Non-Malleable Time-Lock Puzzles and Applications

Cody Freitag[1], Ilan Komargodski[2], Rafael Pass[1], and Naomi Sirkin[1]

[1] Cornell Tech
{cfreitag,rafael,nephraim}@cs.cornell.edu
[2] Hebrew University and NTT Research
ilank@cs.huji.ac.il

Abstract. Time-lock puzzles are a mechanism for sending messages "to the future", by allowing a sender to quickly generate a puzzle with an underlying message that remains hidden until a receiver spends a moderately large amount of time solving it. We introduce and construct a variant of a time-lock puzzle which is *non-malleable*, which roughly guarantees that it is impossible to "maul" a puzzle into one for a related message without solving it.

Using non-malleable time-lock puzzles, we achieve the following applications:

- The first fair non-interactive multi-party protocols for coin flipping and auctions in the plain model without setup.
- Practically efficient fair multi-party protocols for coin flipping and auctions proven secure in the (auxiliary-input) random oracle model.

As a key step towards proving the security of our protocols, we introduce the notion of functional non-malleability, which protects against tampering attacks that affect a specific function of the related messages. To support an unbounded number of participants in our protocols, our time-lock puzzles satisfy functional non-malleability in the fully concurrent setting. We additionally show that standard (non-functional) non-malleability is impossible to achieve in the concurrent setting (even in the random oracle model).

1 Introduction

Time-lock puzzles (TLPs), introduced by Rivest, Shamir, and Wagner [45], are a cryptographic mechanism for committing to a message, where a sender can (quickly) generate a puzzle with a solution that remains hidden until the receiver spends a moderately large amount of time solving it (even in the presence of parallel processors). Rivest et al. [45] gave a very efficient construction of TLPs where security relies on the repeated squaring assumption. This assumption postulates, roughly, that it is impossible to significantly speed up repeated modular exponentiations in a group of unknown order, even when using many parallel processors. This construction and assumption have proven extremely useful in various (and sometimes unexpected) applications [9, 33, 44, 50, 19, 36, 15], some of which have already been implemented and deployed in existing systems.

Non-malleability. In a Man-In-the-Middle (MIM) attack, an eavesdropper tries to actively maul intermediate messages to compromise the integrity of the underlying values. To address such attacks, Dolev, Dwork and Naor [18] introduced the general concept of non-malleability in the context of cryptographic commitments. Roughly speaking, non-malleable commitments are an extension of plain cryptographic commitments (that guarantee binding and hiding) with the additional property that no adversary can maul a commitment for a given value into a commitment to a "related" value. As this is a fundamental concept with many applications, there has been a tremendous amount of research on this topic [1, 41, 40, 35, 39, 31, 43, 49, 22, 32, 23, 24, 11, 12, 29, 33, 30, 27].

Non-malleable TLPs and applications. To date, non-malleability has not been considered in the context of TLPs (or other timed primitives).[3] Indeed, the construction of TLPs of [45] *is malleable*.[4] This fact actually has negative consequences in various settings where TLPs could be useful. For instance, consider a scenario where n parties perform an auction by posting bids on a public bulletin board. To implement this fairly, a natural approach is to use a commit-and-reveal style protocol, where each party commits to its bid on the board, and once all bids are posted each party publishes its opening. Clearly, one has to use non-malleable commitments to guarantee that bids are independent (otherwise, a malicious party can potentially bid for the maximal other bid plus 1). However, non-malleability is not enough since there is a fairness issue: a malicious party may refuse to open after seeing all other bids and so other parties will never know what the unopened bid was.

Using non-malleable TLPs to "commit" to the bids solves this problem. Indeed, the puzzle of a party who refuses to reveal its bid can be recovered after some moderately large amount of time by all honest parties. This style of protocol can also be used for fair multi-party collective coin flipping where n parties wish to agree on a common unbiased coin. There, each party encodes a random bit via a TLP and all parties will eventually agree on the parity of those bits.[5] This gives a highly desirable collective coin flipping protocol with an important property that we refer to as *optimistic efficiency*: when all parties are honest and publish their "openings" immediately after seeing all puzzles, the protocol terminates and all parties agree on an unbiased bit. As we will see, no other known protocol for this (highly important) task has this property. Even ignoring

[3] The concurrent works of [28, 3, 4] consider similar notions of non-malleability for time-lock puzzles. See Section 1.3 for a detailed comparison.

[4] The puzzle of [45] for a message s and difficulty T is a tuple $(g, N, T, s \oplus g^{2^T} \bmod N)$, where N is an RSA group modulus and g is a random element from $\mathbb{Z}_N$. The puzzle is trivially malleable since the message is one-time padded.

[5] In the context of coin flipping, if a malicious party aborts prematurely, this can bias the output [13] causing the fairness issue mentioned above. Boneh and Naor [9] used timed primitives and interaction to circumvent the issue in the two-party case, but we care about the multi-party case and prefer to avoid interaction as much as possible.

optimistic efficiency, such a protocol yields a fully non-interactive coin flipping protocol where each participant solves all published puzzles.

1.1 Our Results

To present our results, we start with a high level definition of a non-malleable TLP. Recall that for some secret s and difficulty t, a time-lock puzzle enables sampling a puzzle z which can be solved in time t to recover s, but guarantees that s remains hidden to any adversary running in time less than t.

For non-malleability, we require that any man-in-the-middle (MIM) attacker $\mathcal{A}$ that receives a puzzle z "on the left" cannot output a different puzzle $\widetilde{z}$ "on the right" to a related value. Formally, we consider the (inefficient) distribution $\mathsf{mim}_{\mathcal{A}}(t, s)$ that samples a puzzle z to s, gets $\widetilde{z} \leftarrow \mathcal{A}(z)$, and outputs the value $\widetilde{s}$ computed by solving $\widetilde{z}$. However, if $z = \widetilde{z}$, then $\widetilde{s} = \perp$ (since simply forwarding the commitment does not count as a valid mauling attack). Then, non-malleability requires that for any solution s and MIM attacker with depth much less than t (so it cannot break hiding), the distribution for a value s given by $\mathsf{mim}_{\mathcal{A}}(t, s)$ is indistinguishable from the distribution $\mathsf{mim}_{\mathcal{A}}(t, 0)$ for an unrelated value, 0. We emphasize that indistinguishability should hold even against arbitrary polynomial time or even *unbounded* distinguishers that, in particular, can solve the TLP. We also consider the natural extension to the bounded *concurrent* setting [42], where the MIM attacker $\mathcal{A}$ receives n_{left} concurrent puzzles on the left and attempts to generate n_{right} puzzles on the right to related values. In this setting, the distinguisher receives the solutions to all n_{right} puzzles. We refer to this as $(n_{\mathsf{left}}, n_{\mathsf{right}})$-concurrency.

We next give our main results. First, we present our results on non-malleable time-lock puzzles, and we discuss the various notions of non-malleability that we consider in this setting. Next, we show how to additionally satisfy a strong public verifiability property using a specific time-lock puzzle based on repeated squaring. Finally, we discuss the applications of our constructions for fair multiparty protocols.

Non-malleable time-lock puzzles. We give two different constructions of non-malleable TLPs. We emphasize that, as explained above, this primitive is not only natural on its own right, but also has important applications to the design of secure protocols for various basic tasks. Our first construction is practically efficient, relies on the existence of any given TLP [45, 6], and is proven secure in the (auxiliary-input) random oracle model [47].

Theorem 1.1 (Informal; See Theorem 4.2 and Corollary 4.3). *For every $n_{\mathsf{left}}, n_{\mathsf{right}}, L \in \mathrm{poly}(\lambda)$, assuming that there is a TLP (supporting 1-bit messages) that is secure for attackers of size $2^{3n_{\mathsf{right}} \cdot L} \cdot \mathrm{poly}(\lambda)$, there exists an* $(n_{\mathsf{left}}, n_{\mathsf{right}})$-concurrent non-malleable *TLP supporting messages of length L. The scheme is proven secure in the auxiliary-input random oracle model.*

In terms of security, our reduction is *depth preserving*: if the given TLP is secure against attackers of depth $T(\lambda)/\alpha(\lambda)$, where $\alpha(\cdot)$ is a fixed polynomial independent of T denoting the advantage of an attacker, then the resulting non-malleable TLP is secure against attackers of depth $T(\lambda)/\alpha'(\lambda)$ for a related fixed polynomial $\alpha'(\cdot)$. In particular, the dependence on T in hardness is preserved. Additionally, note that if $n_{\mathsf{right}} \cdot L \in O(\log \lambda)$, then the underlying TLP only needs to be polynomially secure.

Instantiating the TLP with the construction of [45], our scheme is extremely efficient: encoding a message requires a single invocation of a random oracle and few (modular) exponentiations. Additionally, our construction is very simple to describe: to generate a puzzle for a solution s with randomness r, we sample a puzzle for (s, r) using randomness which itself depends (via the random oracle) on s and r.[6] Nevertheless, the proof of security turns out to be somewhat tricky and non-trivial; see Section 2 for details.

We prove that our scheme is non-malleable against all polynomial-size attackers that cannot solve the puzzles (and this is inherent as the latter ones can easily maul any puzzle). We even allow the attacker's description to depend arbitrarily on the random oracle. We formalize this notion by showing that our TLP is non-malleable in the *auxiliary-input* random oracle model, a model that was introduced by Unruh [47] (see also [14]) in order to capture preprocessing attacks, where a non-uniform attacker obtains an advice string that depends arbitrarily on the random oracle. Thus, in a sense, our construction does *not* require any form of attacker-independent setup.

Our second construction is proven secure in the plain model (without any form of setup) and is based on the non-malleable code for bounded polynomial depth tampering functions due to [15]. This construction relies on a variety of assumptions (including keyless hash functions and non-interactive witness indistinguishable proofs) and is less practically efficient. While the main technical ideas for the construction and proof are given in [15], the threat model they consider is weaker than what we require for non-malleable TLPs; for example, they only consider plain (non-concurrent) non-malleability and do not require security against re-randomization attacks (mauling a code word for m into a different code word for m). We show how to extend their construction to our setting, and prove the following theorem.

Theorem 1.2 (Informal). *Assume a time-lock puzzle, a keyless multi-collision resistant hash function, a non-interactive witness indistinguishable proof for* NP*, and injective one-way functions, all sub-exponentially secure. Then, there exists a bounded concurrent non-malleable time-lock puzzle secure against polynomial size adversaries.*

We emphasize that both of our constructions only achieve *bounded* concurrency, where the number of instances the attacker participates in is a priori

[6] We note that our construction is conceptually similar to the Fujisaki-Okamoto (FO) transformation [21] used to generically transform any CPA-secure public-key encryption scheme into a CCA-secure one using a random oracle. However, since our setting and required guarantees are different, the actual proof turns out to be much more delicate and challenging.

bounded (and the scheme may depend on this bound). We show that the stronger notion of *full* concurrency, which does not place such limitations and is achievable in all other standard settings of non-malleability, is actually impossible to achieve for TLPs. Therefore, our result is best possible in this sense.

Theorem 1.3 (Informal). *There is no fully concurrent non-malleable TLP (even in the random oracle model).*

In a nutshell, the impossibility from Theorem 1.3 is proven by the following generic MIM attack. Given a puzzle z, if the number of "sessions" the attacker can participate in is at least as large as $|z|$, they can essentially generate $|z|$ puzzles encoding *the bits of* z. Since the distinguisher of the MIM game (which is now given those bits) can run in arbitrary polynomial time, it can simply solve the original puzzle and recover the original solution in full. We emphasize that this attack only requires a polynomial-time distinguisher. This attack is circumvented in the bounded concurrency setting (Theorem 1.1) by setting the length of the puzzle to be longer than the concurrency bound. Specifically, to support n concurrent puzzles on the right, we can set the message length to $L \cdot n$, which is what results in exponential security loss $2^{L \cdot n}$ as discussed above.

