{A}$ participating as a sender in many executions of the protocol with multiple receivers $R_1, \ldots, R_m$, and communicating with an environment $\mathcal{Z}$ simultaneously, there exists a simulator Sim which can *statistically* simulate the view of $\mathcal{A}$ for $\mathcal{Z}$ and output trapdoors to all successfully completed puzzles at the same time.

Formally, we consider a concurrent execution of the protocol $\langle S^H, R\rangle$ for an adversary $\mathcal{A}$. In the concurrent execution, $\mathcal{A}$ exchanges messages with a puzzle-environment $\mathcal{Z}$ and participates as a sender concurrently in $m = \text{poly}(k)$ (puzzle)-protocols with honest receivers $R_1, \ldots, R_m$. At the onset of a execution, $\mathcal{Z}$ outputs a session identifier *sid* that all receivers receive as input. Thereafter, $\mathcal{Z}$ is allowed to exchange messages only with the adversary $\mathcal{A}$. In particular, for any queries to the Captcha solving oracle, $\mathcal{Z}$ cannot query H; instead, it can send its queries to $\mathcal{A}$, who in turn, can query H for $\mathcal{Z}$, and report the answer back to $\mathcal{Z}$. We compare a real and an ideal execution.

Real Execution. The real execution consists of the adversary $\mathcal{A}$, which interacts with $\mathcal{Z}$, and participates as a sender in m concurrent interactions of $\langle S^H, R\rangle$. Further, the adversary and the honest receivers have access to H which they can query and receive the solutions over secure channels. The environment $\mathcal{Z}$ does not have access to H; it can query H, by sending its queries to $\mathcal{A}$, who queries H with the query and reports the answers back to $\mathcal{Z}$. Without loss of generality, we assume that after every interaction, $\mathcal{A}$ honestly sends TRANS to $\mathcal{Z}$, where TRANS is the transcript of execution. Let $\textsc{real}^H_{\mathcal{A},\mathcal{Z}}(k)$ be the random variable that describes the output of $\mathcal{Z}$ in the real execution.

Ideal Execution. The ideal execution consists of a PPT machine (the simulator) with oracle access to H, denoted Sim^H. On input 1^k, Sim^H interacts with the environment $\mathcal{Z}$. At the end of the execution, the environment produces an output. We denote the output of $\mathcal{Z}$ in the ideal execution by the random variable $\textsc{ideal}_{\mathsf{Sim}^H,\mathcal{Z}}(k)$.

Definition 51 (UC-Puzzle, adapted from [LPV09]) *Let $\mathcal{C} := (G, H)$ be a* Captcha *puzzle system. A pair $(\langle S^H, R\rangle, \mathcal{R})$ is called UC-puzzle for a polynomial time computable relation $\mathcal{R}$ and the* Captcha *puzzle system $\mathcal{C}$, if the following conditions hold:*

- Soundness. *There exists a negligible function* negl$(\cdot)$ *such that for every* PPT *receiver $\mathcal{A}$, and every sufficiently large k, the probability that $\mathcal{A}$, after an execution with the sender S^H on common input 1^k, outputs y such that $y \in \mathcal{R}(\mathsf{TRANS})$ where TRANS is the transcript of the message exchanged in the interaction, is at most* negl(k).
- Statistical Simulation. *For every* PPT *adversary $\mathcal{A}$ participating in a* concurrent puzzle execution, *there exists an oracle* PPT *machine called the simulator, Sim^H, such that for every* PPT *environment $\mathcal{Z}$ and every sufficiently large k, the random variables $\textsc{real}^H_{\mathcal{A},\mathcal{Z}}(k)$ and $\textsc{ideal}_{\mathsf{Sim}^H,\mathcal{Z}}(k)$ are statistically close over $k \in \mathbb{N}$, and whenever Sim sends a message of the form TRANS to $\mathcal{Z}$, it outputs y in its special output tape such that $y \in \mathcal{R}(\mathsf{TRANS})$.*

Some Tools and Notation. To construct the UC-puzzle, we will use our straight-line extractable commitment scheme $\mathrm{Com}_{\mathcal{C}} := \langle \mathsf{C}^H, \mathsf{R}\rangle$ from figure 1. However, this commitment scheme is too weak for our purposes. In particular, it has the following issues: it is possible for a cheating committer to commit to an invalid string $\perp$ (simply by committing incorrect solutions to Captcha puzzles), and this event cannot be detected by the receiver. We would like to ensure that if the receiver accepts the transcript, then except with negligible probability, there be a unique and valid string fixed by the transcript of the communication. However, since the Captcha solutions cannot be verified by a PPT Turing machine, we cannot use zero-knowledge proofs right-away to guarantee this.

Therefore, we resort to using our extractable commitment scheme along with an ordinary commitment scheme, and then use simple "cut-and-choose" techniques to ensure that the two commitment schemes commit to same values. Once this is ensured, the ordinary commitment scheme will provide us with **NP**-relations which we can work with.

Formally, let $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}}\rangle$ be a non-interactive perfectly-binding and computationally-hiding commitment scheme; and let $\mathrm{Com}_{\mathcal{C}} := \langle \mathsf{C}^H, \mathsf{R}\rangle$ be our extractable-commitment scheme in figure 1 defined over an human-extractable Captcha puzzle system $\mathcal{C} := (G, H)$. Then, define the following commitment scheme:

Scheme $\widehat{\mathsf{Com}}(v)$:

To commit to a value v, the sender commits to v twice: first commit to v using the protocol $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$, then commit to v using the protocol $\langle \mathsf{C}^H, \mathsf{R} \rangle$. The Receiver accepts the commitment if both, R_{PB} and R, accept their respective commitments.

To open to a value v, the sender executes the opening phases of both C_{PB} and $\mathrm{Com}_{\mathcal{C}}$; and if both opening phases accept v as the decommitted value, the receiver of $\widehat{\mathsf{Com}}$ also accepts v as the decommitted value. That fact that $\widehat{\mathsf{Com}}$ is statistically-binding and computationally-hiding follows directly from the corresponding properties of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ and $\langle \mathsf{C}^H, \mathsf{R} \rangle$. Now define the following 3-round "cut-and-choose" protocol for committing to a string s, which is essentially taken from [Ros04,PRS02], except that it uses the scheme $\widehat{\mathsf{Com}}$:

Scheme $\mathsf{PRSCom}(s)$:

1. Sender selects $2k$ strings $\{ s_0^i \}_{i=1}^k, \{ s_1^i \}_{i=1}^k$ such that $s = s_0^i \oplus s_1^1$. Now the sender commits to string s, as well as all $2k$ strings $\{ s_0^i \}_{i=1}^k, \{ s_1^i \}_{i=1}^k$ using the special commitment scheme $\widehat{\mathsf{Com}}$.
2. Receiver sends a uniformly selected challenge $\sigma = \sigma_1 \circ \ldots \circ \sigma_k \in \{ 0, 1 \}^k$.
3. Sender opens to $s_{\sigma_i}^i$ for each $1 \leq i \leq k$ by sending the decommitment information for $\widehat{\mathsf{Com}}$. Receiver accepts the commitment phase, if these are valid openings.

To open the committed string s according to the scheme PRSCom, the sender simply opens the rest of commitment values as well, i.e., decommitment to s as well as to the remaining k unopened strings. The fact that PRSCom is actually a statistically-binding and computationally-hiding commitment scheme follows (only) from the fact that $\widehat{\mathsf{Com}}$ is a statistically-binding and computationally-hiding commitment scheme. This is a standard proof, and is omitted (see, e.g., [Ros04,PRS02]).

For convenience, we define the notion of *well-formedness* of PRSCom. Informally, we say that a PRS commitment is well formed if each pair of commitments in the first step is indeed to valid shares of s.

