Challenges in Timed-Cryptography: A Position Paper

Karim Eldefrawy *, Benjamin Terner † ⋆, and Moti Yung ‡

∗ SRI International karim.eldefrawy@sri.com
† Confidencial ben.terner@confidencial.io
‡ Google and Columbia University motiyung@gmail.com

Abstract. Time-lock puzzles are unique cryptographic primitives that use computational complexity to keep information secret for some period of time, after which security expires. This topic, while over 25 years old, is still in a state where foundations are not well understood: For example, current analysis techniques of time-lock primitives provide no sound mechanism to build composed multi-party cryptographic protocols which use expiring security as a building block. Further, there are analyses that employ idealizations and simulators of unrealistic computational power to be an acceptable sound security argument. Our goal with this short paper is to advocate for understanding what approaches may lead to sound modeling beyond idealization, and what approaches may, in fact, be hopeless at this task of sound modeling.

We explain in this paper how existing attempts at this subtle problem lack either composability, a fully consistent analysis, or functionality. The subtle flaws in the existing frameworks reduce to an impossibility result by Mahmoody et al., who showed that time-lock puzzles with super-polynomial gaps (between committer and solver) cannot be constructed from random oracles alone (or any repetitive computation where the next state is completely random given the prior state); yet still the analyses of algebraic puzzles today treat the solving process as if each step is a generic or random oracle. We point out that if the generation process relies on a trapdoor function that cannot be treated as a random oracle (to allow efficient generation while avoiding this impossibility result), then, to be consistent, the analysis of the solving process should also not treat such a trapdoor function (and its intermediate states) as a random oracle.

We also delineate additional issues with the proof techniques used for time-lock puzzles. Specifically, when a time-lock puzzle must retain privacy for some amount of time, the reduction should bound the running time of the simulator. A simulator that can "simulate" if given time that if given to an adversary allows said adversary to solve the puzzle is not a valid security argument. We survey the adherence of various attempts to this principle, as well as the properties that different attempts achieve toward composition.

⋆ Work performed partially while at SRI International.

1 Introduction

A healthy tradition in cryptographic modeling has been to review an area where results under various models have been produced, and view then from a critic's point of view for further insights. See Canneti, Halevi, and Goldreich [11] Naor [23], and Goldwasser and Kalai [18] as examples of the above. The objective of this short paper is to apply this tradition to the relatively new area of timed-cryptography, and specifically time-lock puzzles.

Time-lock puzzles are unique cryptographic primitives that use computational complexity to keep information secret for some (polynomially long) period of time, after which security expires. The temporal nature of security is quite unique and challenging. We focus on several issues and challenges that should be addressed to establish sound lasting foundations for timed-cryptography.

The first challenge is that existing models (see table 1) idealize the security analysis of time-lock puzzles in a way that presents contradictions with established results [21]. As we explain in Section 2, the generation of a puzzle is typically analyzed in an algebraic model (enabling efficient generation of puzzles based on standard hardness assumptions such as in [25]), while the security of the puzzle solving process is analyzed in a random oracle or generic model. Such an analyzed solving process (relying on random oracles only, whether explicitly or implicitly) cannot achieve the super-polynomial gap between generation time and solving time due to an impossibility result by Mahmoody et al.[21]. The "gap" is the difference between the time to generate a time-lock puzzle and the time required to solve it. Current works effectively exhibit an exclusive OR of two basic properties: the "super-poly gap property" (required for correctness of the functionality and its security) and "consistent analysis property" (for the security guarantees via rigorous analysis to make sense).

Moreover, we note that the existing idealized analyses of time-lock primitives are all in non-falsifiable models [23], whereas eventually we need to model it as in other areas of cryptography, relying on falsifiable assumptions.