Functional non-malleability. We note that the attack on fully concurrent non-malleable time-lock puzzles crucially relies on the fact that the *distinguisher* in the MIM game can solve the underlying puzzles. However, it is easy to see that if the distinguisher is restricted to bounded depth, this attack fails. One could define a *weaker* notion of non-malleability where the MIM distinguisher is depth-bounded, but this results in a weaker security guarantee. In particular, we show in the full version of the paper that there exists a natural TLP construction that satisfies this (weaker) definition yet has a valid mauling attack.[7]

In light of this observation, we introduce a new definition of non-malleability that *generalizes* the standard definition considered in Theorem 1.1. We call the notion *functional* non-malleability and, as the name suggests, the security notion is parameterized by a class of functions $\mathcal{F}$. Denote by L the bit-length of the messages we want to support and by n the number of sessions that the MIM attacker participates in on the right. We think of $f \in \mathcal{F}$ as some *bounded depth* function of the form $f \colon (\{0,1\}^L)^n \to \{0,1\}^m$, which is the target function of the input messages that the MIM adversary is trying to bias. Specifically, the distinguisher of the MIM game now receives the output of the function f when applied to the values underlying the puzzles given by the MIM adversary. When $\mathcal{F}$ includes all identity functions (which are bounded depth and have output length $m = n \cdot L$), functional non-malleability implies the standard definition of concurrent non-malleability (as the distinguisher just gets all the messages from the n mauled puzzles).

Naturally, it makes sense to ask what guarantees can we get if we a priori restrict f, say in its output length, without limiting the number of sessions n.

[7] As we discuss in the Section 1.3, concurrent works allow the distinguisher to be bounded depth.

This turns out to particularly useful when the application at hand only requires non-malleability against a specific form of tampering functions (this indeed will be the case for us below). Concretely, let $\mathcal{F}_m$ be the class of all functions whose output length is at most m bits and which can be computed in depth polynomial in the security parameter λ and in $\log(n \cdot L)$ (using the notation given above). Then, we have the following result.

Theorem 1.4 (Informal; See Theorem 4.2). *Assuming that there exists a TLP, then for every $m \in \mathrm{poly}(\lambda)$ there exists a* fully concurrent functional non-malleable *TLP for the class of functions $\mathcal{F}_m$. The scheme is proven secure in the auxiliary-input random oracle model assuming the given TLP is secure for all attackers of size at most $2^{3m} \cdot \mathrm{poly}(\lambda)$.*

The above construction is *depth preserving* in the same way as the construction from Theorem 1.1. Further, note that as long as $m \in O(\log \lambda)$, we only require standard polynomial hardness from the given TLP. We remark that Theorem 1.4 will turn out to be instrumental for our applications we discuss below. We also believe that the abstraction of functional non-malleability is important on its own right and view it as an independent contribution. We also show how to achieve fully concurrent functional non-malleability for our plain model construction.

Publicly verifiable time-lock puzzles. In addition to non-malleability, we construct TLPs that also have a public verifiability property: after a party solves the puzzle, they can publish the underlying solution together with a proof which can be later used by anyone to *quickly* verify the correctness of the solution. We emphasize that this must hold even if the solver determines that the puzzle has no valid solution. We believe this primitive is of independent interest.

We build our non-malleable, publicly verifiable TLP assuming a very weak form of (partially) trusted setup. The setup of our TLP consists of a set of many public parameters where we only assume that at least one of them was generated honestly. We call this model the All-But-One-string (ABO-string) model.[8] We design this to fit into our multi-party protocol application (see Theorem 1.7 below) in such a way where the parties themselves will generate this setup in the puzzle generation phase. Indeed, as we discuss below, publicly verifiable TLPs in the ABO-string model will imply coin flipping *without setup*.

Theorem 1.5 (Informal). *Assuming the repeated squaring assumption, there exists a publicly verifiable non-malleable TLP in the ABO-string model. The construction is proven secure in the auxiliary-input random oracle model.*

Our construction is depth preserving and has security which depends on the message length. In particular, the security of the resulting TLP is the same as

[8] Our ABO-string model is a variant of the multi-string model of Groth and Ostrovsky [25], where it is assumed that a majority of the public parameters are honestly generated.

in the constructions in Theorem 1.1 and Theorem 1.4, depending on the type of non-malleability desired for the resulting TLP.

To construct our publicly verifiable TLP, we use a *strong trapdoor VDF* which is why our construction is not generic from any time-lock puzzle. Somewhat surprisingly, we need to leverage specific properties of the trapdoor VDF of Pietrzak's [44] using the group of signed quadratic residues QR_N^+ where N is a product of safe primes.[9] For an overview of our construction, see Section 2.2.

Fair multi-party auctions and coin flipping. As we mentioned above, an appealing application of non-malleable TLPs is for tasks such as fair multi-party auctions or coin flipping. Our protocols (for both tasks) are extremely efficient and consist of just two phases: first each party "commits" to their bid/randomness using some puzzle, and then after all puzzles are made public, each party publishes its solution. If some party refuses to open their puzzle, a force-opening phase is performed. Alternatively, we can instantiate our protocols in the fully non-interactive setting where all parties solve every other puzzle.

In what follows, we focus on the task of fair multi-party coin flipping, which is a core building block in recent proof-of-stake blockchain designs; see below. The application to auctions follows in a similar manner. It is convenient to consider our protocol in a setting where there is a public bulletin board. Any party can publish a puzzle to the bulletin board during the commit phase and then publish its solution after some pre-specified amount of time has elapsed.

Relying only our concurrent, functional non-malleable (not necessarily publicly verifiable) TLP constructions, all of our protocols (both non-interactive and two-phase) satisfy fairness, informally defined as follows:

- **Fairness:** No malicious adversary (controlling all but one party) can bias the output of the protocol, even by aborting early. Namely, as long as there is at least one honest participating party, the output will be a (nearly) uniformly random value.

Our two-phase "commit-and-reveal" style protocols have the additional efficiency guarantee:

- **Optimistic Efficiency:** If all participating parties are honest, then the protocol terminates within two message rounds (without the need to wait the pre-specified amount of time for the second phase), and all parties can efficiently verify the output of the protocol.

Using our construction of a publicly verifiable non-malleable TLP, we satisfy the following public verifiability property:

- **Public Verifiability:** In the case that any participating party is dishonest and does not publish their solution, any party can break the puzzle in a moderate amount of time and provide a publicly verifiable proof of the solution.

[9] For this, we assume that sampling uniformly random safe primes can be done efficiently; this is a pretty common assumption, see [48] for more details.

We even require that an honest party can prove that a published puzzle has *no* valid solution.

We focus on two main results from the above discussion, although we get a variety of different protocols depending on what TLP we start with and how we instantiate the protocol. First, we construct fully non-interactive protocols in the plain model without any setup.

Theorem 1.6 (Informal; see Theorem 5.3). *Assume a time-lock puzzle, a keyless multi-collision resistant hash function, a non-interactive witness indistinguishable proof for* NP*, and injective one-way functions, all sub-exponentially secure. Then, there exist fully non-interactive, fair multi-party coin flipping and auction protocols. The protocols support an unbounded number of participants and require no setup.*

Next, we achieve efficient, publicly verifiable two-phase protocols in the auxiliary input random oracle model.

Theorem 1.7 (Informal; See Theorem 5.1). *Assuming the repeated squaring assumption, there exist two-phase fair multi-party coin flipping and auction protocols that satisfy optimistic efficiency and public verifiability. The protocols support an unbounded number of participants and require no trusted setup. Security is proven in the auxiliary-input random oracle.*

The differences between the protocols achieved in these two theorems is that the first is non-interactive and has no setup, while the second is two rounds and is in the random oracle model, yet leverages this to achieve public verifiability and better concrete efficiency. We emphasize that both of the protocols support polynomial-length outputs, relying on sub-exponential security of the underlying time-lock puzzle.

We also emphasize that our protocols support an a priori unbounded number of participants. This may seems strange in light of our impossibility from Theorem 1.3. We bypass this lower bound (as mentioned above) by observing that for most natural applications (including coin flipping and auctions), the notion of functional non-malleability from Theorem 1.4 suffices. The key insight is that we only need indistinguishability with respect to specific depth-bounded functions with a priori bounded output lengths (e.g., parity for coin flipping, or taking the maximum for auctions). Since the output length in both cases is known, we can actually support *full* concurrency which translates into having an unbounded number of participants.

For auctions, we note that our protocols are the first multi-party protocols *under any assumption* that satisfy fairness against malicious adversaries and requires no adversary-independent setup—using the timed commitments of [9] works only in the two-party setting and additionally relies on trusted setup, and using the homomorphic time-lock puzzles of [36] does not satisfy fairness in the presence of malicious adversaries. For coin flipping, our two-round protocol is the first multi-party protocol that is fair against malicious adversaries while satisfying optimistic efficiency. Next, we provide a more in depth comparison of our non-interactive coin flipping protocol with existing solutions.

Simulation-based fairness. As mentioned above, we show that our protocols are fair in the sense that no malicious adversary can bias the output of the protocol. This suffices for applications which only use the *output* of the protocol. To capture applications that additionally depend on the protocol transcript, we show that our protocol satisfies simulation security with full fairness in the programmable random oracle model. This guarantees that the protocol execution in the presence of a malicious adversary (even one aborting early) can be simulated to a uniformly random output in an ideal model where every honest party receives the output (regardless of whether any malicious party aborts early).

Non-interactive coin flipping. We emphasize that our non-interactive coin flipping protocol of Theorem 1.6 is the first such protocol without any form of setup in the plain model. Specifically, we mean that there is no common random string or any assumed common function. Still, our practically efficient protocol of Theorem 1.7 as a non-interactive protocol still enjoys some benefits over existing schemes.

In the non-interactive setting, Boneh et al. [7] proposed a VDF-based protocol. Specifically, each party publishes a random string r_i and then the agreed upon coin is defined by running a VDF on the seed $H(r_1\|\ldots\|r_n)$, where H is a random oracle. As the VDF must be evaluated to obtain the output, this type of protocol does not satisfy optimistic efficiency. Nevertheless, the VDF-based protocol has the advantage that only a single slow computation needs to be computed, whereas our non-interactive protocol requires n such computations for n participants (which can be done in parallel). Malavolta and Thyagarajan [36] address this inefficiency in the context of time-lock puzzles (which do allow for the option of optimistic efficiency) by constructing *homomorphic* time-lock puzzles, where many separate puzzles can be combined into a single puzzle to be solved. However, their TLP scheme is malleable and so cannot be directly used to obtain a fair protocol against malicious adversaries.[10] In the two-phase setting, however, our publicly verifiable protocol has the property that only a single honest party needs to solve each puzzle, and this computation can easily be delegated to an external server.

The VDF-based scheme of [7] can be based on repeated squaring in a group of unknown order based on the publicly verifiable proofs of [50, 44]. In this setting, the protocols can either be instantiated using RSA groups that require attacker-independent trusted setup, or based on class groups that rely only on a common random string. As we do in this work, the common random string can be implemented in the ABO-string model using a random oracle (which the attacker may depend on arbitrarily). Therefore, when restricting our attention to protocols without attacker-independent setup, the previous VDF-based protocols are based on less standard assumptions on class groups, whereas we give

[10] It is possible to make this protocol maliciously secure using concurrent non-malleable zero-knowledge proofs [2, 38, 34, 32], proving that each party acted honestly, but this (1) makes the construction significantly less efficient, and (2) requires either trusted setup and additional hardness assumptions, or additional rounds of interaction.

a protocol that can be instantiated from more standard assumptions on RSA groups with better concrete efficiency.