Definition 52 (Partial Transcripts) *Let $\langle \rho, (\rho_0^1, \rho_1^1), \ldots, (\rho_0^k, \rho_1^k) \rangle$ be the transcripts of the commit phases of the $2k+1$ executions of $\widehat{\mathsf{Com}}$ in Step 1 of a successfully completed execution of PRSCom. For each $(i, b) \in \{ 1, \ldots, k \} \times \{ 0, 1 \}$, parse ρ_b^i as a pair of transcripts (τ_b^i, θ_b^i), where τ_b^i is the transcript of the commit phase of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$, and θ_b^i is the transcript of the commit phase of $\langle \mathsf{C}^H, \mathsf{R} \rangle$, in the (i, b)-execution of $\widehat{\mathsf{Com}}$. Similarly, let $\rho = (\tau, \theta)$. Define the partial transcript to be the tuple $\langle \tau, (\tau_0^1, \tau_1^1), \ldots, (\tau_0^k, \tau_1^k) \rangle$.*

Definition 53 (Well-formed PRS) *Let $\Psi = \langle \tau, (\tau_0^1, \tau_1^1), \ldots, (\tau_0^k, \tau_1^k) \rangle$ be the partial transcript of the commitment phase of PRSCom. Then, we say Ψ is well-formed if for every $1 \leq i \leq k$: $s_0^i \oplus s_1^i = s$ where s_b^i and s denote the strings committed in τ_b^i and τ respectively and $b \in \{0, 1\}$.*

For a partial transcript $\Psi = \langle \tau, (\tau_0^1, \tau_1^1), \ldots, (\tau_0^k, \tau_1^k) \rangle$ we define the $\mathsf{string\ committed\ in}$ Ψ, s_Ψ, as follows: if Ψ is well-formed, then $s_\Psi := s$, where s is the string committed in τ. Else, $s_\Psi := \perp$.

Note that as $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}} \rangle$ is a perfectly-binding commitment scheme, given Ψ, the statement that "Ψ is well-formed" is an **NP**-statement where the witness consists of all the committed strings along with the correct openings (which, in turn, is just the randomness used by algorithm C_{PB}).

The UC-puzzle System. The construction of our UC-puzzle over an human-extractable CAPTCHA puzzle system $\mathcal{C} := (G, H)$, denoted $\big(\langle S^H, R \rangle, \mathcal{R}\big)$ appears in figure 2. The construction uses a family of one-way permutations $\{ f_k \}_{k \in \mathbb{N}}$ such that $f_k : \{0,1\}^k \to \{0,1\}^{\ell'(k)}$ where $\ell' : \mathbb{N} \to \mathbb{N}$ is a polynomial. Let $\ell_1(\cdot)$ be a polynomial such that the length of the transcripts of the UC-puzzle is $\ell_1(k)$. Then, the relation $\mathcal{R}$ associated with our puzzle system is defined over the transcripts of executions TRANS as follows:

$\mathcal{R}(\mathsf{TRANS}, s) = 1$ if and only if:

1. $\mathsf{TRANS} := ((z, f_k), \tau_2, \tau_3) \in \{0,1\}^{\ell_1(k)}$, and $s \in \{0,1\}^k$.
2. $z = f_k(s)$, where (z, f_k) represents sender's Phase-1 message
3. τ_3, representing the transcript of ZK-proof in Phase-3, is accepting.

The string s is defined to be the *trapdoor* of the accepting execution TRANS.

All parties are given k as the security parameter, and $\{ f_k \}_{k \in \mathbb{N}}$ is a family of one-way permutations such that $f_k : \{0,1\}^k \to \{0,1\}^{\ell'(k)}$.

Phase-1. Sender chooses a string $s \in \{ 0,1 \}^k$ uniformly, and sends $z = f_k(s)$ and the description of f_k.

Phase-2. Sender commits to the string s using the scheme $\mathsf{PRSCom}(s)$.

Let $\Psi := \langle \tau, (\tau_0^1, \tau_1^1), \ldots, (\tau_0^k, \tau_1^k) \rangle$ be the *partial transcript.*[a]

Phase-3. Sender proves the following **NP**-statement using a (constant round) zero-knowledge proof system for **NP**:

- $\Psi := \langle \tau, (\tau_0^1, \tau_1^1), \ldots, (\tau_0^k, \tau_1^k) \rangle$ is *well-formed*, AND
- The string committed in τ, say s', is such that $f_k(s') = z$. That is, there exist strings (s', r): $\tau = C_{\mathrm{PB}}(s'; r) \wedge f_k(s') = z$.

If the zero-knowledge proof is accepting, the Receiver accepts and halts.

[a] Recall that PRSCom is instantiated with $\widehat{\mathsf{Com}}$.

Fig. 2. UC Puzzle $\langle S^H, R \rangle$ using a CAPTCHA puzzle system $\mathcal{C} := (G, H)$

Theorem 54 *Assume that $\{f_k\}_{k \in \mathbb{N}}$ is a family of one-way permutations, and that $\mathcal{C} := (G, H)$ is an human-extractable* CAPTCHA *puzzle system. Then, the protocol $\langle S^H, R \rangle$ in figure 2 is a UC puzzle.*

Due to space constraints, we refer the reader to a complete proof of this theorem provided in appendix B. Very briefly and informally, the high-level ideas of the proof proceed as follows. First, the cut-and-choose method in PRSCom ensures that the corresponding components of two commitment schemes (included in $\widehat{\mathsf{Com}}$), are indeed equal with high probability. Then, the ZK-proof guarantees that for one of them that the XOR of the majority of pairs committed to yield a unique and well defined value, say s_Ψ and that it is equal to the desired trapdoor s, with high probability. Then, s_Ψ can be recovered from the extractable-commitment part to prove the statistical simulation. Soundness is proven using a standard hybrid argument.

6 Conclusion

Open Questions and Future Work. Our work presents a basic technique using human-extractable Captcha puzzles to enable straight-line extraction and shows how to incorporate it into the framework of protocol composition to obtain new and interesting feasibility results. However, many other important questions remain to be answered. For examples, can we obtain zero-knowledge proofs for **NP** in 3 or less rounds?[7] Can we obtain plain-text aware encryption-schemes? What about *non-interactive* non-malleable commitments *without* setup [DDN00,CIO98,CKOS01,PPV08]?

One interesting direction is to consider improving upon the recent work of Goyal, Jain, and Ostrovsky on generating a password-based session-keys in the concurrent setting [GJO10]. One of the main difficulties in [GJO10] is to get a control on the number of times the simulator rewinds any given session. They accomplish this by using the technique of precise-simulation [MP06,PPS$^+$08]. However, since we obtain straight-line simulation, it seems likely that our techniques could be used to improve the results in [GJO10]. The reason we are not able to do this is that our techniques are limited to only simulation—they do *not* yield both straight-line simulation and extraction, whereas [GJO10] needs a control over both.

Another interesting direction is to explore the design of extractable Captcha puzzles. In general, investigating the feasibility and drawbacks of the asymptotic formulation for Captcha puzzles presented here and in [ABHL03,CHS06,Dzi10] is an interesting question in its own right. We presented a discussion of these details in section 3, however they still present numerous questions for future work.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[7] By using standard techniques, e.g., coin-tossing using our commitment scheme along with Blum's protocol [Blu87], we can obtain a 5-round (concurrent) zero-knowledge protocol. But we do not know how to reduce it to 3 rounds.

[ABHL03] Luis Von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard AI problems for security. In *EUROCRYPT*, pages 294–311, 2003.

[Blu87] Manual Blum. How to prove a theorem so no one else can claim it. In *International Congress of Mathematicians*, pages 1444–1451, 1987.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, pages 273–289, 2004.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354. IEEE Computer Society, 2006.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, pages 19–40, 2001.

[CHS04] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *TCC*, pages 17–33. Springer-Verlag, 2004.

[CHS06] Ran Canetti, Shai Halevi, and Michael Steiner. Mitigating dictionary attacks on password-protected local storage. In *ADVANCES IN CRYPTOLOGY, CRYPTO*. Springer-Verlag, 2006.

[CIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *STOC*, pages 141–150, 1998.

[CKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In *EUROCRYPT*, pages 40–59, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography. *SIAM J. on Computing*, 30(2):391–437, 2000.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[Dzi10] Stefan Dziembowski. How to pair with a human. In *SCN*, pages 200–218, 2010.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In *CRYPTO*, pages 277–294, 2010.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS*, pages 174–187, 1986.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011.

[HK10] Shai Halevi and Yael Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, pages 1–36, 2010. 10.1007/s00145-010-9092-8.