In this paper, we present arguments that we believe complete the discussion of time-lock primitives, and which to the best of our knowledge are not comprehensively addressed in any existing work. In Section 3 we present a potential remediation for the issue of modeling intermediate states as random. In Section 4 we recall the foundational simulation arguments for secure multiparty computation (MPC) and explain how current attempts fall short of complete proofs for this standard. In Section 5 we survey how a few additional notable works on composable security achieve consistent results but fall short of full composability.

Given the relatively early stage of development of carefully examined timed primitives in general, we argue that any formal and rigorous analysis is appreciated and such an inconsistent situation – while initially tolerated – should not persist forever. While we appreciate all existing efforts and the careful analysis (in Table 1), and the difficulty of coming up with lasting foundations of such an intricate topic, we still argue that the community should strive towards a long-term viable solution to resolve such inconsistencies.

	Analytical Model			
Protocol	**Generate**	**Solve**	**Super-poly Gap**	**Consistent**
Arapinis et al. [2]	Idealized (RO)	Idealized (RO)	NO	YES
Baum et al. '20b [5]	Algebraic	Idealized (RO)	YES	NO
Baum et al. '20a [6]	Algebraic	Idealized (RO)	YES	NO
Katz et al. [19]	Algebraic	Idealized (SAGM)	YES	NO
Chvojka, Jager [12]	Algebraic	Idealized (†)	YES	NO

Table 1: Properties of Analytical Models for Composable Timed Primitives. "Generate" and "Solve" describe the analytical framework used for the respective TLP algorithms, whether they are in algebraic or idealized models. RO stands for "Random Oracle" and SAGM stands for "Strong Algebraic Group Model." "Superpoly Gap" describes whether the analytical framework admits a superpolynomial gap between generation and solving. The "Consistent" column describes whether the analyses of generation and solving are consistent. (†) Depends on the Strong Sequential Squaring Assumption, but is effectively treated as a Random Oracle (RO) in the security proof.

2 Subtleties and Inconsistencies in Random Oracle Based Analysis for Time-Lock Puzzles

We discuss here the details of the subtleties and inconsistencies in (explicit and implicit/hidden) random oracle and generic domain based security analysis of timed-primitives. Similar to the classical result of [10] in the random oracle (RO) model and [13, 17] about the Fiat-Shamir transform [14], we argue that the desired properties of realizable time-lock puzzles do not follow from the current analyses.

All current computational puzzles follow the following blueprint:

1. Puzzle **generation** uses a trapdoor to efficiently sample a puzzle (efficient generation, in fact, is part of the functionality's definition).
2. Puzzle **solving** uses a sequential algorithm. The puzzle is parameterized such that the (polynomial) sequential algorithm is faster than any known (super-polynomial) method to recover the trapdoor.

For a puzzle to be a useful functionality, it must be *efficient to generate* and *hard to solve*. The trapdoor is therefore required for *utility*, while the hardness of the computational problem is required for time-lock *security*.

However, analysis of these algorithms occurs in *inconsistent models* (see Table 1). For all existing constructions that claim to achieve a super-polynomial gap between generation and solving:

- The **generation** algorithm is analyzed **in an algebraic model**.
- The **solution** algorithm is analyzed **as if each step is a random oracle**.

To elucidate the inconsistency, we quote directly from the approach of [5] (full version [4] page 5, second paragraph). In the excerpt, they describe the leakage provided by their time-delay functionality:

> "The intermediate states leaked to the adversary by the functionality are not concrete representations of actual intermediate states but generic random labels assigned to states (similarly to the generic group model treatment given to the RSW assumption in [6]). Learning these intermediate generic states does not allow the environment (or the adversary) any advantage in computing the next states before they are revealed by the functionality, as these states are sampled uniformly at random."

In both of these works [5, 6] – as well as any analyses using either the RO or the strong algebraic group model – the proofs explicitly assume that no information is revealed to the adversary at each step of the solving process, and that each intermediate state is sampled uniformly at random. *The analysis therefore effectively turns the algebraic structure into a random oracle with properties identical to the impossibility by Mahmoody et al [21].* We stress that it is inconsistent to analyze puzzle generation in an algebraic model – where super-polynomial gaps are believed to be achieved – and solution in a random oracle model – which can only achieve polynomial gaps. The current literature overlooks this inconsistency which we argue should be resolved for the topic to progress and be developed on a sound and a consistent basis.