Privacy. Let us remark that the protocols that we described guarantee fairness but not privacy. The latter, however, can be obtained in specific applications by composing our protocols with existing privacy-preserving tools such as Anonize [26].

1.2 Related Work

Timed commitments. Boneh and Naor [9] introduced timed commitments, which can be viewed as a publicly verifiable and interactive TLP. They additionally require that the puzzle (which is an interactive commitment) convinces the receiver that if they brute-force the solution, they will succeed. Because of this additional property, their commitment scheme is interactive and relies on a less standard assumption called the generalized Blum-Blum-Shub assumption. Their scheme is additionally malleable.

Fair coin flipping in blockchains. Generating unbiased bits is one of the largest bottlenecks in modern proof-of-stake crypto-currency designs [5, 16, 17]. Recall that in a proof-of-stake blockchains, the idea is, very roughly speaking, to enforce "one vote per unit of stake". This is usually implemented by choosing random small committees at every epoch and letting that committee decide on the next block. The main question is how to obtain "pure" randomness so that the chosen committee is really "random".

One option is to use the hash of an old-enough block as the randomness. Unfortunately, it is known that the hash of a block is not completely unbiased: an attacker can essentially fully control about logarithmically many of its bits. In existing systems, this is mitigated by "blowing up" parameters to compensate for the (small yet meaningful) advantage the attacker has, making those systems much less efficient. Using a mechanism that generates unbiased bits, we could make proof-of-stake crypto-currencies much more efficient.

1.3 Concurrent Work

Several related papers [4, 3, 28] have been developed concurrently and independently to this work.[11] The works of Baum et al. [4, 3] formalize and construct various (publicly verifiable) time-based primitives, including TLPs, under the Universal Composability (UC) framework [10]. Katz et al. [28] (among other results, less related to ours) introduce and construct non-malleable non-interactive

[11] We emphasize that only Section 1.3, Appendix A, and the separation regarding the different notions of non-malleability (given in the full version) were added based on these works. All other definitions and results that appear are completely independent of these works.

timed commitments. While the notions that are introduced and studied are related, the results are all incomparable as each paper has a somewhat different motivation which leads to different definitions and results.

Comparison with [28]. Let us start by comparing definitions. Katz et al. consider a CCA-style definition adapted to the depth-bounded setting. In the classical setting of unbounded polynomial-time attackers, CCA security definitions are usually stronger than "only" non-malleability, but this is not generally true in the depth-bounded setting.

In more detail, they consider a depth-bounded version of CCA security, where the attacker (who is also the distinguisher) is bounded to run in time less than the hardness of the timed primitive. We, on the other hand, allow the distinguisher of the MIM game to be unbounded (while only the attacker is bounded). We believe this is an important distinction and we provide more insights into the differences between the bounded and unbounded distinguisher settings in the full version. Specifically, we show that non-malleability with a depth-bounded distinguisher is (essentially) equivalent to our definition of functional non-malleability with output length 1. We also give a construction separating the definitions of non-malleability with an unbounded vs. depth-bounded distinguisher, showing that non-malleability in the bounded distinguisher setting gives a strictly weaker security guarantee.

Regarding the primitives constructed, recall that timed commitments [9] (ignoring non-malleability for now) allow one to commit to a message m in such a way that the commitment hides m up to some time T, yet the verifier can be sure that it can be force opened to *some* value after roughly T time. In contrast, plain TLPs are not necessarily guaranteed to contain valid messages. In this context, our notion of publicly-verifiable TLPs is in between these two notions: we treat puzzles without a solution as invalid (say encoding $\perp$) but we additionally provide a way to publicly verify that this is the case after it has been solved. Nevertheless, we note that the construction of Katz et al. does not imply a TLP since their commitment procedure takes T time (while TLP generation should take time essentially independent of T).

Additionally, their constructions achieve non-malleability through the use of NIZKs following the Naor-Yung [37] paradigm for CCA-secure encryption. Known (even interactive) zero-knowledge proofs for correctness of time-lock puzzles are quite expensive (see, e.g., Boneh-Naor [9] which requires parallel repetition). Using generic NIZKs (even in the random oracle model) would be even worse.

Regarding assumptions, their construction is proven secure in the algebraic group model [20] and relies on trusted setup, while ours is proven secure in the (auxiliary-input) random oracle model and hence requires no trusted setup independent of the adversary. Both constructions rely on repeated squaring as the source of depth-hardness, and theirs additionally makes use of NIZKs (which require setup).

Comparison with [4, 3]. Baum et al consider a UC-style definition, which is generally stronger than non-malleability. In this setting, the environment takes the place of the distinguisher in the MIM game. Their definition is closer to ours as the environment may run for an arbitrary polynomial number of rounds and thus does not restrict the depth of the distinguisher. In terms of modeling, the construction of a UC-secure TLP in [4] relies on a programmable random oracle, whereas our construction relies on a non-programmable (auxiliary-input) random oracle. In fact, they prove that their notion of UC security cannot be achieved in the non-programmable random oracle model.

In a follow-up work [3], they show that their time-lock puzzle construction satisfies a notion of public verifiability. However, they achieve public verifiability only for honestly generated puzzles, that is, one can prove that a puzzle has a solution s, but cannot prove that a puzzle has no solution. In our terminology, we refer to this as one-sided public verifiability In contrast, our construction achieves full verifiability. This property is crucial for our efficient coin flipping protocol since it allows only one honest party to (attempt to) solve any invalid puzzle. With only one-sided public verifiability, every participant would need to solve all invalid puzzles, and the output of the coin-flip can only be efficiently verified (in time less than T) in the case that all puzzles are honestly generated.

1.4 Paper Organization

In Section 2, we give an overview of our techniques. Next, we give preliminaries in Section 3. In Section 4, we give our construction of functional non-malleable time-lock puzzles in the random oracle model. In Section 5, we give our construction of fair multi-party coin flipping. In Appendix A we discuss the various notions of non-malleability for TLPs introduced in this and related works. Additional results are provided in the full version, including our non-malleable TLP construction in the plain model, an impossibility result for unbounded concurrency, our publicly verifiable TLP construction, our simulation-secure coin flipping protocol, and a separation between unbounded and depth-bounded non-malleability.

2 Technical Overview

In Section 2.1, we give an overview of our non-malleable time-lock puzzle construction (in the random oracle model) and its proof of security. Then in Section 2.2, we overview our construction of publicly verifiable (and non-malleable) time-lock puzzles from repeated squaring. Finally in Section 2.3, we discuss how our non-malleable time-lock puzzle constructions can be used for fair multi-party coin flipping with various desirable properties.

We start by recalling the definition of TLPs, as necessary to give an overview of our techniques. A TLP consists of two algorithms $(\mathsf{Gen}, \mathsf{Sol})$. Gen is a probabilistic procedure that takes as input an embedded solution s and a time parameter t, and outputs a puzzle z. Sol is a deterministic procedure that on input a puzzle z for time bound t, outputs a solution in depth (or parallel time) roughly t. We note that TLPs can be thought of as a fine-grained analogue to commitments where "hardness" of the puzzle means that the puzzles are hiding against distinguishers of depth less than t. On the other hand, hiding *can be broken* in depth t (using Sol). Additionally, we require that Sol always finds the correct underlying solution s for a puzzle z. This corresponds to perfect binding in the language of commitments.

2.1 Non-Malleability for Time-Lock Puzzles

In this section, we overview our non-malleable time-lock puzzle construction in the random oracle model (for the plain model construction, we refer the reader to the overview in [15], as the main ideas are the same). Our construction relies on any time-lock puzzle TLP and a common random oracle $\mathcal{O}$. We now describe our non-malleable TLP, which we denote nmTLP. In order to generate a puzzle for a solution s that can be broken in time t, $\mathsf{nmTLP.Gen}$ uses randomness r and feeds $s \| r$ into the random oracle to get a string r_{tlp}. It then uses $\mathsf{TLP.Gen}$ to create a puzzle with difficulty t for $s \| r$ using randomness r_{tlp}. That is,

$$\mathsf{nmTLP.Gen}(t, s; r) := \mathsf{TLP.Gen}(t, s \| r; \mathcal{O}(s \| r)).$$

Note that in order to solve the puzzle output by $\mathsf{nmTLP.Gen}$, it suffices to just solve the puzzle generated using $\mathsf{TLP.Gen}$, which takes time t. In other words, $\mathsf{nmTLP.Sol}(t, z)$ simply computes $s \| r = \mathsf{TLP.Sol}(t, z)$ and outputs s. In fact, the solver can even check to make sure that the solutions s is valid by checking that $s = \mathsf{TLP.Gen}(t, s; r)$.

We note that our construction is conceptually similar to the Fujisaki-Okamoto (FO) transformation [21] for transforming CPA-secure encryption to CCA-secure encryption using a random oracle. However, as we will see below, our proof is substantially different. In particular, the FO transformation achieves unbounded CCA security, which we show is impossible in our setting!

Hardness. To show the hardness of nmTLP relative to a random oracle, we rely on the hardness of TLP in the plain model, against attackers of depth much less than t. At a high level, we show that breaking the hardness of nmTLP requires either guessing the randomness r used to generate the randomness $r_{\mathsf{tlp}} = \mathcal{O}(s||r)$ for the underlying puzzle, or directly breaking the hardness of TLP, both of which are infeasible for bounded attackers. To formalize this, we consider any depth-bounded distinguisher $\mathcal{D}^{\mathcal{O}}$, who receives as input a nmTLP puzzle z corresponding to solution s_0 or s_1 and distinguishes the two cases with non-negligible probability. By construction, z actually corresponds to a TLP puzzle for $s_0||r_0$ or $s_1||r_1$, so we would like to use $\mathcal{D}$ to construct a distinguisher against the hardness of TLP.

We first note that if $\mathcal{D}$ never makes a query to $\mathcal{O}$ containing the randomness r_b underlying z, then we can simulate $\mathcal{O}$ by lazily sampling it in the plain model, and hence use $\mathcal{D}$ as a distinguisher for the hardness of TLP. If $\mathcal{D}$ does make a query containing r_b, then with overwhelming probability it must have received a puzzle corresponding to $s_b||r_b$ (since in this case, r_{1-b} is independent of $\mathcal{D}$ and its input z). Moreover, all of its queries up until that point have uniformly random answers independent of z, so we can simulate them as well, up until receiving this query. Therefore, in both cases, we can carry out this attack in the plain model and rely on the hardness of TLP.

Non-malleability. To show non-malleability of nmTLP, we want to argue that any depth-bounded man-in-the-middle (MIM) attacker $\mathcal{A}$ cannot maul a puzzle z for s (received on the left) to a puzzle $\widetilde{z}$ (output on the right) for a related value $\widetilde{s} \neq s$. At a high level, whenever $\mathcal{A}$ changes the underlying value s to $\widetilde{s}$, then the output of the random oracle on $\widetilde{s}$ is now uniformly random and independent of z. Indeed, we show that for any fixed puzzle $\widetilde{z}$ and a value $\widetilde{s}$, a randomly generated puzzle for $\widetilde{s}$ will not be equal to $\widetilde{z}$ with high probability (otherwise we show how to break the hardness of TLP). So, intuitively, the only way to generate a *valid* puzzle $\widetilde{z}$ for $\widetilde{s}$ is to "know" the underlying value $\widetilde{s}$, but hardness intuitively implies that no depth-bounded adversary can "know" s.

We formalize this intuition by a hybrid argument to show that the MIM distribution $\widetilde{s} \leftarrow \mathsf{mim}_{\mathcal{A}}(t, s)$ is indistinguishable from $\mathsf{mim}_{\mathcal{A}}(t, 0)$. At a high level, we first replace the inefficient distribution $\mathsf{mim}_{\mathcal{A}}(t, s)$ by a low-depth circuit $\mathcal{B}$. At this point, we want to use the hiding property to indistinguishably swap the puzzle to 0, so the hybrid is now unrelated to s. We describe the key ideas for these hybrids below.