[HO09] Brett Hemenway and Rafail Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:127, 2009.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO*, pages 408–423, 1998.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *In 44th FOCS*, pages 394–403, 2003.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In Moni Naor, editor, *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222. Springer Berlin / Heidelberg, 2004.

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21:200–249, 2008. 10.1007/s00145-007-9015-5.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188, 2009.

[MP06] Silvio Micali and Rafael Pass. Local zero knowledge. In *STOC*, pages 306–315, 2006.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.

[Off] The Official CAPTCHA Site. `www.captcha.net`.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, 2004.

[PPS$^+$08] Omkant Pandey, Rafael Pass, Amit Sahai, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Precise concurrent zero knowledge. In *EUROCRYPT*, pages 397–414, 2008.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *CRYPTO*, pages 57–74, 2008.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, 2003.

[PR05] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, pages 533–542, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In *TCC*, pages 191–202, 2004.

A Proof of Theorem 42

In this section we present the complete proof of security of our extractable commitment scheme.

Theorem A1 (Restated) *Assume that $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}}\rangle$ is an ordinary, non-interactive, perfectly-binding and computationally-hiding commitment scheme, $\mathcal{C} = (G, H)$ is a human-extractable* Captcha *puzzle system, and $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}}\rangle$ is a two-round statistically-secure oblivious transfer protocol. Then, protocol* $\mathrm{Com}_{\mathcal{C}} = \langle \mathsf{C}^H, \mathsf{R}\rangle$ *described in figure 1 is a 3-round perfectly-binding and computationally-hiding commitment scheme which admits straight-line extraction.*

Proof. Statistical binding of our scheme follows from perfect binding of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}}\rangle$ and the fact that except with negligible probability over the randomness of G, $s_i^0 \neq s_i^1$ for every $i \in [k]$ (since otherwise $\mathrm{p}_{\mathsf{inv}}(k)$ will not be negligible).

In addition, the computational-hiding of this scheme follows by a standard hybrid argument which uses the following two conditions: the *receiver security* property of $\langle S_{\mathrm{OT}}, R_{\mathrm{OT}}\rangle$, and computational-hiding of $\langle C_{\mathrm{PB}}, R_{\mathrm{PB}}\rangle$. The proof of this part is standard, and omitted.

We now show that the scheme admits straight-line extraction. Suppose that string $(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux})$ represents the output of an execution of our commitment protocol; then by statistical binding of our commitment scheme, except with negligible probability there is a unique message defined by this string. In fact, this unique message is completely defined by only the strings $(c^*, \{z_i^0, z_i^1\}_{i=1}^k)$, where $\{z_i^0, z_i^1\}_{i=1}^k$ are Captcha puzzles of the honest receiver (included in aux). Recall that to refer to this message, we use the variable $\mathcal{M}$, and write $\mathcal{M}(c^*, \mathsf{aux})$ to explicitly mention a transcript.

Now suppose that for a commitment scheme, it holds that $\Pr[\mathcal{M} \neq \bot]$ is negligible, then straight-line extraction property as in Definition 41 holds trivially. Therefore, in our analysis, it suffices to analyze malicious committers who satisfy the condition that the event $\mathcal{M} \neq \bot$ happens with non-negligible probability. The formal description of our commitment-extractor, $\mathsf{ComExtr}$, follows. It uses the (extractor-)machine Extr guaranteed for the Captcha system $\mathcal{C}$ (by definition 32), to extract m bit-by-bit.

```
Algorithm $\mathsf{ComExtr}^H(1^k, \bar{q}, (c^*, \mathsf{aux}))$:
    1. Parse $\mathsf{aux}$ to obtain the puzzles $\{z_i^0, z_i^1\}_{i=1}^k$.
    2. For every $i \in [k]$, set $b_i \leftarrow \mathsf{Extr}^H(1^k, (z_i^0, z_i^1), \bar{q})$. If $\mathsf{Extr}^H$ outputs $\bot$, then set $b_i = \bot$.
    3. Output the string $b_1 \circ \ldots \circ b_k$.
```

Observe that the extraction algorithm *does not* use the solutions $\{s_i^0, s_i^1\}_{i=1}^k$ included in the string aux, and this information is redundant. Also, the output of the algorithm above, may include the $\bot$ symbols in some places.

We say that $\mathsf{ComExtr}^H(1^k, \bar{q}, (c^*, \mathsf{aux}))$ *fails at step* i if $b_i = \bot$ or if $b_i \neq \mathcal{M}_i$, where $\mathcal{M}_i$ denotes the i^{th} bit of the unique message $\mathcal{M}$ if it exists. Further, let $p_i(k)$ denote the probability that this happens. Define the following probability $p_i(k)$ over the randomness of $\widehat{\mathsf{C}}$ and R and Extr:

$$p_i(k) := \Pr\left[\begin{array}{l}(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux}) \leftarrow \langle \widehat{\mathsf{C}}^H(1^k, \cdot), \mathsf{R}(1^k)\rangle; \mathcal{M} := \mathcal{M}(c^*, \mathsf{aux});\\ b_i = \mathsf{Extr}^H(1^k, (z_i^0, z_i^1), \bar{q}) : \mathcal{M}_i \neq \bot \wedge b_i \neq \mathcal{M}_i\end{array}\right] \tag{A.1}$$

We remark again, that $p_i(k)$ is not affected by the actual solutions $\{s_i^0, s_i^1\}_{i=1}^k$ included in aux in the equation above. This is because $\mathcal{M}$ is completely defined by c^* and the puzzles $\{z_i^0, z_i^1\}_{i=1}^k$, and every other expression in the equation does not depend on aux.

If we prove that $p_i(k)$ is negligible for every $i \in [k]$, then by union bound, it follows that, except with negligible probability, the output of $\mathsf{ComExtr}$ is always equal to $\mathcal{M}$ (or $\mathcal{M}$ equals $\perp$). This will complete the proof of the theorem. To show $p_i(k)$ is negligible, we construct an adversary for the extraction game in Definition 32(we will call such an adversary an CAPTCHA *extraction adversary*), and then show that $p_i(k)$ and $\mathrm{p}_{\mathsf{fail}}$ are negligibly close.

Informally, the proof follows from the fact that when the commitment adversary solves a particular CAPTCHA puzzle, say z_i^b, and commits to its solution s_i^b, the "other" CAPTCHA puzzle z_i^{1-b} is statistically hidden from his view due to the statistical security of the OT protocol. Thus, this commitment adversary can be converted to a CAPTCHA extraction adversary by setting $z_b^i = \tilde{z}$, where $\tilde{z}$ is the CAPTCHA puzzle that the CAPTCHA extraction adversary gets to see in the extraction game (Definition 32).

To prove the above statement formally, we analyze the following hybrid games.

Hybrid $\mathcal{H}_1$. In this hybrid experiment, we consider the interaction of $\widehat{\mathsf{C}}$ with the following simulator S. Simulator S interacts with $\widehat{\mathsf{C}}$ on common input 1^k exactly like an honest receiver R of our commitment protocol except that in session i, the messages corresponding to the OT protocol are forwarded to an external (honest) OT-sender S_{OT}. The OT-sender S_{OT} is given inputs (x_0, x_1) where $(x_b, s_b) \leftarrow G(1^k)$ for $b \in \{0,1\}$ by S. At the end of the experiment, denote by let $(d', \bar{q}', \bar{a}')$ denote the contents of output tape of $\widehat{\mathsf{C}}$, and c' denote the commitment string. Define the puzzles and solutions corresponding to session i as follows: $z_i^0 = x_0, z_i^1 = x_1, s_i^0 = \perp, s_i^1 = \perp$. By defining these values, we have completely defined aux' (which includes puzzles and solutions for all sessions). Define $(c', (d', \bar{q}', \bar{a}'), \mathsf{aux}')$ to be the output of $\mathcal{H}_1$.

Let $q_1(k)$ denote the probability that the event in equation (A.1) occurs, when we replace the string $(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux})$ by the output of $\mathcal{H}_1$—i.e., $(c', (d', \bar{q}', \bar{a}'), \mathsf{aux}')$.