The Strong Algebraic Group Model. In the strong algebraic group model of [3, 19] and the generic ring model of [26], each element is expressed as a function of factors or as an inverse of another element, which gives algebraic structure to the elements that have been seen, but leaks no more about the solution than a random oracle. It therefore incurs the same problems as the previous analysis. As discussed, this is also how some work [5, 6] analyzes the utilized algebraic functionalities.

Other Approaches with Similar Subtleties. Other works do not explicitly model the solving process via a random oracle, but either the modeling implies a random oracle or it overlooks leakage as the puzzle is partially solved. For example, the base time-lock puzzle in the construction of Freitag et al.[16] defer to analysis by Pietrzak [24] that assumes the hardness of repeated squaring. Chvojka and Jager [12] similarly reduce to the Strong Sequential Squaring assumption. These formalizations simply assume it is infeasible to guess the solution of a repeated squaring until the final squaring; either it implicitly treats the process as if the probability of guessing the solution before the end is negligible, or it uses a game-based definition that implies the solution process is essentially a random oracle until the very end. Therefore, these techniques as well are not differentiated in any meaningful way from the analysis of Mahmoody et al. [21].

3 On Random Oracles, Generic Models, and Inconsistent Analysis

We point out inconsistencies inherent in the approaches described in Section 2. It may be the case that previous efforts could not find a better way to argue or

prove security in the desired models, but in this case justification is required for the difficult modeling issues.

We point out that it is inconsistent to argue that puzzle generation occurs in one model (algebraic) and puzzle solving occurs in another model (random, or generic which implicitly models the random oracle). The impossibility by Mahmoody argues that puzzle generation in the random oracle model cannot be "sped up" exactly because of this randomness – in order to compute the solution, one must go through the effort of computing all the steps of the solution. It is for this reason that researchers may be turning to generic algebraic models, because the algebraic structure allows for some trapdoor which circumvents the issue raised by Mahmoody. We argue that this is too good to be true: the generic group approaches explicitly treat the intermediate states as elements with *exactly the same properties* used by Mahmoody in order to prove the impossibility.

We claim that intermediate states in the generic group model are effectively random states that model the same properties used by Mahmoody et al. in their impossibility result. If this is not actually the case, and there are additional subtleties overlooked by our claim, then this should be falsified. We invite the community to develop approaches to rigorously argue this point.

Looks Random to Whom? One may justify utilizing the generic group model by arguing that such intermediate states are not *actually* random. Instead, they only *"look"* random to a distinguisher that is computationally bounded, specifically by the length of time required to solve the puzzle.

This may be the case, but then the following questions must be asked:

1. Is treating the intermediate states of a puzzle as random consistent with the goals of the proof?
2. More specifically, to whom must the sequence look random, and is that party bounded in computation sufficiently for these states to look random?
3. If the algebraic sequence looks random to the right parties, then is it even possible to utilize an algebraic structure for the proof?

Rather than answering the above questions satisfactorily, we find that instead the current analyses lose sight of the goal. More often, it is *not* the case that the sequence of intermediate states look random to the party in question, because that party is provided with sufficient computational power (i.e., time) to distinguish the sequence of states from a random one, or more specifically, there is a point at which this occurs that is not explicitly handled by the analysis.

Random to Some. As an example for the sake of illustration, we recall the definition of the generalized Blum-Blum-Shub assumption [7]. The assumption was introduced by [7] in the design of a pseudorandom generator, and later generalized by [8] in the development of timed commitments. We will use this assumption to highlight the fact that to some computationally bounded distinguisher, a sequence of numbers may look random (or the next number in the sequence may look random), but a (slightly) more powerful distinguisher can easily distinguish such a distribution from random.

