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 at Austin
omkant@cs.utexas.edu

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 ([1], 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.

⋆ Department of Computer Science and Mathematics, UCLA, Email: rafail@cs.ucla.edu. Research supported in part by NSF grants CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

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 [1] 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 [1] 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 [3] 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.

[3] For many more uses of Captcha, see [2]

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 [4] 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 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 [5], "straight-line" extractable commitment schemes, constant-round fully concurrent zero-knowledge for **NP** [6], fully concurrent two/multi-party computation [7–9], universal composition *without* trusted setup assumptions [10, 11], 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" [12] in the protocol of Barak, Prabhakaran, and Sahai [13], we obtain a constant-round protocol for *concurrent non-malleable zero-knowledge* [13] (see appendix D of the full version [14]).[4]

As a natural extension, we investigate the issue of incorporating CAPTCHA puzzles in the UC framework introduced by Canetti [10]. 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. [4], 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

[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 [13]: we need to use a constant round non-malleable commitment scheme and not that of [15]. We can use any of the schemes presented in [16–19].

[5] We assume basic familiarity with the model of universal composition, and briefly recall it in appendix C.1 of the full version [14] .

construct a constant-round *UC-puzzle* protocol as defined by Lin, Pass, and Venkitasubramaniam [20]. By the results of [20], 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** [21, 22].

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 [23] (details reproduced in appendix E.1 of the full version [14]). 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 [24, 25], 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 [26]. 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 [4, 3] 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 [4, 3], 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 [1, 4, 3] 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 [27, 28] for a detailed treatment of this notion.

Notation. The output of an oracle ITM M^H is denoted by a triplet $(\mathtt{out}, \bar{q}, \bar{a})$ where $\mathtt{out}, \bar{q}$, 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 [27]).

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 [29].

Definition 21 (Statistically Secure Oblivious Transfer), [29] *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* $\mathrm{negl}(\cdot)$ *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 [29], 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). [30] 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 [1, 3, 4] 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" [1]. However, a non-asymptotic formulation will be insufficient for cryptographic purposes. For many puzzles, typically hardness can be amplified by sequential or parallel repetition[31].

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 [1, 3]. 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* [15]. 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 [3, 4] 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 $\bar{q} = (q_1, q_2, \ldots)$ 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* $\mathrm{negl}(\cdot)$*, such that for every oracle* ppt *algorithm* $\mathcal{A}^H$*, and every sufficiently large* $k \in \mathbb{N}$*, we have that:*

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

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 [32, 33]. 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 [34].

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 [3, 4], 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.

[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

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 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 $\mathcal{M}(c, \mathsf{aux}) = \perp$.

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:*

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

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 1)

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*

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 $\langle \mathsf{C}^H, \mathsf{R} \rangle$

figure 1 is a 3-round perfectly-binding and computationally-hiding commitment scheme which admits straight-line extraction.

Proof. A full proof may be found in Appendix A of the full version [14]. ■

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* [20] 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 [10], 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 of the full version [14] .

Lin, Pass and Venkitasubramaniam [20] 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 [20] 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 [20].

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 $m = \mathrm{poly}(k)$ (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 $\text{IDEAL}_{\mathsf{Sim}^H,\mathcal{Z}}(k)$.

Definition 51 (UC-Puzzle, adapted from [20]) *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* $\text{negl}(\cdot)$ *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* $\text{negl}(k)$.
- 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 $\text{REAL}^H_{\mathcal{A},\mathcal{Z}}(k)$ and $\text{IDEAL}_{\mathsf{Sim}^H,\mathcal{Z}}(k)$ 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 $y \in \mathcal{R}(\mathsf{TRANS})$.*

The UC-puzzle System. Due to space constraints, here we only sketch the construction of our UC-puzzle, and defer the details to the full version [14]. A straightforward approach that does not quite work is to use our extractable commitment from Figure 1. That is, the sender of the UC puzzle picks random string s, which will serve as the trapdoor, and commits to it using our extractable commitment. Although this scheme allows extraction of the trapdoor s, it is not clear how, given a transcript and a purported trapdoor, it can be verified in PPT whether it is the correct trapdoor or not. Further, a malicious sender may commit to an invalid string (by using incorrect Captcha solution, for example). The receiver can not detect this and will accept, while there is no well-defined trapdoor for such a transcript. Moreover, we can not use the standard trick of using zero-knowledge to enforce correct sender behaviour because checking validity of Captcha solutions is not a PPT process.

We solve the first problem by making the sender additionally send $z := f(s)$ to the receiver, where $f(\cdot)$ is a one-way function. The idea is to make it easy to verify the trapdoor, by simply checking if z is the image of the trapdoor under $f(\cdot)$. However, for this to work, we must ensure that the pre-image of z and the string committed in the extractable commitment are the same.