Observe that except for the solutions (s_i^0, s_i^1), all components of the string $(c', (d', \bar{q}', \bar{a}'), \mathsf{aux}')$, are distributed identically to the corresponding components of string $(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux})$. Further, as noted earlier, the event in equation (A.1) is independent of (s_i^0, s_i^1). Therefore, $\mathsf{ComExtr}$ *fails at step i* with probability $p_i(k)$ even if we use the strings $(c', (d', \bar{q}', \bar{a}'), \mathsf{aux}')$ sampled according to Hybrid 1, in equation A.1. Therefore, we have that $q_1(k) = p_i(k)$. Note that experiment $\mathcal{H}_1$ is a polynomial time process.

Before going to the next set of hybrid experiments, we make some observations. Recall that our protocol consists of k parallel OT-executions; let us denote them by $\mathsf{OT}_1, \ldots, \mathsf{OT}_k$. Each OT-execution has exactly 2-rounds, and the message corresponding to execution OT_i, say α_i is forwarded to S_{OT}(in $\mathcal{H}_1$). To compute α_i, our simulator S, selects a random-tape r which is independent from the randomness used to sample x_0 and x_1 of experiment $\mathcal{H}_1$. Let $\mathcal{R}$ be the domain from which r is sampled.

Consider an exponential number of experiments $\mathcal{H}_r$, defined below, one for each $r \in \mathcal{R}$. We show that in each one of them, $q_r(k)$, defined analogous to $q_1(k)$, is negligible. We note that unlike "standard" hybrid arguments which use indistinguishability of consecutive hybrids, we have that $q_1(k)$ is related to $q_r(k)$ via equation A.2.

Hybrid $\mathcal{H}_r$. This experiment is identical to $\mathcal{H}_1$ except that the randomness sampled by S apart from that used to sample (x_0, x_1) of is fixed to the string r. The output of this experiment is defined exactly as in $\mathcal{H}_r$. Let $q_r(k)$ denote the probability that event in equation (A.1) occurs when we replace the string $(c^*, (d^*, \bar{q}, \bar{a}), \mathsf{aux})$ by the output of $\mathcal{H}_1$. By definition, we have that for all $k \in \mathbb{N}$:

$$q_1(k) = \sum_{r \in \mathcal{R}} q_r(k) \cdot \Pr[r \text{ is the randomness sampled by } S] \leq \max_r q_r(k) \quad \text{(A.2)}$$

We consider the hybrids $\mathcal{H}_r$ to enable us to use the statistical security property of the OT protocol. Consider any r (one that makes the value $q_r(k)$ maximum is most convenient). To measure q_r, we will make use of definition 21 and get rid of one of the puzzles (x_0, x_1). Note that, every string $r \in \mathcal{R}$ results in our simulator S sending a string α_i to S_{OT}. Since r generates α_i, and r is hardwired in $\mathcal{H}_r$, this experiment always sends the same string α_i to S_{OT}, who responds with some reply, say $\beta \leftarrow S_{\mathrm{OT}}((x_0, x_1), \alpha_i)$. By definition 21, it holds that random-variable β is statistically-close to one of the following: either $S_{\mathrm{OT}}((x_0, 0^{\ell(k)}), \alpha_i)$ or $S_{\mathrm{OT}}((0^{\ell(k)}, x_1), \alpha_i)$. Let $\gamma(r)$ be the bit that indicates which of these two cases is true (with ties broken arbitrarily). Note that the value of $\gamma(r)$ for every $r \in \mathcal{R}$, is fixed and can be given to a non-uniform machine as an advice.

Hybrid $\mathcal{H}_r^1$. This hybrid is identical to $\mathcal{H}_r$, except that it non-uniformly receives the value of the bit $\gamma(r)$. The output of this experiment is therefore distributed identically to that of $\mathcal{H}_r$.

Hybrid $\mathcal{H}_r^2$. This hybrid is identical to $\mathcal{H}_r$ except for the following difference: instead of computing $\beta \leftarrow S_{\mathrm{OT}}((x_0, x_1), \alpha_i)$ it computes the output of S_{OT} on either $(x_0, 0^{\ell(k)})$ or $(0^{\ell(k)}, x_1)$ depending upon the value of $\gamma(r)$. If $\gamma(r) = 0$, S_{OT} is given $((x_0, 0^{\ell(k)}), \alpha_i)$ as input; else, if $\gamma(r) = 1$, it is given $((0^{\ell(k)}, x_1), \alpha_i)$.

By construction, and by definition 21, it holds that the output of this distribution is statistically-close to that of $\mathcal{H}_r^1$, with distance $\Delta_{\gamma(r)}(k)$. Therefore, if $q'_r(k)$ denotes the probability that the event in equation (A.1) occurs, when we replace the string $(c^*, (d^*, \bar{q}, \bar{a})$ by the output of $\mathcal{H}_r^2$, then

$$q'_r(k) \leq q_r(k) + \Delta_{\gamma(r)}(k) \quad \text{(A.3)}$$

Next, we show that q'_r is at most $\mathrm{p}_{\mathsf{fail}}(k)$ as defined in definition 32. To see this, consider the following adversary for breaking the extractability property of the Captcha system $\mathcal{C}$.

Adversary $A^H_{\mathcal{C}}$. The adversary incorporates the entire hybrid experiment $\mathcal{H}^2_r$, including the values of r and $\gamma(r)$. The adversary receives a challenge puzzle z as input. The execution of the adversary, on input z, internally simulates $\mathcal{H}^2_r$ (including the honest sender S_{OT}), except for the following difference: if $\gamma(r) = 0$, it sets $x_0 = z$, and otherwise it sets $x_1 = z$. Note that $A_{\mathcal{C}}$ must use its oracle H to correctly simulate $\mathcal{H}^2_r$ since the hybrid includes a cheating committer $\widehat{\mathsf{C}}$ who requires access to H.

When the execution of H^r_2 halts, let the contents of the tapes of the cheating committer $\widehat{\mathsf{C}}$, corresponding to the i^{th}-execution be $(d^*_i, \bar{q}, \bar{a})$. The decommitment information d^*_i includes a value s^*_i, supposedly a solution for $x_{\gamma(r)} = z$. The adversary outputs s^*_i as its solution to the challenge z.

First off, note that if $\mathcal{M} \neq \perp$, it holds that except with negligible probability s^*_i is a unique and well-defined value, which is indeed the solution of z. Next, observe that the internal execution of $A_{\mathcal{C}}$, on input z sampled honestly using G, is in fact identical to that of $\mathcal{H}^2_r$. Finally, observe that the queries made by A, are independent of the "other" CAPTCHA puzzle. That is, if $\gamma(r) = 0$, the queries made by A are independent of x_1 or any other puzzle. The case for $\gamma(r) = 1$ is symmetric.

Therefore, if the event in equation (A.1) occurs on outputs of $\mathcal{H}^r_2$ with probability $q'_r(k)$, then it must happen with same probability in internal execution of A on a random challenge puzzle. But occurrence of this event is equivalent to the failure of the CAPTCHA extractor as defined in definition 32. Therefore, it holds that $q'_r(k) \leq \mathrm{p}_{\mathsf{fail}}(k)$. By combining the results from all the hybrids in the sequence, we conclude that $p_i(k)$ is at most negligible. This completes the proof straight-line extraction. ■

B Proof of UC Puzzle Construction

We now prove that $(\langle S^H, R\rangle, \mathcal{R})$, as defined in Section 5, is a UC puzzle, and begin with the soundness property. Recall that the soundness property requires that no malicious receiver can output the trapdoor s. In our case, the trapdoor is the pre-image (under the one-way permutation family $\{ f_k \}_{k\in\mathbb{N}}$) of the first message of the protocol, z. We will show that if there exists a malicious receiver that succeeds in outputting the trapdoor with non-negligible probability, then there exists an adversary that inverts the one-way permutation with some non-negligible probability. Informally, it will follow from the hiding property of PRSCom, and the zero-knowledge property of the proof in Phase 3, that the receiver does not learn s from Phases 2 and 3 of the protocol. Thus, if it succesfully outputs the trapdoor, it must do so by inverting the one-way permutation. As a technical point, note that when we are constructing the adversary for $f_k(\cdot)$, we are no longer inside the UC framework, and in particular, we are allowed to rewind the adversary. Details follow.