Definition 1 ($(n, n', \delta, \varepsilon)$-Generalized BBS Assumption [8]). *For $g \in \mathbb{Z}$ and a positive integer $k > n'$, let $W_{g,k} = \langle g^2, g^4, g^8, \ldots, g^{2^i}, \ldots, g^{2^k} \rangle$. Then for any integer $n' < k < n$ and any PRAM algorithm $\mathcal{A}$ whose running time is less than $\delta * 2^k$, we have that*

$$| \Pr[\mathcal{A}(N, g, k, W_{g,k} \mod N, g^{2^{2^{k+1}}}] - \Pr[\mathcal{A}(N, g, k, W_{g,k} \mod N, R^2] | \leq \varepsilon$$

where the probability is taken over the random choice of an n-bit RSA modulus $N = p_1 p_2$, where p_1 and p_2 are equal length primes satisfying $p_1 = p_2 = 3 \mod 4$, and element $g \in \mathbb{Z}_N$, and $R \in \mathbb{Z}_N$.

As noted in [8], the assumption states that given the sequence of repeated squares $W_{g,k}$, the next element in the sequence $g^{2^{2^{k+1}}}$ is indistinguishable from a random quadratic residue, for any party whose running time is much less than 2^k. This assumption suffices for showing the pseudo-randomness of the BBS generator [7]. However, for time-lock puzzles, where eventually the depth of the puzzle solver *must* approach the sequence length 2^k, the guarantee of the BBS assumption breaks down. Indeed, the guarantee of the assumption breaks down at the point that a time-lock puzzle begins to lose its "time-lock" property, since the next item in the sequence can be predicted when δ is quite close to 1.

When proving the security of a time-lock primitive, *if one makes the argument that the intermediate states look random*, then one must take care that the running time of the simulator does not approach the bound by which the intermediate states stop looking random.

4 Meaningful Simulation of Time-Lock Puzzles

When proving security via simulation for timed primitives, we find that there is an argument missing for the validity of the simulation. Because timed primitives are specifically constructed for security against computationally bounded adversaries, the simulation argument must also account for the computational power of the simulator.

Recall the standard definition of security of secure multiparty computation (MPC) defined via the Real-Ideal paradigm. A protocol π securely realizes functionality $\mathcal{F}$ if for every adversary $\mathcal{A}$ and environment $\mathcal{Z}$ there exists a simulator $\mathcal{S}$ such that the ensembles

$$\{\mathsf{REAL}_{\pi,\mathcal{A}(z),\mathcal{Z}}(\overline{x})\}_{\overline{x},z,\lambda}$$

and

$$\{\mathsf{IDEAL}_{\mathcal{F},\mathcal{S}(z),\mathcal{Z}}(\overline{x})\}_{\overline{x},z,\lambda}$$

are computationally indistinguishable.

Consider the reduction and its meaning. By showing that the distribution of views produced by the simulator is indistinguishable from the view of the adversary, we imply that the adversary can do "no worse" than a simulator

that did not know the honest parties' inputs. Therefore, it is concluded that the adversary can learn only what it gleans from its own inputs and from the protocol output. When moving into a paradigm where the timing of the protocol matters – such as the time spent solving a puzzle – the relationship between the adversary and the simulator must be preserved, inasmuch as we wish to show that the adversary can "do no worse" than a given simulator which did not know any inputs.

However, in the common practice today, some simulator are still by default assumed to be granted *arbitrary polynomial time* to conduct their simulations, and in some cases proofs are written in which a simulator explicitly solves a time-lock puzzle. When such a proof it written, it argues for privacy against an adversary that can "do no worse" than a simulator *which can explicitly solve the puzzle guarding privacy!*

Such an argument may be the correct one, depending on context. But the point remains that every simulator written for a timed primitive must be forthright about the computational bounds on the simulator, in order for the argument to be complete.