For the first hybrid, the key insight is that we can compute $\mathsf{mim}_{\mathcal{A}}(t, s)$ *in low depth* using an algorithm $\mathcal{B}$ by simply looking at the oracle queries made by $\mathcal{A}$. In this sense, we are relying on the extractability property of random oracles to say that $\mathcal{A}$ must know any valid value $\widetilde{s}$ it generates a puzzle for. Specifically, let $\widetilde{z}$ be the output of $\mathcal{A}$. For every query $(s_i\|r_i)$ that $\mathcal{A}$ makes to $\mathcal{O}$, $\mathcal{B}$ outputs s_i if $\widetilde{z} = \mathsf{nmTLP}(t, s_i\|r_i; \mathcal{O}(s_i\|r_i))$. If there are no such queries, $\mathcal{B}$ outputs $\perp$. $\mathcal{B}$ requires depth comparable to the depth of $\mathcal{A}$ since all of these checks can be done in parallel. Furthermore, the output of $\mathcal{B}$ is indistinguishable from the true output given the above observation that $\mathcal{A}$ cannot output a valid puzzle for a value it doesn't query.

For the next hybrid, we would like to indistinguishably replace the underlying puzzle for s with a puzzle for 0, which would suffice to show non-malleability. Because $\mathcal{B}$ is low-depth, it seems that we should be able to use the hiding property of nmTLP to say that the output of $\mathcal{B}$ does not depend on the underlying value s. Specifically, we want to conclude that if the output of $\mathcal{B}$ (who outputs many bits) is statistically far when the underlying value is s versus 0, then there exists a distinguisher (who outputs a single bit) that can distinguish puzzles for s and 0. Towards this claim, we show how to "flatten" any (possibly unbounded) distinguisher $\mathcal{D}$ who distinguishes between the output of $\mathcal{B}$ in the case where the underlying value is s versus 0. Specifically, we encode the truth table of $\mathcal{D}$ as a low-depth distinguishing circuit of size roughly $2^{|s|}$ to make this reduction go through. As a result, we need to rely on a sub-exponentially security of the underlying TLP when $|s| = \lambda$. Namely, the underlying TLP cannot be broken by sub-exponential sized circuits with depth much less than t. However, when $|s| \in O(\log \lambda)$, we only need to rely on *polynomial* security of the underlying TLP.

Impossibility of fully concurrent non-malleability. Ideally, we would like to achieve fully concurrent non-malleability, meaning that any MIM attacker that receives any polynomial n number of puzzles on the left cannot maul them to n puzzles for related values. However, we show that this is impossible to achieve.

Consider an arbitrary TLP for a polynomial time bound t. We construct a MIM attacker $\mathcal{A}$ that receives only a single puzzle z on the left with solution s where the length of z is L. Then, $\mathcal{A}$ can split z into L bits and output a puzzle on the right for each bit *of the puzzle* z. Then, the values underlying the puzzles output by $\mathcal{A}$ when viewed together yield z, which is related to the value s! More formally, there exists a polynomial time distinguisher that solves the puzzle z in polynomial time t and can distinguish $\mathcal{A}$'s output in the case when it receives a puzzle for s or an unrelated value, say 0.

This implies that for any n which is greater than the size of a puzzle, the TLP cannot be non-malleable against MIM attackers who output at most n puzzles on the right. At a high level, the impossibility follows from the fact that hardness does not hold against arbitrary polynomial-time distinguishers (which usually *is* the case for hiding of standard commitments).

Despite this impossibility, we show that we actually *can* achieve concurrent non-malleability against a *specific class of distinguishers* in the non-malleability game. We refer to this notion as concurrent *functional* non-malleability.

Achieving concurrent functional non-malleability. In many applications, we only need a form of non-malleability to hold with respect to certain classes of functions. For example, in our application to coin flipping, we only need that a puzzle z with solution s cannot be mauled to a set of puzzles $\widetilde{z_1}, \ldots, \widetilde{z_n}$ with underlying values $\widetilde{s_1}, \ldots, \widetilde{s_n}$ such that $\bigoplus_{i \in [n]} \widetilde{s_i}$ "depends on" s. With this in mind, we define a concurrent functional non-malleability with respect to a class of functions $\mathcal{F}$. We say that a TLP satisfies *functional* non-malleability for a class $\mathcal{F}$ if the output of $f(\mathsf{mim}_{\mathcal{A}}(t, s))$ is indistinguishable from $f(\mathsf{mim}_{\mathcal{A}}(t, 0))$ for any $f \in \mathcal{F}$, which also naturally generalizes to the concurrent setting. We note that functional non-malleability for a class $\mathcal{F}$ actually implies standard non-malleability whenever the class $\mathcal{F}$ contains the identity function, so functional non-malleability generalizes the standard notion of non-malleability.

Going back to the proof of standard (non-concurrent) non-malleability for our construction nmTLP, we observe that the security we need for the underlying time-lock puzzle we use depends on $2^{|s|}$ where $|s|$ is the size of the puzzle solutions. Specifically, given any distinguisher in the non-malleability that had input of size $|s|$, we were able to construct a distinguisher for hardness of size $2^{|s|}$. In fact, this exact same proof works in the context of concurrent functional non-malleability for functions f that have *low depth* and bounded output length

m. We require f to be low depth so the reduction constitutes a valid attack against hardness, and then we only require security proportional to 2^m!

We briefly discuss how our nmTLP construction works for concurrent functional non-malleability for the class $\mathcal{F}_m$ of function with low depth and output length m. Specifically, for every m, we define a scheme nmTLP_m assuming that TLP is secure against attackers of size roughly 2^m. Because TLP requires security against 2^m size attackers, our construction nmTLP_m also only achieves security against 2^m size attackers. As such, our $\mathsf{nmTLP.Gen}$ algorithm needs to use at least $\Omega(m + \lambda)$ bits of randomness (otherwise an attacker could cycle through all choices of randomness to break security). Recall that $\mathsf{nmTLP}_m.\mathsf{Gen}$ with randomness r outputs a puzzle using $\mathsf{TLP.Gen}$ with solution $s\|r$. As a result, if we want to support solutions of size $|s|$ in nmTLP_m, we need our underlying TLP to support solutions of size $O(|s| + m + \lambda)$. By correctness, this implies that our schemes outputs puzzles of size roughly $O(|s| + m + \lambda)$.

Bounded concurrent non-malleability. Our construction of time-lock puzzles for concurrent functional non-malleability can also be seen as a construction for bounded concurrent (plain) non-malleability. Specifically, consider the case where the MIM attacker outputs at most n puzzles on the right. We can think of this as functional non-malleability where the low depth function is simply identity on $n \cdot |s|$ bits. From the above discussion, this implies a protocol assuming a TLP with security against size $2^{n \cdot |s|}$ attackers, with puzzles of size roughly $O(n \cdot |s| + \lambda)$.

Security in the auxiliary-input random oracle model. Finally, we note that the most of our constructions and formal proofs are in the auxiliary-input random oracle model (AI-ROM) introduced by Unruh [47]. In this model, the non-uniform attacker is allowed to depend arbitrarily on the random oracle, so there is no attacker-independent non-uniform advice. At a high level, we use the result from [47] to conclude that the view of any bounded-size MIM attacker $\mathcal{A}$ with oracle access to $\mathcal{O}$ (where $\mathcal{A}$ may depend arbitrarily on $\mathcal{O}$) is indistinguishable the view of $\mathcal{A}$ with access to a "lazily sampled" oracle $\mathcal{P}$ that is fixed at a set of points F (which depend on $\mathcal{A}$). Formally, in the non-malleability analysis, we switch to an intermediate hybrid where the MIM attacker has access to a partially fixed, lazily sampled oracle $\mathcal{P}$. Then, because the MIM attacker $\mathcal{A}$ must maul honestly generated puzzles that have high entropy, we show that it is necessary for $\mathcal{A}$ to query the oracle $\mathcal{P}$ outside the fixed set of points F. From this, we carefully show that a similar analysis follows as discussed above for the ROM.

2.2 Publicly Verifiable Time-Lock Puzzles

We observe that the non-malleable time-lock puzzle construction nmTLP we described above has a very natural—yet incomplete—public verifiability property. Solving a puzzle yields both the solution s and the randomness r use to generate that puzzle. As such, anyone who solves a valid puzzle can send the opening r to another party, and convince them that s is the unique valid solution to the puzzle. However, we emphasize that this only works for *valid* puzzles and solutions.

Consider the following problematic scenario for our nmTLP construction. Suppose a party "commits" to a value via a puzzle z and refuses to open the commitment. As we said before, if z is a valid puzzle, any party can solve the puzzle, get the solution s and an opening r that proves that s is the unique solution. What if the puzzle corresponds to *no solution*? We refer to this scenario by saying that the puzzle corresponds to the solution $\bot$. In this case (by definition), there is no solution s and opening r for any such that $z = \mathsf{Gen}(t, s; r)$. Anyone who solve the invalid puzzle—which requires a lot of computational power—*will* be able to conclude that the puzzle is malformed, but they will not be able to convince anyone else that this is the case. Ideally, we would have a time-lock puzzle where Sol additionally outputs a publicly verifiable proof π that the solution it computes is correct, even if the solution may be $\bot$! We refer to such a time-lock puzzle as a *publicly verifiable* time-lock puzzle. We next discuss the definition and our construction of publicly verifiable time-lock puzzles.

Defining public verifiability. More formally, a publicly verifiable time-lock puzzle consists of algorithms $(\mathsf{Gen}, \mathsf{Sol}, \mathsf{Verify})$. As with normal time-lock puzzles, $\mathsf{Gen}(t, s)$ outputs a puzzle z. The algorithm $\mathsf{Sol}(t, z)$ outputs the solution s as well as a proof π that it computed s correctly. Finally $\mathsf{Verify}(t, z, (s, \pi))$ checks that s is indeed the correct solution for the puzzle z (corresponding to $\mathsf{Sol}(t, z)$), using the proof π. In addition to $(\mathsf{Gen}, \mathsf{Sol})$ being a valid time-lock puzzle, we require that Sol and Verify constitute a sound non-interactive argument. In fact, we require a very strong notion of soundness. We need it to be the case that even for maliciously chosen puzzles that have no solution, the time-lock puzzle is still sound—even against the adversary that generated the malformed puzzle. In other words, we require that no attacker can compute a puzzle z, a value s', and a proof π' such that $\mathsf{Verify}(t, z, (s', \pi'))$ accepts yet s' is not the value s computed by $\mathsf{Sol}(t, z)$, which may be $\bot$.

Ideally, we would want a publicly verifiable time-lock puzzle that requires *no setup*. We instead consider a weak form of setup which we refer to as the All-But-One-string (ABO-string) model. In this model, Sol and Verify additionally take as input a string $\mathsf{mcrs} = (\mathsf{crs}_1, \ldots, \mathsf{crs}_n) \in (\{0, 1\}^\lambda)^n$, and we require that soundness holds as long as one of the values of crs_i is sampled uniformly (without necessarily knowing which one); this is why we refer to it as the all-but-one string model. We note that in multi-party protocols, the ABO-string model is realistic as each participant $i \in [n]$ can post a value for crs_i. Then, we require soundness to hold as long as one participant is honest, which is a reasonable assumption in this multi-party setting.