Lemma B1 *For any malicious receiver* R^{*H}*, the probability that it outputs* s *such that* $\mathcal{R}(\mathsf{TRANS}, s) = 1$*, where* TRANS *is an accepting transcript, is negligible in* n*.*

Proof. Let R^{*H} be a malicious receiver that outputs the correct pre-image for the first message of an accepting execution with probability ϵ. We construct an adversary $\mathcal{A}$ that inverts $f_k(\cdot)$ with probability negligibly close to ϵ. We construct $\mathcal{A}$ through a series of hybrid adversaries, wherein we maintain the invariant that each intermediate adversary outputs the pre-image with probability negligibly close to ϵ. As observed before, we are no longer in the UC framwork and are thus allowed to rewind the adversary.

Hybrid $\mathcal{A}_0$: Adversary $\mathcal{A}_0$ starts an internal execution of $\langle S^H, R^{*H}\rangle$ along with the environment. In particular, $\mathcal{A}_0$ starts by setting the random tapes of the environment and R^{*H}, and starts simulating the execution of $\langle S^H, R^{*H}\rangle$, conveying messages between the various parties. The adversary $\mathcal{A}_0$ simulates an honest S^H itself. In the end, $\mathcal{A}_0$ outputs whatever R^{*H} outputs. It is clear that this internal simulation is identical to the real execution, and $\mathcal{A}_0$ outputs the correct pre-image of the first message with probability ϵ.

Hybrid $\mathcal{A}_1$: Adversary $\mathcal{A}_1$ is the same as $\mathcal{A}_0$ except that in Phase 3, instead of giving the zero-knowledge proof from S^H to R^{*H}, it uses the simulator. More precisely, let $\mathsf{Sim}_{\mathsf{ZK}}$ be the simulator for the zero-knowledge proof in Phaes 3. Adversary $\mathcal{A}_1$ runs the execution of the puzzle similar to $\mathcal{A}_0$, except that it conveys all Phase 3 messages from R^{*H} to $\mathsf{Sim}_{\mathsf{ZK}}$, and all $\mathsf{Sim}_{\mathsf{ZK}}$ messages back to R^{*H}. If $\mathsf{Sim}_{\mathsf{ZK}}$ needs to rewind the verifier, then $\mathcal{A}_1$ rewinds the entire execution, including the environment and the honest sender. This is simply done be restarting the entire execution with same random tapes for the environment and R^{*H}, and using same messages from the sender till the point of rewind.

We claim that $\mathcal{A}_1$ outputs the trapdoor with probability negligibly close to ϵ: consider the following stand-alone malicious verifier $\bar{V}^H$ for the zero-knowledge proof used in Phase 3 - verifier $\bar{V}^H$ has access to S^H, R^{*H} and the environment, and starts an execution as in hybrid $\mathcal{A}_0$. The verifier $\bar{V}^H$ relays the the Phase 3 messages from the R^{*H} to the external prover, and sends the responses of the external prover back to R^{*H}. Finally, $\bar{V}^H$ outputs whatever R^{*H} outputs.

Note that the output of $\mathcal{A}_0$ is identical to the output of an interaction of $\bar{V}^H$ with a real ZK prover, while the output of $\mathcal{A}_1$ is identical to the output of a simulation. If the probabilities of outputting the trapdoor by $\mathcal{A}_0$ and $\mathcal{A}_1$ differ noticeably, then we can distinguish between the real interaction and a simulation of the ZK protocol, which contradicts the zero-knowledge property.

Hybrid $\mathcal{A}_2$: Adversary $\mathcal{A}_2$ is the same as $\mathcal{A}_1$ except in Phase 2, instead of committing to s, it commits to all-zeros string. The rest of the execution is the same.

we claim that the probabilities that $\mathcal{A}_1$ and $\mathcal{A}_2$ output the trapdoor are negligibly close. Consider the following malicious receiver $\bar{R}^H$ for PRSCom, that interacts with the external challenger. The challenger either commits to the all-zeros string, or to a random string, and $\bar{R}^H$ tries to distinguish between the two cases. Receiver $\bar{R}^H$ has access to S^H, R^{*H} and the environment, and starts an execution as in $\mathcal{A}_1$. It runs the PRSCom protocol with the challenger on one side, and conveys the messages to R^{*H} on the other side. In the end, it outputs whatever R^{*H} outuputs.

Observe that the output of $\mathcal{A}_1$ is identical to the output of the above game when the challenger commits to a random string, while the output of $\mathcal{A}_2$ is identical to the output of the above game when the challenger commits to the all-zeros string. Thus, if the probability of outputting the trapdoor differ noticeably, then we contradict the hiding property of PRSCom.

Our final adversary $\mathcal{A}$ receives $f_k(s)$ from the external challenger, for a randomly chosen s. It runs the same execution as $\mathcal{A}_2$, except that in Step 1, it sends $f_k(s)$ to R^{*H}. This execution is identical to the execution of $\mathcal{A}_2$, and thus $\mathcal{A}$ outputs the correct pre-image with probability negligibly close to ϵ. It follows from the hardness of $f_k(\cdot)$ that ϵ is negligible in n. ■

Now we consider the simulation and extraction property. It is easy to statistically simulate a malicious sender's view, as the simulator only has to play the honest receivers' parts. For extraction, we can extract from the ExtCom part of $\widehat{\mathsf{Com}}$.

Lemma B2 *Let S^{*H} be a malicious sender. Then there exists a simulator Sim^H that statistically simulates the view of S^{*H}. Further, the probability that S^{*H} sends an accepting transcript TRANS to the environment* and *Sim^H does not output the trapdoor s such that $\mathcal{R}(\mathsf{TRANS}, s) = 1$, is negligible in n.*

Proof. The simulator Sim^H, on input 1^k, starts an internal execution of the adversary S^{*H} on input 1^k. All messages that S^{*H} sends to receivers are handled by Sim^H by emulating honest receiver strategies. All communicatin between S^{*H} and the environment is relayed back and forth by Sim^H. As the simulator plays the honest receiver strategy, $\textsc{ideal}_{\mathsf{Sim}^H,\mathcal{Z}}(k)$ and $\textsc{real}^H_{\mathcal{A},\mathcal{Z}}(k)$ are identical.

We now show how Sim^H extracts the trapdoor. For each puzzle execution j, for each $(i, b) \in [k] \times \{0, 1\}$, let $t^j_{i,b} := (\bar{q}, c, \mathsf{aux})^j_{i,b}$ be the queries, transcript and auxilliary information corresponding to the $(i, b)^{th}$ execution of of ExtCom in the j^{th} puzzle execution. Once a puzzle execution terminates, the simulator runs procedure $\mathsf{TrapExtr}$ (described below) on input $\{t^j_{i,b}\}_{(i,b)\in[k]\times\{0,1\}}$. This procedure returns a string s^j, which Sim^H outputs as the trapdoor for the j^{th} puzzle execution. We call $\{t^j_{i,b}\}_{(i,b)\in[k]\times\{0,1\}}$ the extraction transcript of the puzzle execution.

```
ALGORITHM TrapExtr($\{t_{i,b}\}_{(i,b)\in[k]\times\{0,1\}}$)

  1. For each $(i, b) \in [k] \times \{0, 1\}$, obtain $s^i_b \leftarrow \mathsf{ComExtr}(1^k, t_{i,b})$.
  2. For each $i \in [k]$, set $s^i = s^i_0 \oplus s^i_1$. Let $s$ be the most frequent string in
     the sequence $(s^1, \ldots, s^k)$. Output $s$.
```

We prove that $\mathsf{TrapExtr}$ returns the correct trapdoor with overwhelming probability. First, observe that soundness of the zero-knowledge proof in Phase 3 of the UC Puzzle immediately implies the following lemma,

Proposition B3 *The probability (over that random coins of the receiver) that the zero-knowledge proof in Phase 3 of puzzle execution is accepting and the partial transcript of that execution, Ψ, is not well-formed, is negligible in k.*

Next, we show that whenever that partial transcript is well-formed, the string returned by $\mathsf{TrapExtr}$ is the same as the string commited in partial transcript Ψ, with overwhelming probability.