How Much is Too Much: Simulation is a proof technique that argues about the ability of a computationally bounded adversary to break a given primitive or protocol. For all applications of the simulation technique, there is some bound on the adversary that the simulator must respect. Let F represent a class of functions for which we wish to prove that any circuit $\mathcal{A}$, whose size (as a function of the security parameter) belongs to a function in F, cannot break a given primitive. In the simulation argument used in the proof, the simulator must also be bounded by the class F. As an example, for any argument that no non-uniform probabilistic polynomial time adversary can break primitive P, one cannot argue this fact by designing a simulator which runs in super-polynomial time!

Sound Simulation: A Step-by-Step Companion. The most common, and obvious, technique for simulating the result of a time-release primitive (whether time-lock puzzle, verified delay function, or other), is the step-by-step revelation pattern. Simply, as the adversary makes each step towards solving a puzzle, through whatever idealization/functionality exists to gatekeep the next intermediate state, the simulator is one step ahead of the adversary and queries the appropriate functionality so that it can provide the right answer to the adversary. We argue that this is the most natural design pattern for simulation, and indeed it formalizes the notion that *the simulator is just as powerful* as the adversary! (We mean just as powerful computationally, since it will cheat to equivocate puzzles, either by programming a random oracle or by reading the input to a functionality which it emulates.)

Global Clocks and Untethered Simulators. As a case study in the use of time, consider the work of Baum et al.[6]. To model time without requiring specific control of time-based resources, they delegate to a global time-keeping

functionality in the GUC model ([9]), adapting the spirit of Kiayias et al.[20] by relinquishing the responsibility of all parties to keep track of the global time, and simply deferring to a global timekeeper.

Importantly in this framework, functionalities must tick the global timekeeping functionality in order for the protocol time to move forward. This removes control of time from the domain of the adversary, and effectively constrains the adversary's ability to solve a puzzle to the limits of the time elapse controlled by the functionalities.

However, although (to the authors credit) great care is taken in the framework to idealize the functionality of a time-lock puzzle and limit the queries that the adversary makes to the solve routine – whose query limit depends on the global clock beyond the adversary's control – unfortunately, no such care is taken to limit the power of the simulator. In fact, the simulator in the reduction(s) is able to explicitly follow the honest party's solution algorithm in order to open puzzles *without* ticking the global functionality. What results in the reduction is an adversary that is strongly constrained in ways that the simulator is not, particularly with the ability to advance a time-lock computation.

The work of Arapinis et al.[1], which is the only other construction to claim UC-security for time-lock puzzles (but does not suffer from inconsistent modeling because it resides entirely in the RO model), similarly abstracts time to a global functionality. Although the simulator in their proofs does not explicitly solve a time-lock puzzle, they also do not explicitly bound the computation of the simulator. With access to the appropriate random oracles, their simulator therefore could potentially solve a time-lock puzzle on its own by following the honest party's solving strategy.

5 Limited Composition

Other works on the composition of time-lock puzzles fall short of sufficient composability properties that they can be placed within MPC.

Combining Puzzles. For example, the work of Malavolta and Thyagarajan [22] achieves homomorphic time-lock puzzle constructions which combine time-lock puzzles. But this exists only to compose time-lock puzzles with other time-lock puzzles, with the result of a single time-lock primitive. Although all of the adversaries in their analyses and reductions *are* properly bounded in depth to be less than the time to solve a puzzle, they do not achieve composition with a second primitive or another copy of a time-lock puzzle that isn't homomorphically combined.

Concurrent Puzzles. The work by Freitag et al.[16] achieves functional non-malleability, which securely achieves solving multiple puzzles simultaneously, while preventing the adversary from constructing a puzzle whose solution is a function of an honest party's puzzle. In this work as well, the computational constraints of all machines are explicitly stated. However, the analysis does not yield general composition for MPC.