Constructing publicly verifiable time-lock puzzles. Our construction of a publicly verifiable time-lock puzzle follows the blueprint of Rivest, Shamir, and

Wagner [45] for constructing time-lock puzzles from repeated squaring. Namely, we use the output of a sequential function (repeated squaring in a suitable group) essentially as one-time pad to mask the value underlying the time-lock puzzle. As in [45], we require that the sequential function has a trapdoor so that puzzles can be generated efficiently. Unlike [45], we additionally require that the sequential function is publicly verifiable to enable publicly verifiability for the time-lock puzzle. Finally, we apply the non-malleability transformation described above to achieve full public verifiability. In what follows, we describe each of these steps in more detail.

For the underlying sequential function, we use what we call a *strong trapdoor verifiable delay function* (VDF). A VDF (introduced by Boneh et al. [7]) is a publicly verifiable sequential function that can be computed in time t but not much faster, even with lots of parallelism. A trapdoor VDF (formalized by Wesolowski [50]) additionally has a trapdoor for quick evaluation. We require a trapdoor VDF in the ABO-string model that satisfies additional properties required by our application. While the properties we define—and achieve—are heavily tailored towards our application, we believe some of the techniques may be of independent interest. More specifically, a strong trapdoor VDF comes with a Sample algorithm to generate inputs for an evaluation algorithm Eval. We emphasize that, even in the ABO-string model, Sample is independent of any form of setup. Previous definitions of VDFs require the proof to be sound with probability over an *honestly sampled* input. In contrast, we require that the proof is sound for any maliciously chosen input that is in the support of the Sample algorithm. We note that this property is satisfied by a variant of Pietrzak's VDF [44] based on repeated squaring. At a high level, this is because Pietrzak's VDF is sound (at least in the random oracle model) for any group of unknown order where no adversary can find a group of low order (see e.g.[8] for further discussion), so by using any RSA group with no low order elements (as in [44]), the proof is sound even if the group is maliciously chosen (yet still a valid RSA group), which gives the strong property we need. We note that the proof of soundness for our strong trapdoor VDF in the ABO-string and auxiliary-input random oracle model follows by a similar argument to that of [44] in the (plain) random oracle model after applying Unruh's result [47].

Next, we construct what we refer to as a *one-sided* publicly verifiable time-lock puzzle in the ABO-string model by using the strong trapdoor VDF in the RSW-style construction described above. By one-sided, we mean that completeness and soundness hold only for puzzles in the support of Gen (again, we emphasize that this is in contrast to a randomly sampled puzzle). Then, our full construction applies our non-malleability transformation to a one-sided publicly verifiable time-lock puzzle. We already argued that the non-malleability transformation provides a form of public verifiability for puzzles z in the support of Gen. Namely, anyone can prove to another party that a valid puzzle z has a solution s, but the proof may not be sound when trying to prove that a puzzle has no solution. However, we next show that if the underlying puzzle satisfies one-sided public verifiability, then the resulting (non-malleable) publicly verifiable TLP is sound for any $z \in \{0,1\}^*$ (possibly not in the support of Gen).

Proof of full public verifiability. Let $(\mathsf{Gen}, \mathsf{Sol}, \mathsf{Verify})$ be the TLP resulting from applying our non-malleability transformation to a one-sided PV TLP $(\mathsf{Gen}_{\mathsf{tlp}}, \mathsf{Sol}_{\mathsf{tlp}}, \mathsf{Verify}_{\mathsf{tlp}})$. Consider any puzzle $z \in \{0,1\}^*$. If z is in the support of Gen, we want to ensure that no one can prove that $s' = \bot$ is a valid solution. At the same time, if z is not in the support of Gen, we want to ensure that no one can prove that $s' \neq \bot$ is a valid solution.

When we run $\mathsf{Sol}(t, z)$, we first run $\mathsf{Sol}_{\mathsf{tlp}}(t, z)$ and get a solution $s_{\mathsf{tlp}} = \hat{s} \| \hat{r}$ with a proof π_{tlp}. If $\hat{r}$ is a valid opening for the proposed solution $\hat{s}$, then Sol can simply output the solution $s = \hat{s}$ and the proof $\pi = \hat{r}$. If $\hat{r}$ is not a valid opening for $\hat{s}$, Sol must output $\bot$ and a proof π that this is the case. We set $\pi = (s_{\mathsf{tlp}}, \pi_{\mathsf{tlp}})$, which intuitively gives anyone else a way to "shortcut" the computation of $\mathsf{Sol}_{\mathsf{tlp}}$.

Now suppose that an adversary tries to falsely convince you that a puzzle z with no solution has a solution $s' \neq \bot$ using a proof $\pi' = r'$. To do so, it must be the case that r' is a valid opening for s' with respect to Gen. But if that were the case, then z *would have a solution*, in contradiction.

On the other hand, suppose that an adversary tries to falsely convince you that a puzzle z with solution s has no solution, i.e. $s' = \bot$, using a proof $\pi' = (s'_{\mathsf{tlp}}, \pi'_{\mathsf{tlp}})$. Since z has a solution, it means that z is in the support of $\mathsf{Gen}_{\mathsf{tlp}}$. By one-sided public verifiability, this means that π'_{tlp} is a valid proof that $s'_{\mathsf{tlp}} = \hat{s} \| \hat{r}$ is the correct solution to z with respect to $\mathsf{Gen}_{\mathsf{tlp}}$. So if $\hat{r}$ is not a valid opening for $\hat{s}$ with respect to Gen, we know the adversary must be lying. In other words, the only way the adversary can cheat is by cheating in the underlying one-sided PV TLP on a puzzle z in the support of $\mathsf{Gen}_{\mathsf{tlp}}$.

Discussion of our non-malleable PV TLP. We note that the publicly verifiable time-lock puzzle we described above can be made to satisfy the same non-malleability guarantees as we discuss in Section 2.1 (as we construct it using the same transformation but with a specific underlying time-lock puzzle). Thus, assuming the repeated squaring assumption, we get a publicly verifiable time-lock puzzle that satisfies concurrent function non-malleability for any class of low depth functions $\mathcal{F}_m$ with output length m. Our construction is in the ABO-string model, and we prove security in the auxiliary-input random oracle model (which is needed for soundness of the strong trapdoor VDF in the ABO-string model in addition to the non-malleability transformation). This model is reasonable for our practical applications to multi-party protocols, as we will see below. Due to the fact that this is a non-black box construction, we note that it does not apply to our non-malleable TLP construction in the plain model.

We also note that our explicit repeated squaring assumption states that repeated squaring in RSA groups for n-bit integers cannot be sped up even by adversaries of size roughly 2^m. The repeated squaring assumption is closely related to the assumption on factoring (which has recently been formalized in different generic models by the works of [46, 28]). The current best known algorithms for factoring run in time at least $2^{n^{1/3}}$. In the case where $m \in O(\log \lambda)$, for example, we only require that polynomial-size attackers cannot speed up repeated squaring, which is a relatively mild assumption. In the case where m is larger, say $m = \lambda$, then we need to choose n to be at least λ^3 (based on known algorithms for factoring). This gives an example of the various trade-offs we get for the security and efficiency of our construction depending on the class of low depth functions $\mathcal{F}_m$ that we want non-malleability for.

2.3 Fair Multi-Party Protocols

We will focus on coin flipping for concreteness, and note that for auctions the ideas are similar. We give a protocol in auxiliary-input random oracle model, and one in the plain model, depending on which non-malleable TLP construction we use to instantiate it (which result in different guarantees). Here, we describe our random oracle protocol, which captures the main ideas and various properties we can achieve.

At a high level, the coin flipping protocol is very simple. Each party chooses a random bit and publishes a time-lock puzzle that encodes the chosen bit. After all puzzles are published, each party opens their puzzle by revealing the bit that they used as well as the randomness used to generate the puzzle. Any puzzle that is not opened can be "solved" after a moderately large amount of time t. Once all puzzles have been opened, the agreed upon bit (i.e., the output of the protocol) is the XOR of all revealed bits. The above protocol template is appealing because it naturally satisfies optimistic efficiency: if all parties are honest and open their puzzles, the protocol terminates immediately. When using time-lock puzzles which are both non-malleable (as discussed in Section 2.1) and publicly verifiable (as discussed in Section 2.2), we achieve the following highly desirable properties:

- **Fairness:** No malicious party can bias the output of the protocol.
  This crucially relies on non-malleability for the underlying time-lock puzzle. For a protocol with n participants, we need the time-lock puzzle to satisfy n-concurrent non-malleability. This guarantees that as long as one party is honest, the output of the protocol will be (at least statistically close to) a uniformly random bit.
- **Unbounded participants:** Anyone can participate in the protocol.
  This property might come as a surprise since we show fully concurrent non-malleability is impossible to achieve. However, we emphasize that our time-lock puzzle achieves fully concurrent *functional* non-malleability for the XOR function. This allows us to deal with any a priori unbounded number of participants, which is important in many decentralized and distributed settings.
- **Public verifiability:** Only one party needs to solve each unopened puzzle, and can provide a publicly verifiable proof that it solved it correctly.
  This follows immediately by the public verifiability property we achieve for the underlying time-lock puzzle. Without this property, any unopened puzzles may need to be solved by every party that want to know the output of

the protocol, which is prohibitively expensive. However, public verifiability instead opens up the application to *any* party, not even involved in the protocol. Furthermore, this work can even be delegated to an external server since trust is guaranteed by the attached proof.

We note that our non-malleable and publicly verifiable time-lock puzzle is defined in the All-But-One-string (ABO-string) model, which is required for public verifiability. To implement this model, we have each participant i publish a fresh random string $\mathsf{crs}_i \leftarrow \{0,1\}^\lambda$ in addition to its puzzle z_i. Then, whenever some party tries to solve (or verify) a puzzle, it puts all of the random strings together as a multi-common random string $\mathsf{mcrs} = (\mathsf{crs}_1, \dots, \mathsf{crs}_n)$ from all n participants, and uses this for the publicly verifiable proof. As long as a single party is honest and publishes a random string crs_i independent of all other participants, then the publicly verifiable proof system will be sound.

Simulation-based fairness. Finally, we discuss how fairness in the above protocol can be strengthened to a simulation-style definition. Consider running our protocol to get a value s, where s is the XOR of bits underlying the adversary's and honest players' time-lock puzzles. In our simulation-secure protocol, we will set the output to $\mathcal{O}(s)$ where $\mathcal{O}$ is a programmable random oracle. This enables a simulator running in polynomial time to solve the adversary's puzzles and program $\mathcal{O}(s)$ to the desired output value. It then suffices to show that the adversary $\mathcal{A}$ does not detect this change in the oracle, meaning that $\mathcal{A}$ does not query s before publishing its time-lock puzzles. We observe that if $\mathcal{A}$ does indeed query s, it implies an adversary against the game-based fairness of our protocol, that runs $\mathcal{A}$ to get s and outputs a TLP to s along with $\mathcal{A}$'s puzzles, thus biasing the output to $s \oplus s = 0$. We note that this shows that game-based fairness for a standard commit-and-reveal style protocol (with TLPs instead of commitments) can be generically transformed into a simulation-secure protocol by feeding the output into a programmable random oracle.

3 Preliminaries

We first define time-lock puzzles without any additional properties.