Proposition B4 *Let Ψ be the partial transcript of a puzzle execution, and let s_Ψ be the string committed in Ψ. Further, let s be the string returned by* $\mathsf{TrapExtr}$ *when run on the extraction transcript of the puzzle execution. Then conditioned on the event that Ψ is well formed, the probability (over receiver's random coins) that $s_\Psi \neq s$ is negligible in k.*

Proof. For a particular execution of $\widehat{\mathsf{Com}}$, we say that the commitment is *invalid* if the strings committed in both the commit phases (that is, C_{PB} and ExtCom) are not the same. Note that if the number of invalid commitments is $\omega(\mathsf{log}(k))$, then with probability negligibly close to 1, the reciever will reject. Thus, given that Ψ is well-formed, more than half of $\widehat{\mathsf{Com}}$ exeuctions are valid. Therefore, the probability that $s_\Psi \neq s$ is negligible. ■

Finally we observe that it follows from the soundness of the zero-knowledge proof in Phase 3 of the puzzle that with all but negligible probability, $f_k(s_\Psi) = z$. The lemma follows from combining this with Propositions B3 and B4.

■

C Brief Review of Protocol Composition

Protocol composition is a general term to describe how the security of various cryptographic protocols behave when they execute in a complex environment in which many other types of protocols are running at the same time. Roughly speaking, there are mainly three types of protocol compositions considered in the literature: self-composition, general-composition, and universal-composition. Barring some technical conditions, a sequence of results in the literature shows that for most "interesting" functionalities (except Zero Knowledge), all three notions are essentially (equivalent and) impossible to achieve [CF01,Lin03a,Lin03b,Lin04].

In section 5, we constructed UC puzzle. Below, we provide a brief review of UC framework, and how to incorporate the Captcha systems in this framework. This is followed by a brief discussion about the modeling, and concurrent self-composition. For a detailed exposition of these topics we refer the reader to the works of Canetti [Can01] and Lindell [Lin03b].

C.1 Universal Composition

The framework for universal composition considers the execution of a protocol π in a complex environment by introducing a special entity, called the *environment* $\mathcal{Z}$.

The environment drives the whole execution. The execution of π with the environment $\mathcal{Z}$, an adversary A, and a trusted party $\mathcal{G}$ proceeds as follows. To start an execution of π, $\mathcal{Z}$ initiates a *protocol execution session*, identified by session identifier *sid*, and activates all parties and assigns a unique identifier to each of them at invocation. An honest party, upon activation, starts executing π on inputs provided by $\mathcal{Z}$; adversarially controlled parties may deviate from the protocol arbitrarily. During the execution, $\mathcal{Z}$ is allowed to interact with $\mathcal{A}$ arbitrarily; in addition, it can see all the outputs of honest parties. We assume asynchronous authenticated communication over point-to-point channels; the scheduling of all messages is controlled by the adversary/environment. Some protocol executions may involve calls to "trusted parties" $\mathcal{G}$, who compute a specific functionality for the parties. Let k be the security parameter. We consider two types of executions.

Ideal Execution. Let $\mathcal{F}$ be a functionality (i.e., a trusted party); and let π_{ideal} be the "ideal protocol" which instructs its parties to call $\mathcal{F}$ with their private inputs. At the end of the computation, the parties then receive the output of the computation from $\mathcal{F}$. The *ideal model execution* of functionality $\mathcal{F}$ is then execution of protocol π_{ideal} with environment $\mathcal{Z}$, adversary $\mathcal{A}$, and trusted party $\mathcal{F}$. At the end of the execution, $\mathcal{Z}$ outputs a bit, denoted by the random variable $\textsc{ideal}^{\mathcal{F}}_{\pi_{\mathsf{ideal}},\mathcal{A},\mathcal{Z}}(k)$.

Real Execution. Let π be a multiparty protocol implementing $\mathcal{F}$. The *real model execution* of π, is the execution of π with $\mathcal{Z}$, and $\mathcal{A}$. Note that there are no calls to $\mathcal{F}$ in this execution. At the end of the execution, $\mathcal{Z}$ outputs a bit, denoted by the random variable $\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}(k)$.

Informally speaking, we say that π UC-realizes $\mathcal{F}$, if π is a *secure emulation* of the protocol π_{ideal}. This is formulated by saying that for every adversary $\mathcal{A}$ participating in the real model execution of π, there exists an adversary $\mathcal{A}$', called the simulator, which participates in the ideal model execution of $\mathcal{F}$ such that no environment $\mathcal{Z}$ can tell apart whether it is interacting with $\mathcal{A}$ or $\mathcal{A}$'. That is, variables $\textsc{ideal}^{\mathcal{F}}_{\pi_{\mathsf{ideal}},\mathcal{A}',\mathcal{Z}}(k)$ and $\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}(k)$ are computationally indistinguishable.

Modeling Access to H. Let $\mathcal{C} := (G, H)$ be an extractable Captcha puzzle system. To incorporate the use of $\mathcal{C}$ in the UC framework, we provide all entities—i.e., the honest parties, the adversary $\mathcal{A}$, and the environment $\mathcal{Z}$—access to the solution function H.[8] Note that providing access to H in the UC framework is not a new formulation; it has been considered before by Canetti, Halevi, and

[8] In analogy with the "trusted set up" models such as the CRS model, we assume that all parties are using the same Captcha puzzle system $\mathcal{C}$. However, we insist that this is *not* essential to obtain our results. If there are multiple types of Captcha puzzles $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathrm{poly}(k)}$ in use, then all we really need is that the simulator can access the queries made by cheating parties to the corresponding solution functions, say $H_1, \ldots, H_{\mathrm{poly}(k)}$.

Steiner [CHS06], who construct a UC-secure password-based key-generation protocol in this model. We follow the same approach and allow the honest parties and the adversary $\mathcal{A}$ to directly access the solution function H.

However, we observe that there are two *different* ways in which we the environment $\mathcal{Z}$ can access H. This is a crucial point and the difference between what is possible and what is not.

- Indirect Access: The work of Canetti et. al. [CHS06], does *not* provide $\mathcal{Z}$ direct access to H. Instead, in their framework, all queries of $\mathcal{Z}$ to H are first sent to $\mathcal{A}$, who queries H and obtains the answers. These answers are then forwarded to $\mathcal{Z}$. We call this, the *indirect access model.*
- Direct Access: In the *direct access model*, $\mathcal{Z}$ is given direct access to H. In particular, $\mathcal{A}$ cannot see the queries sent by $\mathcal{Z}$ to H.

In section 5, we prove that in the indirect access model, under the assumption that extractable Captcha puzzles and one-way permutations exist, for *every* ppt functionality $\mathcal{F}$, there exists a protocol π that UC-realizes $\mathcal{F}$.

On the other hand, in the direct access model, clearly there is no advantage that a simulator (acting in the ideal world for adversary $\mathcal{A}$) will have compared to the classical UC-framework. This is because since $\mathcal{A}$ cannot see the queries of $\mathcal{Z}$, the simulator will also not be able to do so. Therefore, existing impossibility results for the UC-framework should also hold in the direct-access model. This is indeed quite trivial to show—for example by reproducing the proof of Canetti-Fischlin [CF01] for commitment schemes (see Appendix E.1).

Discussion. An interesting question to consider is which of these two models is the "right" model. Let us first compare the indirect access model to the other "trusted setup" models such as the CRS-model. In the CRS-model, the simulator S is in control of generating the CRS in the ideal world—this enables S to have a "trapdoor" to continue its actions without having to "rewind" the environment. All parties, *including* the environment $\mathcal{Z}$ use the (same) CRS generated S. Viewed this way, the indirect access model can be seen as some sort of a setup (i.e., the oracle H) where all parties, including $\mathcal{Z}$, use the same setup. However, the indirect access model is better than the CRS model (or other "trusted setup" models) in the sense that there is *no trust involved.* That is, in the indirect access model, there is no party who is trusted to generate the setup according to some specific settings, e.g., a random string in case of the CRS-model.