In an addendum to their work [15], Freitag et al. expand upon their definition of functional non-malleability and its relationship to nonmalleability with a depth-bounded distinguisher. Intuitively, in their nonmalleability definition, they insert a function f that pre-processes the output of the man-in-the-middle adversary and potentially changes the length of the output. (This function is necessary because they originally considered only distinguishers with unbounded depth.) When the function has low depth and small output length (such as a single bit), then this function essentially serves as the distinguisher.

The following commentary is *not* in the domain of [16, 15], whose analysis is self-consistent. However, for the purpose of simulation-based analysis, we argue that the distinguisher (or this intermediate function f) should always have depth at least that of the simulator, because the distinguisher should always at least be able to run the simulator as part of its algorithm.

For example, let the simulator in such an analysis depend upon the (functional) non-malleability of time-lock puzzles. The simulator may depend on the fact that the (man-in-the-middle adversary, distinguisher) pair cannot together distinguish real puzzles from puzzles constructed via dummy inputs. The distinguishing environment may run f on the output produced by the experiment. If the protocol is only analyzed for f with low depth, this may not capture the power of the distinguishing environment.

IND-CCA. The works of Katz et al.[19] and later Chvojka and Jager [12] achieve IND-CCA security. Katz et al. provide the running time of all of their simulators. All of the proofs by Chvojka and Jager avoid the need to brute-force solve a puzzle within the security argument. We have argued that these are necessary arguments for a full proof of timed cryptographic primitives. However, these works still fall short of the composability standards for composing with MPC.

6 Conclusion and Moving Forward

The foundations of timed cryptography are still not fully understood. Our position is that new directions are needed to understand which approaches may lead to sound modeling and which approaches may, in fact, be hopeless at this task.

To recap, in this paper we have shown how current analysis techniques of time-lock primitives exhibit the following shortcomings:

1. Concurrently employ idealizations and proof techniques which are inconsistent when paired together in one argument.
2. Fail to constrain the computational power of simulators to imply meaningful statements about the adversary.
3. Provide no sound mechanism to build (composed) multi-party protocols with expiring security as a building block.

Specifically, the first issue we highlight is that existing models idealize the security analysis of time-lock puzzles (by modeling intermediate states as tags from random oracles) in a way that presents contradictions with established

impossibility results. The established impossibility result we refer to shows time-lock puzzles with super-polynomial gaps (between committer and solver) cannot be constructed from random oracles alone. The same impossibility result applies also for constructions from any repetitive computation where the next state is completely random given the prior state.

The second issue is that a simulator which serves as part of a security argument in timed-cryptography requires additional attention so that it does not serve as an underlying problem breaker which trivializes the security argument.

The third issue is that we would like the model of timed cryptography to support (general) composability of timed protocols as a subroutine in a larger cryptographic protocols (again, as in other cryptographic protocol areas).

We argue that inconsistent and incomplete modeling and analysis should be considered an *initial* and *temporary* state of investigation. Given the issues we discuss above, such idealized inconsistent modeling and security analyses should not be regarded as the final word on actual realizable timed-cryptographic protocols and constructions. We appreciate the fact that indeed there are profound difficulties due to the peculiarities of timed-cryptography. Furthermore, we completely accept the fact that initial investigations took shortcuts and used idealizations to probe a formal treatment of the area, and hence provided insights that will guide future investigations. This has been the tradition in designing primitives in cryptography throughout modern cryptography; the profound difficulties posed by timed primitives should not deter us from the solid and tested tradition which is at the core of the field. As we move forward, the field should establish foundations of composable timed cryptography in consistent models.