Definition 3.1. *Let $B \colon \mathbb{N} \to \mathbb{N}$. A B-*hard time-lock puzzle (TLP) *is a tuple* $(\mathsf{Gen}, \mathsf{Sol})$ *with the following syntax:*

- $z \leftarrow \mathsf{Gen}(1^\lambda, t, s)$**:** *A PPT algorithm that on input a security parameter $\lambda \in \mathbb{N}$, a difficulty parameter $t \in \mathbb{N}$, and a solution $s \in \{0,1\}^\lambda$, outputs a puzzle $z \in \{0,1\}^*$.*
- $s = \mathsf{Sol}(1^\lambda, t, z)$**:** *A deterministic algorithm that on input a security parameter $\lambda \in \mathbb{N}$, a difficulty parameter $t \in \mathbb{N}$, and a puzzle $z \in \{0,1\}^*$, outputs a solution $s \in (\{0,1\}^\lambda \cup \{\bot\})$.*

We require $(\mathsf{Gen}, \mathsf{Sol})$ *to satisfy the following properties.*

- **Correctness:** *For every $\lambda, t \in \mathbb{N}$, solution $s \in \{0,1\}^\lambda$, and $z \in \mathrm{Supp}\big(\mathsf{Gen}(1^\lambda, t, s)\big)$, it holds that $\mathsf{Sol}(1^\lambda, t, z) = s$.*
- **Efficiency:** *There exist a polynomial p such that for all $\lambda, t \in \mathbb{N}$, $\mathsf{Sol}(1^\lambda, t, \cdot)$ is computable in time $t \cdot p(\lambda, \log t)$.*
- **B-Hardness:** *There exists a positive polynomial function α such that for all functions T and non-uniform distinguishers $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ satisfying $\alpha(\lambda) \leq T(\lambda) \in B(\lambda) \cdot \mathrm{poly}(\lambda)$, $\mathsf{size}(\mathcal{A}_\lambda) \in B(\lambda) \cdot \mathrm{poly}(\lambda)$, and $\mathsf{depth}(\mathcal{A}_\lambda) \leq T(\lambda)/\alpha(\lambda)$ for all $\lambda \in \mathbb{N}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, and $s, s' \in \{0,1\}^\lambda$,*

  $$\left|\Pr\left[\mathcal{A}_\lambda(\mathsf{Gen}(1^\lambda, T(\lambda), s)) = 1\right] - \Pr\left[\mathcal{A}_\lambda(\mathsf{Gen}(1^\lambda, T(\lambda), s')) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

  where the probabilities are over the randomness of Gen and $\mathcal{A}_\lambda$.

When $B(\lambda) \in \mathrm{poly}(\lambda)$, we say that the TLP is polynomially-hard.

In the above definition, we assume for simplicity that the solutions s are λ-bits long. We can naturally generalize this to consider the case where solutions have some specified length $L(\lambda)$. We emphasize that the notion of B-hardness above suffices to capture both polynomial security and sub-exponential security, as it captures hardness against adversaries of size $B(\lambda)$, up to polynomial factors.

Non-malleable time-lock puzzles. To formalize non-malleability in the context of time-lock puzzles, we introduce a Man-In-the-Middle (MIM) adversary. Because time-lock puzzles are designed to be broken in some depth t, we restrict our MIM adversary to have at most depth $t/\alpha(\lambda)$ for a function α denoting the advantage of the adversary. Furthermore, we allow for concurrent MIM adversaries that possibly interact with many senders and receivers at the same time.

Definition 3.2 (MIM Adversaries). *Let $n_L, n_R, B_{\mathsf{nm}}, \alpha, T \colon \mathbb{N} \to \mathbb{N}$. An $(n_L, n_R, B_{\mathsf{nm}}, \alpha, T)$-*Man-In-the-Middle (MIM) adversary *is a non-uniform algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ satisfying $\mathsf{depth}(\mathcal{A}_\lambda) \leq T(\lambda)/\alpha(\lambda)$ and $\mathsf{size}(\mathcal{A}_\lambda) \in B_{\mathsf{nm}}(\lambda) \cdot \mathrm{poly}(\lambda)$ for all $\lambda \in \mathbb{N}$ that receives $n_L(\lambda)$ puzzles on the left and outputs $n_R(\lambda)$ puzzles on the right.*

We next define the MIM distribution, which corresponds to the values underlying the puzzles output by the MIM adversary. To capture adversaries that simply forward one of the puzzles on the left to a receiver on the right, we set the value for any forwarded puzzle to be $\perp$.

Definition 3.3 (MIM Distribution). *Let $n_L, n_R, B_{\mathsf{nm}}, \alpha, T \colon \mathbb{N} \to \mathbb{N}$. Let $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ be an $(n_L, n_R, B_{\mathsf{nm}}, \alpha, T)$-MIM adversary. For any $\lambda \in \mathbb{N}$ and $\vec{s} = (s_1, \ldots, s_{n_L(\lambda)}) \in (\{0,1\}^\lambda)^{n_L(\lambda)}$, we define the distribution*

$$(\widetilde{s}_1, \ldots, \widetilde{s}_{n_R(\lambda)}) \leftarrow \mathsf{mim}_{\mathcal{A}}(1^\lambda, T(\lambda), \vec{s})$$

as follows. $\mathcal{A}_\lambda$ *receives puzzles* $z_i \leftarrow \mathsf{Gen}(1^\lambda, T(\lambda), s_i)$ *for all* $i \in [n_L(\lambda)]$ *and outputs puzzles* $(\widetilde{z}_1, \ldots, \widetilde{z}_{n_R(\lambda)})$*. Then for each* $i \in [n_R(\lambda)]$*, we define*

$$\widetilde{s}_i = \begin{cases} \bot & \text{if there exists a } j \in [n_L(\lambda)] \text{ such that } \widetilde{z}_i = z_j, \\ \mathsf{Sol}(1^\lambda, T(\lambda), \widetilde{z}_i) & \text{otherwise.} \end{cases}$$

Intuitively, a time-lock puzzle is non-malleable if the MIM distribution of a bounded depth attacker does not depend on the solutions underlying the puzzles it receives on the left. We formalize this definition below.

Definition 3.4 (Concurrent Non-malleable). *Let* $n_L, n_R, B_{\mathsf{nm}} \colon \mathbb{N} \to \mathbb{N}$*. A time-lock puzzle is* (n_L, n_R)-concurrent non-malleable against adversaries of size B_{nm} *if there exists a positive polynomial* α *such that for every function* T *with* $\alpha(\lambda) \leq T(\lambda) \in B_{\mathsf{nm}}(\lambda) \cdot \mathrm{poly}(\lambda)$ *for all* $\lambda \in \mathbb{N}$*, and every* $(n_L, n_R, B_{\mathsf{nm}}, \alpha, T)$*-MIM adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$*, the following holds.*

For any distinguisher $\mathcal{D}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$ *and* $\vec{s} = (s_1, \ldots, s_{n_L(\lambda)}) \in (\{0,1\}^\lambda)^{n_L(\lambda)}$*,*

$$\left| \Pr\left[\mathcal{D}(\mathsf{mim}_{\mathcal{A}}(1^\lambda, T(\lambda), \vec{s})) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{mim}_{\mathcal{A}}(1^\lambda, T(\lambda), (0^\lambda)^{n_L(\lambda)})) = 1\right] \right|$$
$$\leq \mathsf{negl}(\lambda).$$

When $B_{\mathsf{nm}}(\lambda) = 1$*, we say the TLP is* (n_L, n_R)-concurrent non-malleable. *When this only holds against non-uniform PPT distinguishers* $\mathcal{D}$*, we say that the time-lock puzzle is* computationally (n_L, n_R)-concurrent non-malleable.

Relation to non-malleable commitments. When defining non-malleability for TLPs, a natural attempt is to view TLPs as commitments, and give a definition analogous to non-malleable commitments. This is usually formalized as either non-malleability with respect to commitment, or non-malleability with respect to extraction. The former notion requires that no man-in-the-middle adversary can maul a commitment z to s into a commitment $\widetilde{z}$ whose unique underlying value is related to s, whereas the latter notion requires that $\mathcal{E}(\widetilde{z})$ is unrelated to s, where $\mathcal{E}$ is a given extractor. When $\mathcal{E}$ has the guarantee that it outputs the committed value on valid commitments and $\bot$ on invalid ones, these notions are equivalent. However, when considering extractors that may output arbitrary values when given invalid commitments, these notions are incomparable in general. In the context of time-lock puzzles, we observe that Sol is the natural extractor for Gen, and moreover that non-malleability should capture adversaries that maul a puzzle into one that solves to a related value. Therefore, our definition above is analogous to non-malleability with respect to extraction, where Sol is the extractor.

Next, we consider standard variants for the definition of non-malleable above.

Definition 3.5. *We say the a TLP satisfies the following non-malleability properties when Definition 3.4 holds against* $(n_L, n_R, B_{\mathsf{nm}}, \alpha, T)$*-MIM adversaries for the following settings of* n_L *and* n_R*:*

- fully concurrent non-malleable *if the definition holds against any* $n_L, n_R \in \mathrm{poly}(\lambda)$*,*
- one-many non-malleable *if the definition holds for any* $n_R(\lambda) \in \mathrm{poly}(\lambda)$ *and* $n_L = 1$*,*
- n-concurrent non-malleable *if the definition holds for* $n_L = n_R = n$*,*
- one-n non-malleable *for* $n_L(\lambda) = 1$ *and* $n_R = n$*,*
- *and simply* non-malleable *(not concurrent) for* $n_L(\lambda) = n_R(\lambda) = 1$*.*

4 Non-Malleable Time-Lock Puzzles

We start by defining the notion of functional non-malleability for time-lock puzzles. Then, we give the transformation from any time-lock puzzle to one that satisfies concurrent functional non-malleability for depth bounded functions, in the auxiliary input random oracle model, and discuss how this result implies a time-lock puzzle satisfying bounded concurrent (standard) non-malleability.

Functional non-malleability. In the following definition, we focus on the case of unbounded concurrency, but note that can be defined for restricted cases as in Definition 3.5.

Definition 4.1 (Concurrent Functional Non-malleable). *Let* $B_{\mathsf{nm}}, L \colon \mathbb{N} \to \mathbb{N}$*, and* $(\mathsf{Gen}, \mathsf{Sol})$ *be a time-lock puzzle for messages of length* $L(\lambda)$*. Let* $\mathcal{F}$ *be a class of functions of the form* $f \colon (\{0,1\}^{L(\lambda)})^* \to \{0,1\}^*$*. We say that* $(\mathsf{Gen}, \mathsf{Sol})$ *is* concurrent functional non-malleable for $\mathcal{F}$ against B_{nm}-size adversaries *if for any function* $f \in \mathcal{F}$ *and polynomial* n*, there exists a polynomial* α *such that for every function* T *with* $\alpha(\lambda) \leq T(\lambda) \in B_{\mathsf{nm}}(\lambda) \cdot \mathrm{poly}(\lambda)$ *for all* $\lambda \in \mathbb{N}$*, every* $(n, n, B_{\mathsf{nm}}, \alpha, T)$*-MIM adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$*, the following holds.*

For any distinguisher $\mathcal{D}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$ *and* $\vec{s} = (s_1, \ldots, s_{n(\lambda)}) \in (\{0,1\}^{L(\lambda)})^{n(\lambda)}$*,*

$$\Big| \Pr\left[\vec{\tilde{s}} \leftarrow \mathsf{mim}_{\mathcal{A}}(1^\lambda, T(\lambda), \vec{s}) : \mathcal{D}(f(\vec{\tilde{s}})) = 1\right] \\ - \Pr\left[\vec{\tilde{s}} \leftarrow \mathsf{mim}_{\mathcal{A}}(1^\lambda, T(\lambda), (0^{L(\lambda)})^{n(\lambda)}) : \mathcal{D}(f(\vec{\tilde{s}})) = 1\right] \Big| \leq \mathsf{negl}(\lambda).$$

When $B_{\mathsf{nm}}(\lambda) = 1$*, we say the TLP is* concurrent functional non-malleable for $\mathcal{F}$*. When the above only holds against non-uniform PPT distinguishers* $\mathcal{D}$*, we say the TLP is* computationally functional non-malleable for $\mathcal{F}$*.*

We note that functional non-malleability for a class $\mathcal{F}$ that contains the identity function id implies standard non-malleability as $\mathcal{D}(\mathsf{id}(\vec{\tilde{s}})) = \mathcal{D}(\vec{\tilde{s}})$.