However, since the environment *must* access H through the adversary (enforcing, in some sense, the same setup condition), the indirect access model does not retain the true spirit of the *plain* or the vanilla model (where there is no setup to begin with). Intuitively, the Captcha model does not have any setup since every party is going to have its own "human" helping it to solve the Captcha puzzles: e,.g., our commitment scheme in section 4, is a scheme in the *plain* model. However, intuitively, in the plain model (irrespective of access no H), UC-security should ideally imply self-composition. The fact that the positive results in the indirect access model, do *not* carry over to the setting of self-composition, show that the direct access model is more natural and retains the true spirit of the plain model.

C.2 Concurrent Self Composition

Concurrent self composition, refers to the situation where many instances of a *single* protocol π are executed *concurrently* many times on the network. The concurrent attack model has a specific meaning in which the adversary is allowed to control the schedule and delivery of various protocol messages. The adversary can corrupt parties participating in various execution of π, either adaptively (i.e., in the middle of the execution) or statically (before any of the protocol executions begin).

In a series of results, Lindell [Lin03a,Lin03b,Lin04] proves a general theorem which, informally speaking, shows that for the so called "bi-directional bit-transmitting functionalities", security in the concurrent self-composition model implies security in the universal-composition model. It is not hard to see that his proof in fact holds in our setting (where access to H is granted to all parties) as well with respect to the *direct access model* (i.e. where environment accesses H directly and not through the adversary). Therefore, we obtain similar impossibility results for concurrent self-composition.

While the class of bi-directional bit-transmitting functionalities includes almost all interesting functionalities, zero-knowledge functionality does not fall in this class. Indeed, it is possible to have concurrent self-composition for zero-knowledge. By modifying Blum-Hamiltonicity protocol so that verifier's challenge is decided by using a coin-tossing phase (in which verifier first commits to its challenge using our extractable commitment scheme from figure 1), we can obtain constant-round and straight-line concurrent zero-knowledge for **NP**. Likewise, by replacing the initial PRS-phase by our extractable commitment scheme, the protocol of Barak, Prabhakaran, and Sahai [BPS06] yields a concurrent non-malleable zero-knowledge protocol which is constant-round with straight-line simulation. For completeness, the resulting protocol is given in appendix D.

One might wonder, if we can get straight-line simulation in concurrent NMZK, why are we not able to obtain UC-Zero-knowledge in the direct access model. The reason is that our protocol (in appendix D) can only guarantee straight-line simulation, *but not straight-line extraction* of the witness from man-in-the-middle. Also, there cannot be any method to convert this protocol so that one gets *both* simulation and extraction in straight-line since such a construction will imply UC-ZK which in turn will imply UC-security for computing all PPT functionalities in the direct access model [CLOS02], which is impossible.

D Concurrent NMZK: Constant Round, Straight-line

We assume basic familiarity with zero-knowledge protocols and their execution in a concurrent non-malleable experiment. Briefly, in such an experiment a man-in-the-middle A interacts with many provers $P_1, \ldots, P_m$ on "left" side, and with many verifiers $V_1, \dots, V_m$ or "right" side. Interaction of A with P_i is called the i^{th}-left-session; likewise, A's interaction with V_i is called i^{th}-right-session. The statements being proven to A by the provers are chosen before the execution; the statements on right that A proves to the verifiers are chosen adaptively by A as the interaction proceeds. A controls the scheduling and delivery of all messages in this experiment. Without loss of generality, we can assume that A is a deterministic polynomial time machine with $z \in \{0,1\}^*$ as its auxiliary input. Concurrent Non-Malleable Zero-Knowledge is defined as follows ([BPS06,PR05]):

Definition D1 (Concurrent Non-Malleable Zero-Knowledge) *A protocol is a Concurrent Non-Malleable Zero Knowledge (CNMZK) argument of knowledge for membership in an NP language L with witness relation R (that is, $y \in L$ iff there exists w such that $R(y,w) = 1$), if it is an interactive proof system between a prover and a verifier such that*

Completeness: *if both the prover and the verifier are honest, then for every (y, w) such that $R(y,w) = 1$, the verifier will accept the proof, and*

Soundness, Zero-Knowledge and Non-Malleability: *for every (non-uniform PPT) adversary A interacting with provers $P_1, \dots, P_{m_L}$ in m_L "left sessions" and verifiers $V_1, \dots, V_{m_R}$ in m_R "right sessions" of the protocol (with A controlling the scheduling of all the sessions), there exists a simulator S such that for every set of "left inputs" $y_1, \dots, y_{m_L}$, we have $S(y_1, \dots, y_{m_L}) = (\nu, z_1, \dots, z_{m_R})$, such that*

1. *ν is a simulated view of A: i.e., ν is distributed indistinguishably from the view of A (for any set of witnesses $(w_1, \dots, w_{m_L})$ that $P_1, \dots, P_{m_L}$ are provided with).*
2. *For all $i \in \{1, \dots, m_R\}$, if in the i^{th} right hand side session in ν the common input is x_i and the verifier V_i accepts the proof, then z_i is a valid witness to the membership of x_i in the language, except with negligible probability ($z_i = \perp$ if V_i does not accept.)*

Further, we call the protocol a black-box CNMZK if there exists a universal simulator S_{BB} such that for any adversary A, it is the case that $S = S_{BB}^A$ satisfies the above requirements.

To obtain concurrent non-malleable zero-knowledge (CNMZK), we use the protocol of Barak-Prabhakaran-Sahai [BPS06]. Briefly, the protocol has five phases, of which the first phase is the the PRS-preamble [PRS02]. We obtain our protocol by using our extractable commitment scheme from section 4 in phase I, instead of the PRS preamble. The protocol uses following ingredients, all of which can be constructed from standard number theoretic assumptions (or even general assumptions such as Claw Free permutations [GK96]): a two-round statistically-hiding commitment scheme denoted $\mathsf{Com}_{\mathsf{SH}}$ [GK96], a (constant round) statistical zero-knowledge argument-of-knowledge for **NP**, denoted SZKAOK [Blu87,GK96], and a (constant round) non-malleable commitment scheme denoted $\mathsf{Com}_{\mathsf{NM}}$ [Goy11,LP11,PR05,PPV08]. The resulting protocol is depicted in figure 3.

Common Input: $x \in L$.
Prover's Auxiliary Input: $y \in R_L(x)$.

Phase I: $(V \leftrightarrow P)$ The verifier chooses a random string ρ and commits to it using the commitment scheme $\mathrm{Com}_{\mathcal{C}} := \langle \mathsf{C}^H, \mathsf{R} \rangle$ from section 4.
Phase II: $(P \leftrightarrow V)$ P commits to the all-zero string using $\mathsf{Com}_{\mathsf{SH}}$. Then it uses SZKAOK to prove knowledge of the randomness and inputs to this execution of $\mathsf{Com}_{\mathsf{SH}}$.

Phase III: $(V \leftrightarrow P)$ Execute the Opening Phase of $\mathrm{Com}_{\mathcal{C}}$. Let the committed string (as revealed by the verifier) be σ.
Phase IV: $(P \leftrightarrow V)$ P commits to the witness y using $\mathsf{Com}_{\mathsf{NM}}$.
Phase V: $(P \leftrightarrow V)$ P proves the following statement using SZKAOK: either the value committed to in Phase IV is y such that $y \in R_L(x)$, or the value committed to in Phase II is σ. P uses the witness corresponding to the first part of the statement.

Fig. 3. Straight-Line Concurrent Non-Malleable Zero-Knowledge $(\mathbf{P}, \mathbf{V})$.

To prove the security of their protocol BPS only require the following two properties form the PRS-preamble: computational-hiding of the PRS-challenge, and extraction of each session-wise PRS-challenge (by means of rewinding) as soon as the PRS-phase ends. Since both of these properties are also satisfied by our extractable commitment scheme, we do not need to change the proof of BPS. In addition, since there are no rewindings involved during simulation (to extract the PRS-challenge), the proof in fact gets simpler. Note that the extraction of witness is performed from Phase IV which still uses rewindings. The details are omitted.