References

1. Arapinis, M., Kocsis, Á., Lamprou, N., Medley, L., Zacharias, T.: Universally composable simultaneous broadcast against a dishonest majority and applications. In: PODC. pp. 200–210. ACM (2023)
2. Arapinis, M., Lamprou, N., Zacharias, T.: Astrolabous: A universally composable time-lock encryption scheme. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 13091, pp. 398–426. Springer (2021)
3. van Baarsen, A., Stevens, M.: On time-lock cryptographic assumptions in abelian hidden-order groups. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 13091, pp. 367–397. Springer (2021)
4. Baum, C., David, B., Dowsley, R., Kishore, R., Nielsen, J.B., Oechsner, S.: CRAFT: Composable randomness beacons and output-independent abort MPC from time. Cryptology ePrint Archive, Paper 2020/784 (2020), `https://eprint.iacr.org/2020/784`, `https://eprint.iacr.org/2020/784`
5. Baum, C., David, B., Dowsley, R., Kishore, R., Nielsen, J.B., Oechsner, S.: CRAFT: composable randomness beacons and output-independent abort MPC from time. In: Public Key Cryptography (1). Lecture Notes in Computer Science, vol. 13940, pp. 439–470. Springer (2023)
6. 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)

7. Blum, L., Blum, M., Shub, M.: Comparison of two pseudo-random number generators. In: Crypto82. pp. 61–78. Plenum (1982)
8. Boneh, D., Naor, M.: Timed commitments. In: Crypto'00. p. 236–254. LNCS, Springer-Verlag, Berlin, Heidelberg (2000)
9. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: TCC. Lecture Notes in Computer Science, vol. 4392, pp. 61–85. Springer (2007)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: STOC. pp. 209–218. ACM (1998)
11. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited **51**(4) (2004). https://doi.org/10.1145/1008731.1008734, `https://doi.org/10.1145/1008731.1008734`
12. Chvojka, P., Jager, T.: Simple, fast, efficient, and tightly-secure non-malleable non-interactive timed commitments. In: Public Key Cryptography (1). Lecture Notes in Computer Science, vol. 13940, pp. 500–529. Springer (2023)
13. Dachman-Soled, D., Jain, A., Kalai, Y.T., Lopez-Alt, A.: On the (in)security of the fiat-shamir paradigm, revisited. Cryptology ePrint Archive, Paper 2012/706 (2012), `https://eprint.iacr.org/2012/706`, `https://eprint.iacr.org/2012/706`
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
15. Freitag, C., Komargodski, I., Pass, R., Sirkin, N.: Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Paper 2020/779 (2020), `https://eprint.iacr.org/2020/779`, `https://eprint.iacr.org/2020/779`
16. Freitag, C., Komargodski, I., Pass, R., Sirkin, N.: Non-malleable time-lock puzzles and applications. In: TCC (3). Lecture Notes in Computer Science, vol. 13044, pp. 447–479. Springer (2021)
17. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. In: FOCS. pp. 102–113. IEEE Computer Society (2003)
18. Goldwasser, S., Tauman Kalai, Y.: Cryptographic assumptions: A position paper. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography. pp. 505–522. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
19. Katz, J., Loss, J., Xu, J.: On the security of time-lock puzzles and timed commitments. In: TCC (3). LNCS, vol. 12552, pp. 390–413. Springer (2020)
20. Kiayias, A., Zhou, H., Zikas, V.: Fair and robust multi-party computation using a global transaction ledger. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 9666, pp. 705–734. Springer (2016)
21. Mahmoody, M., Moran, T., Vadhan, S.P.: Time-lock puzzles in the random oracle model. In: CRYPTO. LNCS, vol. 6841, pp. 39–50. Springer (2011)
22. Malavolta, G., Thyagarajan, S.A.K.: Homomorphic time-lock puzzles and applications. In: CRYPTO (1). LNCS, vol. 11692, pp. 620–649. Springer (2019)
23. Naor, M.: On cryptographic assumptions and challenges. In: CRYPTO. Lecture Notes in Computer Science, vol. 2729, pp. 96–109. Springer (2003)
24. Pietrzak, K.: Simple verifiable delay functions. In: ITCS. LIPIcs, vol. 124, pp. 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
25. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep. (1996)
26. 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)