4.1 Non-Malleable Time-Lock Puzzle Construction

In this section, we give our construction of a fully concurrent functional non-malleable time-lock puzzle for functions with bounded depth and output length. We rely on the following building blocks and parameters.

- A function m denoting the output length for our function non-malleability. We require $m(\lambda) \in \mathrm{poly}(\lambda)$. Throughout this section, where λ is clear from context, we let $m = m(\lambda)$.
- A B_{tlp}-hard time-lock puzzle $\mathsf{TLP} = (\mathsf{Gen}_{\mathsf{tlp}}, \mathsf{Sol}_{\mathsf{tlp}})$ for $B_{\mathsf{tlp}}(\lambda) = 2^{3m}$. We let $\lambda_{\mathsf{tlp}} = \lambda_{\mathsf{tlp}}(\lambda) \in \mathrm{poly}(\lambda, m)$ be the bits of randomness needed for TLP on security parameter λ, for solutions of length $2m + 2\lambda$.
- A class of functions $\mathcal{F}_m$ of the form $f \colon (\{0,1\}^{\lambda})^* \to \{0,1\}^{m(\lambda)}$. We assume that there exists a polynomial d such that for every polynomial n, every function $f \in \mathcal{F}_m$ can be computed in depth $d(\lambda, \log n(\lambda))$ and polynomial size on inputs of length at most $\lambda \cdot n(\lambda)$.
- A random oracle $\mathcal{O} \in \mathsf{RF}^{\lambda_{\mathsf{tlp}}}_{2\lambda+2m}$, where $\mathcal{O}$ on input $(s, r) \in \{0,1\}^{\lambda+(2m+\lambda)}$ outputs a random value $r' \in \{0,1\}^{\lambda_{\mathsf{tlp}}}$.

Our construction $\mathsf{nmTLP}_m = (\mathsf{Gen}, \mathsf{Sol})$ in the random oracle model:

- $z = \mathsf{Gen}^{\mathcal{O}}(1^\lambda, t, s; r)$:
  1. Get $r' = \mathcal{O}(s, r)$.
  2. Output $z = \mathsf{Gen}_{\mathsf{tlp}}(1^\lambda, t, (s||r); r')$.
- $s = \mathsf{Sol}^{\mathcal{O}}(1^\lambda, t, z)$:
  1. Compute $s' = \mathsf{Sol}_{\mathsf{tlp}}(1^\lambda, t, z)$ and parse $s' = s||r$.
  2. If $z = \mathsf{Gen}^{\mathcal{O}}(1^\lambda, t, s; r)$, output s.
  3. If not, output $\bot$.

Theorem 4.2 (Fully Concurrent Functional Non-Malleable TLPs). *Let $m(\lambda) \in \mathrm{poly}(\lambda)$, $B_{\mathsf{hard}}(\lambda) = 2^{m(\lambda)}$, and $B_{\mathsf{tlp}}(\lambda) = 2^{3m(\lambda)}$. Assuming TLP is a B_{tlp}-hard time-lock puzzle, then nmTLP_m is a B_{hard}-hard fully concurrent functional non-malleable time-lock puzzle in the AI-ROM for the class of functions $\mathcal{F}_m$.*

We observe the following corollaries to the above theorem:

- If $m(\lambda) \in O(\log(\lambda))$ then we can simply assume a polynomially-hard TLP.
- For any $m(\lambda) \in \mathrm{poly}(\lambda)$, our theorem follows by assuming a sub-exponentially secure TLP. Specifically, it suffices that there exists a constant $\gamma \in (0, 1)$ such that $B_{\mathsf{tlp}}(\lambda) = 2^{\lambda^\gamma}$, and we can instantiate this with $\lambda_{\mathsf{tlp}} = (\lambda + 3m(\lambda))^{1/\gamma}$ bits of randomness.

We also observe that the above theorem can be used to get n-bounded concurrency for any polynomial n, simply by setting the output length m of the functions in $\mathcal{F}_m$ to $\lambda \cdot n(\lambda)$. Specifically, let f_{id} be the identity function with input and output length $\lambda \cdot n(\lambda)$. Since $f_{\mathsf{id}} \in \mathcal{F}_{\lambda \cdot n(\lambda)}$, a fully concurrent functional non-malleable TLP for $\mathcal{F}_{\lambda \cdot n(\lambda)}$ implies an n-concurrent non-malleable TLP, which gives the following corollary.

Corollary 4.3 (n-Concurrent Non-Malleable TLPs). *Let $n(\lambda) \in \mathrm{poly}(\lambda)$, $B_{\mathsf{hard}}(\lambda) = 2^{\lambda \cdot n(\lambda)}$, and $B_{\mathsf{tlp}}(\lambda) = 2^{3\lambda \cdot n(\lambda)}$. Assuming TLP is a B_{tlp}-hard time-lock puzzle, then $\mathsf{nmTLP}_{\lambda \cdot n(\lambda)}$ is a B_{hard}-hard n-concurrent non-malleable time-lock puzzle in the AI-ROM.*

The proof of Theorem 4.2 is deferred to the full version.

5 Applications to Multi-Party Coin Flipping and Auctions

In this section, we discuss our fair multi-party protocols. We focus on the case of multi-party coin flipping and address auctions in Remark 5.4 below. We note that this section focuses on game-based fairness, and the extension to simulation security is given in the full version.

Our multi-party coin flipping protocol is based generically on any time-lock puzzle. Fairness follows when the time-lock puzzle satisfies concurrent functional non-malleability for the XOR function $f_\oplus$. Specifically, in order to produce L bits of randomness, we need concurrent functional non-malleability for the function $f_\oplus \colon (\{0,1\}^L)^* \to \{0,1\}^L$ that on input $(r_1, \ldots, r_n)$ outputs $\bigoplus_{r_i \neq \perp} r_i$. Our protocol satisfies various additional properties, depending on the time-lock puzzle:

- Given a publicly verifiable time-lock puzzle, the resulting protocol is publicly verifiable. In this setting, our protocol can either be made interactive, or non-interactive.
- If the time-lock puzzle is not publicly verifiable, the resulting protocol is non-interactive, and does not achieve public verifiability.

In what follows, we present our results in the public verifiability setting, and discuss differences with the non-publicly verifiable setting when relevant.

We describe our protocol in a public bulletin board model, where any party may "publish" a message that all other parties will see within some fixed time. Our protocol consists four phases: a commit phase, open phase, force open phase, and output phase. The commit and open phases consist of a single synchronous round of communication where all participating parties publish a message on the bulletin board. The force open phase can be computed by any party, and only needs to be computed by a single (honest) party if the underlying time-lock puzzle is publicly verifiable. Once all puzzles have been opened (or force opened), any party can run the output phase to get the output of the protocol. In the non-interactive version of the protocol, the open phase is omitted and every party runs the force open phase themselves, and uses the resulting values to compute the output of the protocol locally. When we refer to an *honest* participant, we mean a party that runs the protocol as specified, independent of all other participants.

For any $L \colon \mathbb{N} \to \mathbb{N}$, let $(\mathsf{Gen}, \mathsf{Sol}, \mathsf{Verify})$ be a publicly verifiable time-lock puzzle (in the ABO string model) with message length $L(\lambda)$ that satisfies concurrent functional non-malleability for the function $f_\oplus$ (which has output length $L(\lambda)$). We additionally let $\alpha(\lambda)$ be the advantage of any attacker guaranteed by the functional non-malleability of the time-lock puzzle. The protocol takes as common input a security parameter λ and a polynomial time bound $t = T(\lambda)$ that satisfies the following requirements. First, we require that the commit phase takes time less than $T(\lambda)/\alpha(\lambda)$ such that functional non-malleability (and hence hardness) are preserved during the protocol. At the same time, the commit phase needs to be long enough so that all participants can generate and publish their puzzles.

- **Commit phase:** Each participant i samples $s_i \leftarrow \{0,1\}^{L(\lambda)}$ and $r_i, \mathsf{crs}_i \leftarrow \{0,1\}^{\lambda}$, computes $z_i = \mathsf{Gen}(1^{\lambda}, t, s_i; r_i)$, and publishes z_i and crs_i. Let $\mathsf{mcrs} = (\mathsf{crs}_1, \ldots, \mathsf{crs}_n)$. Any puzzle that is a copy of a previously posted puzzle is ignored.
- **Open phase:** Each participant i that published in the commit phase publishes the solution s_i and with an opening r_i.
- **Force open phase:** For each puzzle z_j, if either (a) there is no published solution s_j and opening r_j or (b) if $z_j \neq \mathsf{Gen}(1^{\lambda}, t, s_j; r_j)$, compute and publish $(s_j, \pi_j) \leftarrow \mathsf{Sol}(1^{\lambda}, \mathsf{mcrs}, t, z_j)$ (where s_j might be $\perp$).
- **Output phase:** If for every puzzle z_j and solution s_j, either (a) there is a published opening r_j such that $z_j = \mathsf{Gen}(1^{\lambda}, t, s_j; r_j)$ or (b) a published proof π_j such that $\mathsf{Verify}(1^{\lambda}, \mathsf{mcrs}, t, z_j, (s_j, \pi_j)) = 1$, then output $s = \bigoplus_{s_j \neq \perp} s_j$.

We note that the protocol above does not assume an a priori bound on the number of participants. Furthermore, there is no external setup needed by the protocol. All participants, however, do publish a random string $\mathsf{crs}_i \leftarrow \{0,1\}^{\lambda}$ that can be used to implement the ABO-string model for $(\mathsf{Gen}, \mathsf{Sol}, \mathsf{Verify})$.

Theorem 5.1. *Let $L(\lambda) \in \mathrm{poly}(\lambda)$. Assume the existence of a publicly verifiable time-lock puzzle for $L(\lambda)$ bit messages in the ABO-string model that satisfies concurrent function non-malleability for $f_{\oplus}$ with $L(\lambda)$ bit output. Then, there exists a multi-party coin flipping protocol that outputs $L(\lambda)$ bits and satisfies optimistic efficiency, fairness, and public verifiability. The protocol supports an unbounded number of participants and requires no adversary-independent trusted setup.*

We obtain the following result by using our publicly verifiable non-malleable TLP construction (given in the full version) with the above theorem.

Corollary 5.2. *Let $B, L \colon \mathbb{N} \to \mathbb{N}$ where $B(\lambda) = 2^{3L(\lambda)}$. Assuming the B-repeated squaring assumption for* RSWGen*, there exists a multi-party coin flipping protocol that outputs $L(\lambda)$ bits and satisfies optimistic efficiency, fairness, and public verifiability. The protocol supports an unbounded number of participants and requires no adversary-independent trusted setup. Security is proven in the auxiliary-input random oracle model.*

Finally, we note that if we instead start with our non-malleable time-lock puzzle in the plain model (which is not publicly verifiable) the non-interactive variant of our protocol gives non-interactive coin flipping in the plain model. In particular, we obtain the following theorem based on our plain model construction (given in the full version).

Theorem 5.3. *Let $L \colon \mathbb{N} \to \mathbb{N}$ and $S(\lambda) = 2^{\lambda + L(\lambda)}$. Assume a time-lock puzzle, a keyless multi-collision resistant hash function, a non-interactive witness indistinguishable proof for* NP*, and injective one-way functions, all sub-exponentially* *secure, where in particular the time-lock puzzle is secure against polynomial-depth adversaries of size S. Then, there exist fully non-interactive fair multi-party coin flipping protocol that outputs $L(\lambda)$ bits, where fairness holds against non-uniform polynomial time distinguishers. The protocol supports an unbounded number of participants and requires no setup.*

We note that if we only consider protocols that output $L(\lambda) \in O(\log \lambda)$ bits, then fairness against polynomial time distinguishers implies statistical fairness. This is because if there is an unbounded distinguisher for $O(\log \lambda)$ bits, we can construct a polynomial time distinguisher that simply hard codes the truth table of the unbounded distinguisher.