E Negative Results

E.1 Universal Composition in the Direct Access Model

Canetti and Fischlin [CF01] prove that universally composable commitments are impossible to construct in the plain model. We reproduce here the details of their result in our framework where we assume that there exists Captcha puzzles (as in Definition 31) and the environment has direct access to a Captcha solving oracle H. This implies that the environment's queries cannot be seen by the ideal world adversary. For this, we just focus on the the one-time commitment functionality, and prove that it is impossible to UC-realize it in direct access model. This functionality is shown in figure 4.

In any commitment protocol which attempts to implement the functionality Figure 4, denote by C the committer P_i and by R the receiver P_j.

Definition E1 (Terminating Commitment Protocol) *A commitment protocol $\pi = \langle \mathsf{C}^H, \mathsf{R}^H \rangle$ is called* terminating *if there is a non-negligible probability*

Functionality $\mathcal{F}_{\mathbf{com}}$

$\mathcal{F}_{\text{COM}}$ acts in the presence of parties $P_1^H \cdots P_n^H$ and an ideal model adversary $\mathcal{S}^H$

1. Upon receiving the message $(\texttt{Commit}, sid, P_i, P_j, b)$, from the party P_i, $b \in \{0, 1\}$ record the value b and send the message $(\texttt{Receipt}, sid, P_i, P_j)$ to $\mathcal{S}$ and P_j, who writes it to his output tape. Ignore any subsequent $(\texttt{Commit}, \cdots)$ messages
2. Upon receiving the message $(\texttt{Open}, sid, P_i, P_j)$ from P_i, if some bit b was previously recorded, send $(\texttt{Open}, sid, P_i, P_j, b)$ to $\mathcal{S}$ and P_j (who writes it to his output tape) and halt. Else do nothing. Halt.

Fig. 4. A one-time commitment functionality[CF01] $\mathcal{F}_{\text{COM}}$

that R^H *outputs* $(\texttt{Receipt}, \cdots)$ *and moreover if the receiver, upon getting a valid decommitment for a session id sid and a committed bit b from the sender, outputs* $(\texttt{Open}, sid, C, R, b)$ *with non-negligible probability*

We say that a protocol is *bilateral* if only two parties participate in the protocol and all other parties are idle and do not send or receive messages.

Theorem E2 *There exists no bilateral terminating commitment protocol* π *that UC securely realizes the functionality* $\mathcal{F}_{\text{COM}}$ *in the direct access model of* Captcha *puzzles. This holds even if the ideal-model adversary* $\mathcal{S}^H$ *is allowed to depend on the environment* $\mathcal{Z}^H$*.*

Our impossibility result is a restatement of Canetti-Fischlin impossibility [CF01] in the plain model. We present it here for completeness, but the details are precisely as is from [CF01]. It proceeds as follows.

Recall that all turing machines described below are oracle turing machines. Assuming that a Captcha puzzle system (Definition 31) exists, and modelling it in the UC framework via the direct access model, all turing machines described below have access to the solution oracle H. Informally, let π be any protocol attempting to realize the functionality $\mathcal{F}_{\text{COM}}$ securely. We would like to construct an adversary $\mathcal{A}_2$ and environment $\mathcal{Z}_2$ such that no ideal world adversary exists satisfying the definition of UC security. To this goal, let $\mathcal{Z}_1$ and $\mathcal{A}_1$ be the environment and adversary who do the following. The adversary $\mathcal{A}_1$ corrupts the committer C and forwards all messages that he receives from the environment on behalf of the committer. The environment chooses a bit b at random and executes the honest protocol with b as input. Then $\mathcal{Z}_1$ advises the corrupt committer to start the decommitment phase, and once again lets the adversary $\mathcal{A}_1$ forward messages generated by $\mathcal{Z}_1$ on behalf of the committer. When the receiver R outputs a bit b', the environment outputs 1, iff b equals b'. Observe the following about the adversary $\mathcal{A}_1$ and π,

1. $\mathcal{A}_1$ sees nothing but the messages of the protocol. In particular, $\mathcal{A}_1$ does not make any oracle queries itself.

2. By definition of UC security, there exists an ideal model adversary $\mathcal{S}_1$ such that $\mathcal{S}_1$ sends a $(\texttt{Commit}, sid, C, R, b')$ message to the functionality $\mathcal{F}_{\text{COM}}$. Moreover, by the indistinguishability property of UC security, $\Pr[b = b']$ differs only negligibly between the real and ideal executions.
3. By the definition of a terminating commitment protocol, $\Pr[b = b']$ is non-negligible.

Formally, Let $\pi = (\mathsf{C}^H, \mathsf{R}^H)$ be any terminating protocol that UC-securely implements the functionality $\mathcal{F}_{\text{COM}}$. Let $\mathcal{A}_1^H$ and $\mathcal{Z}_1^H$ be the adversary and environment as above. Let $\mathcal{S}_1^H$ be the ideal model advesary (possibly depending on $\mathcal{Z}_1^H$) whose existence is guaranteed by the definition of UC-security. Note that due to us assuming the direct access model for CAPTCHA,

- the environment can execute a honest committer's protocol (as it can access the human oracle to solve the CAPTCHA puzzles in $\mathsf{ComExtr}$).
- the view of the ideal model adversary $\mathcal{S}_1^H$ and that of the real adversary $\mathcal{A}_1^H$ both consist of only the messages of the protocol, and in particular, neither of these machines make any oracle queries.

To prove a contradiction, consider the following environment and adversary $\mathcal{Z}_2^H$ and $\mathcal{A}_2^H$. The environment $\mathcal{Z}_2^H$ instructs the committer to pick a (secret) bit b randomly and commit to it using an honest execution of the protocol π. The adversary $\mathcal{A}_2^H$, corrupts the receiver and then obtains all the messages sent to the now corrupted receiver and forwards it to an internal copy of the simulator $\mathcal{S}_1^H$ from the previous experiment. When $\mathcal{S}_1^H$ sends a $(\texttt{Commit}, \cdots, b')$ message to $\mathcal{F}_{\text{COM}}$, the adversary forwards the bit b' to the environment. The environment outputs 1 if $b = b'$ and 0 otherwise.

The contradictions follow from the following observation. In the real world experiment of the above protocol, using $\mathcal{S}_1^H$ internally, adversary $\mathcal{A}_2$ obtains an advantage in guess the bit b. However in the ideal world execution of the protocol, since a decommitment is not being done, the ideal world adversary, $\mathcal{S}_2$ (even if it depends on $\mathcal{Z}_2$) has no advantage over $\frac{1}{2}$ in guessing the bit b. Thus for every ideal world adversary $\mathcal{S}_2$ the environment is capable of distinguishing the real world from the ideal world, contradicting the definition of UC-security.

E.2 Concurrent Self-composition of general functionalities

As noted earlier, much like the Canetti-Fischlin impossibility result, it is possible to repeat the proof of Lindell [Lin08] step-by-step with intuitive changes to obtain a negative result for concurrent self-composition of general functionalities when modelling CAPTCHA puzzles via the direct access model. Here we briefly discuss the steps involved in Lindell's result. We recommend that the reader familiarizes himself with the proof of [Lin08].

At a very high level, the first step in Lindell's result is to show that concurrent self-composition implies concurrent general-composition. The proof outline for this step is as follows. For any bi-directional bit-transmitting functionality $\mathcal{F}$, if there exists a secure self-composing protocol π which implements $\mathcal{F}$, it can be used concurrently many times to simulate the execution of π concurrently composed with any general protocol $\mathcal{G}$ by transmitting the messages of $\mathcal{G}$ bit-by-bit. This step remains completely unaltered and goes through in our setting as well; this is because in our model, the only difference is that machines are equipped with with oracle access to H. This is only a "cosmetic" change which does not affect simulation of $\mathcal{G}$ by using π concurrently many times.

The next step (to complete negative result of Lindell), shows that general composition implies universal composition for the case of so called *specialized simulator UC* [Lin03b]. Once again, since the only modification in our model is to equip machines with access to H, this step also goes through without any change. Finally, the result of Canetti-Fischlin (reproduced in theorem E2 for our model) proves that specialized-simulator UC is not possible for the commitment functionality. This is essentially the entire outline of our impossibility claim for concurrent self-composition of bi-directional bit-transmitting functionalities. A complete proof can be obtained by a step-by-step reproduction of Lindell's results.