To solve this problem, we use the following modified commitment scheme in the above protocol to commit to the trapdoor: to commit to a string s, the sender commits s twice, first using our straight line extractable commitment from Figure 1, and then using any non-interactive perfectly binding scheme $\langle C_{\text{PB}}, R_{\text{PB}}\rangle$ (which can be constructed from, for eg., one-way functions). Using this, we tackle the aforementioned problem in two steps:

1. First, the committer proves that the commitment is 'well-formed': that is, the string committed in both the commitments is the same. This is done by using secret sharing and cut-and-choose.
2. Thereafter, the committer gives a zero-knowledge proof that the string committed using the commitment scheme $\langle C_{\text{PB}}, R_{\text{PB}} \rangle$ is the same as the pre-image of z under $f(\cdot)$.

The first step ensures that the string committed using the commitment scheme of Figure 1 is the same as that committed by $\langle C_{\text{PB}}, R_{\text{PB}} \rangle$. As $\langle C_{\text{PB}}, R_{\text{PB}} \rangle$ is in the plain model and does not involve CAPTCHA, we can give a proof of correctness using standard zero-knowledge. For full details, please refer to the full version [14].

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 [15, 35, 36, 17]?

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 [37]. One of the main difficulties in [37] 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 [38, 39]. However, since we obtain straight-line simulation, it seems likely that our techniques could be used to improve the results in [37]. 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 [37] 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 [1, 4, 3] 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

1. Ahn, L.V., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using hard AI problems for security. In: EUROCRYPT. (2003) 294–311

[7] By using standard techniques, e.g., coin-tossing using our commitment scheme along with Blum's protocol [22], we can obtain a 5-round (concurrent) zero-knowledge protocol. But we do not know how to reduce it to 3 rounds.

2. : The Official CAPTCHA Site `www.captcha.net`.
3. Dziembowski, S.: How to pair with a human. In: SCN. (2010) 200–218
4. Canetti, R., Halevi, S., Steiner, M.: Mitigating dictionary attacks on password-protected local storage. In: ADVANCES IN CRYPTOLOGY, CRYPTO, Springer-Verlag (2006)
5. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: EUROCRYPT. (1994) 92–111
6. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC. (1998) 409–418
7. Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC. (2003) 683–692
8. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds. In: FOCS. (2003)
9. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: STOC. (2004) 232–241
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. (2001) 136–145
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC. (2002) 494–503
12. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS. (2002) 366–375
13. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, IEEE Computer Society (2006) 345–354
14. Kumarasubramanian, A., Ostrovsky, R., Pandey, O., Wadia, A.: Cryptography using captcha puzzles. Cryptology ePrint Archive, Report 2012/689 (2012) `http://eprint.iacr.org/`.
15. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. SIAM J. on Computing **30**(2) (2000) 391–437
16. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC. (2005) 533–542
17. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: CRYPTO. (2008) 57–74
18. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: STOC. (2011) 705–714
19. Goyal, V.: Constant round non-malleable protocols using one way functions. In: STOC. (2011) 695–704
20. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC. (2009) 179–188
21. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: FOCS. (1986) 174–187
22. Blum, M.: How to prove a theorem so no one else can claim it. In: International Congress of Mathematicians. (1987) 1444–1451
23. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO. (2001) 19–40
24. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: In 44th FOCS. (2003) 394–403
25. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. Journal of Cryptology **21** (2008) 200–249 10.1007/s00145-007-9015-5.

26. Blum, M., Santis, A.D., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. **20**(6) (1991) 1084–1118
27. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press (2001)
28. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press (2009)
29. Halevi, S., Kalai, Y.: Smooth projective hashing and two-message oblivious transfer. Journal of Cryptology (2010) 1–36 10.1007/s00145-010-9092-8.
30. Hemenway, B., Ostrovsky, R.: Lossy trapdoor functions from smooth homomorphic hash proof systems. Electronic Colloquium on Computational Complexity (ECCC) **16** (2009) 127
31. Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: TCC, Springer-Verlag (2004) 17–33
32. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: CRYPTO. (1998) 408–423
33. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: CRYPTO. (2004) 273–289
34. Naor, M.: On cryptographic assumptions and challenges. In: CRYPTO. (2003) 96–109
35. Crescenzo, G.D., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: STOC. (1998) 141–150
36. Crescenzo, G.D., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: EUROCRYPT. (2001) 40–59
37. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: CRYPTO. (2010) 277–294
38. Micali, S., Pass, R.: Local zero knowledge. In: STOC. (2006) 306–315
39. Pandey, O., Pass, R., Sahai, A., Tseng, W.L.D., Venkitasubramaniam, M.: Precise concurrent zero knowledge. In: EUROCRYPT. (2008) 397–414