We remark how we can adapt our protocol to deal with auctions.

Remark 5.4 (Multi-Party Auctions). For our application to auctions, we consider a standard second-price, sealed-bid auction, in which the auctioned item is assigned to the highest bidder who pays the second highest bid for the item. We assume some form of authenticated channels so we can know the bidders' identities in order to distribute the auctioned items. We leave these as external implementation details for the protocol. The main protocol proceeds as follows.

In the commit phase, each participant computes a time-lock puzzle to their bid. The open and force open phases are identical to the case of coin flipping. Then in the output phase, we need to determine the identity of the highest bidder and the value of the second highest bid.

The function that computes the output consists of finding the top two values in a set. This can be computed in low depth (doing a tree of comparisons in parallel) and has output length $\log n + \log M$ where n is the number of participants and M is a bound on the largest valid bid. Thus, using our publicly verifiable time-lock puzzle that satisfies concurrent functional non-malleability for this function, the resulting protocol is secure assuming $n \cdot M \cdot \text{poly}(\lambda)$ security for the repeated squaring assumption. Assuming n and M are polynomially bounded, we only need polynomial security assumptions.

The proof of Theorem 5.1 is deferred to the full version.

Acknowledgements. This work was supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, NSF Award DGE-1650441, AFOSR Award FA9550-18-1-0267, DARPA Award HR00110C0086, and a JP Morgan Faculty Award. Ilan Komargodski is supported in part by an Alon Young Faculty Fellowship and by an ISF grant (No. 1774/20). This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

1. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: 43rd Symposium on Foundations of Computer Science FOCS. pp. 345–355 (2002)
2. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS. pp. 345–354 (2006)
3. Baum, C., David, B., Dowsley, R., Nielsen, J.B., Oechsner, S.: Craft: Composable randomness and almost fairness from time. Cryptology ePrint Archive, Report 2020/784 (2020), https://eprint.iacr.org/2020/784
4. Baum, C., David, B., Dowsley, R., Nielsen, J.B., Oechsner, S.: TARDIS: A foundation of time-lock puzzles in UC. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 12698, pp. 429–459. Springer (2021)
5. Bentov, I., Pass, R., Shi, E.: Snow white: Provably secure proofs of stake. IACR Cryptology ePrint Archive 2016, 919 (2016)
6. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: ITCS (2016)
7. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Advances in Cryptology - CRYPTO. pp. 757–788 (2018)
8. Boneh, D., Bünz, B., Fisch, B.: A survey of two verifiable delay functions. IACR Cryptol. ePrint Arch. 2018, 712 (2018)
9. Boneh, D., Naor, M.: Timed commitments. In: Advances in Cryptology - CRYPTO. pp. 236–254 (2000)
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145. IEEE Computer Society (2001)
11. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Concurrent non-malleable commitments (and more) in 3 rounds. In: Advances in Cryptology - CRYPTO. pp. 270–299 (2016)
12. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Four-round concurrent non-malleable commitments from one-way functions. In: Advances in Cryptology - CRYPTO. pp. 127–157 (2017)
13. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, STOC. pp. 364–369 (1986)
14. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.P.: Random oracles and non-uniformity. In: Advances in Cryptology - EUROCRYPT. pp. 227–258 (2018)
15. Dachman-Soled, D., Komargodski, I., Pass, R.: Non-malleable codes for bounded parallel-time tampering. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 12827, pp. 535–565. Springer (2021)
16. Daian, P., Pass, R., Shi, E.: Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In: Financial Cryptography and Data Security - 23rd International Conference, FC. pp. 23–41 (2019)
17. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Advances in Cryptology - EUROCRYPT. pp. 66–98 (2018)
18. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, STOC. pp. 542–552 (1991)
19. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: Continuous verifiable delay functions. IACR Cryptology ePrint Archive 2019, 619 (2019)

20. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018. pp. 33–62. Springer (2018)
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptology 26(1), 80–101 (2013)
22. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC. pp. 695–704 (2011)
23. Goyal, V., Lee, C., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp. 51–60 (2012)
24. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC. pp. 1128–1141 (2016)
25. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. J. Cryptology 27(3), 506–543 (2014)
26. Hohenberger, S., Myers, S., Pass, R., Shelat, A.: An overview of ANONIZE: A large-scale anonymous survey system. IEEE Security & Privacy 13(2), 22–29 (2015)
27. Kalai, Y.T., Khurana, D.: Non-interactive non-malleability from quantum supremacy. In: Advances in Cryptology - CRYPTO. pp. 552–582 (2019)
28. Katz, J., Loss, J., Xu, J.: On the security of time-lock puzzles and timed commitments. In: TCC (3). Lecture Notes in Computer Science, vol. 12552, pp. 390–413. Springer (2020)
29. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: Theory of Cryptography - 15th International Conference, TCC. pp. 139–171 (2017)
30. Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS. pp. 564–575 (2017)
31. Lin, H., Pass, R.: Non-malleability amplification. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC. pp. 189–198 (2009)
32. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC. pp. 705–714 (2011)
33. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS. pp. 576–587. IEEE Computer Society (2017)
34. Lin, H., Pass, R., Tseng, W.D., Venkitasubramaniam, M.: Concurrent non-malleable zero knowledge proofs. In: Advances in Cryptology - CRYPTO (2010)
35. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC. pp. 571–588 (2008)
36. Malavolta, G., Thyagarajan, S.A.K.: Homomorphic time-lock puzzles and applications. In: Advances in Cryptology - CRYPTO. Lecture Notes in Computer Science, vol. 11692, pp. 620–649. Springer (2019)
37. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC. pp. 427–437. ACM (1990)
38. Ostrovsky, R., Pandey, O., Visconti, I.: Efficiency preserving transformations for concurrent non-malleable zero knowledge. In: Theory of Cryptography, 7th Theory of Cryptography Conference, TCC. pp. 535–552 (2010)

39. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Advances in Cryptology - CRYPTO. pp. 57–74 (2008)
40. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS. pp. 563–572 (2005)
41. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC. pp. 533–542 (2005)
42. Pass, R., Rosen, A.: Concurrent nonmalleable commitments. SIAM J. Comput. 37(6), 1891–1925 (2008)
43. Pass, R., Wee, H.: Constant-round non-malleable commitments from subexponential one-way functions. In: Advances in Cryptology - EUROCRYPT (2010)
44. Pietrzak, K.: Simple verifiable delay functions. In: 10th Innovations in Theoretical Computer Science Conference, ITCS. pp. 60:1–60:15 (2019)
45. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto (1996), technical report, Massachusetts Institute of Technology, Cambridge, MA, USA
46. Rotem, L., Segev, G.: Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 12172, pp. 481–509. Springer (2020)
47. Unruh, D.: Random oracles and auxiliary input. In: Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. pp. 205–223 (2007)
48. Von Zur Gathen, J., Shparlinski, I.E.: Generating safe primes. Journal of Mathematical Cryptology 7(4), 333–365 (2013)
49. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS (2010)
50. Wesolowski, B.: Efficient verifiable delay functions. In: Advances in Cryptology - EUROCRYPT. pp. 379–407 (2019)

A Discussion of Non-Malleable Definitions

We briefly discuss the different notions of non-malleability studied in this work. Specifically, we compare standard non-malleability (Definition 3.4), non- malleability against depth-bounded distinguishers, and functional non-malleability (Definition 4.1). In section 1.3, we also discuss the definitions considered in the concurrent works of [28, 3, 4].

Common to all of our definitions, there is a depth-bounded man-in-the-middle (MIM) attacker, which we call $\mathcal{A}$, that on input a puzzle z with solution s tries to output a different puzzle $\tilde{z}$ to a related value $\tilde{s}$. Here, $\mathcal{A}$ is depth-bounded relative to the difficulty of the puzzle, so it should not be able to solve the puzzle. The definitions vary in what it means for $\tilde{s}$ to be "related" to s. For our standard notion of non-malleability, we require that no unbounded distinguisher $\mathcal{D}$ on input $\tilde{s}$ can tell if it came from the experiment starting with s or the all-zero string. In the definition of non-malleability against depth-bounded distinguishers, $\mathcal{D}$ is restricted to be depth-bounded in the same way as $\mathcal{A}$. In the case of functional non-malleability, the (unbounded) distinguisher $\mathcal{D}$ receives instead as input $f(\tilde{s})$ where f is a low-depth function. We parameterize functional

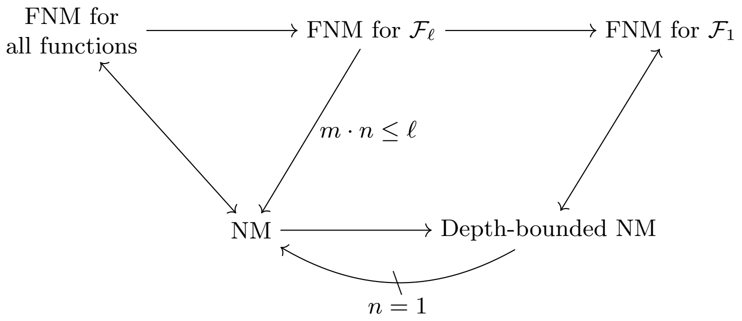

Fig. 1. Relationship between notions of non-malleability. An arrow from A to B indicates that any construction satisfying A also satisfies B. Here, m is the message length, n is the concurrency, and $\mathcal{F}_\ell$ is class of depth-bounded functions with ℓ-bit output.

non-malleability by an output length m. When $m = |s|$, this captures plain non-malleability by considering f to be the identity function. When $m = 1$, this captures depth-bounded distinguisher non-malleability as f essentially plays the role of the depth-bounded distinguisher $\mathcal{D}$. In Theorem 4.2, we show how to construct a time-lock puzzle satisfying functional non-malleability for any output length m assuming a time-lock puzzle that is $2^m \cdot \mathrm{poly}(\lambda)$ secure.

When considering concurrent non-malleability, the MIM attacker $\mathcal{A}$ receives possibly multiple puzzles $z_1, \ldots, z_{n_L}$ that have solutions $s_1, \ldots, s_{n_L}$ as input and tries to output multiple puzzles $\tilde{z}_1, \ldots, \tilde{z}_{n_R}$ (different from its inputs) corresponding to $\tilde{s}_1, \ldots, \tilde{s}_{n_R}$. In the most general form, we can consider some distinguisher $\mathcal{D}$ that receives as input $f(\tilde{s}_1, \ldots, \tilde{s}_{n_R})$ and tries to tell if it came from the experiment starting with $s_1, \ldots, s_{n_L}$ or with n_L all-zero strings. We show in the full version that if the MIM attacker can encode a time-lock puzzle into the value $f(\tilde{s}_1, \ldots, \tilde{s}_{n_R})$ (where f may be the identity), then the construction cannot be secure against an unbounded distinguisher. In particular, if the function's output length m is greater than the output length of the time-lock puzzle, the scheme may not be secure. On the other hand, our construction of Theorem 4.2 works for functional non-malleability even in the fully concurrent setting, as the output length of f is bounded. So, as long as the output length of the function f is sufficiently small, we can support unbounded concurrency.

Finally, our separation in the full version gives a construction that satisfies plain (non-concurrent) non-malleability against depth-bounded distinguishers yet does not satisfy non-malleability against unbounded distinguishers. We remark that in the setting where the message length for the puzzle is 1 bit, these notions are equivalent by simply considering the depth-bounded distinguisher that outputs the bit it gets as input. Moreover, it can be shown that they are equivalent as long as the message length is in $O(\log \lambda)$. Therefore, this separation necessarily relies on the fact that the message length for the puzzle is in $\omega(\log \lambda)$.

We summarize the various relationships between the definitions in Figure 1.