Synchrony Amplification

Ueli Maurer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
Email: maurer@inf.ethz.ch

Björn Tackmann
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
Email: bjoernt@inf.ethz.ch

***Abstract*—Various protocols in the cryptography and distributed systems literature assume some notion of time: One major (but not the only) example are "synchronous" models which assume that a protocol is executed in a well-defined sequence of rounds with round switches that occur (almost) simultaneously at the parties. In many of the considered models, the notion of time is either implicit, or it is closely interweaved with other mechanics of the model such that formally proving even simple statements becomes a tedious task.**

In this work, we develop an abstract formal model that captures exactly how the availability of clocks with "weak" synchrony guarantees can benefit parties; in particular, we show how—and at what cost—the "synchrony" of clocks can be improved. Proofs in this model are simple and the statements transfer to all models that satisfy the abstraction.

The main contribution of this paper is not the actual statements we prove (which mostly verify folklore beliefs), but the formal model that follows the construction paradigm of abstract cryptography and allows to state these proofs in a simple yet rigorous manner. Indeed, the paper is a step towards a treatment of synchronous cryptographic protocols in this constructive sense.

I. Introduction

Many protocols that are used in computer networks make use of time. This may either be explicit, as for "time-out" messages that are issued if some expected message is delayed for too long, or implicit, as for synchronous protocols that are executed in a sequence of "rounds," where the local round switches at the parties involved in the protocol execution occur (almost) simultaneously. The usual justification for these models is that the "rounds" can be emulated using weakly synchronous clocks and communication channels with a known upper bound on the delay until a message is delivered.

One natural goal of parties that have access to weakly synchronous clocks is to improve the synchrony, potentially at the cost of reducing the clock speed. In the spirit of [MR11], we consider clocks as resources available to the parties and a protocol as a construction of one resource from another one. In particular, we ask which constructions can be achieved.

A. Previous Work

The arguably most influential paper on the effects of time on distributed systems is [Lam78]. In fact, the physical clock model introduced there is continuous but otherwise similar to the formalization we present here. The main benefit of our model is the cleaner and more rigorous formalism that makes the goals and assumptions of protocols explicit and is derived from [MR11].

The model presented in [KLP05] extends the interactive Turing machine (ITM) framework widely used in cryptography by extending each such ITM with an additional "clock tape." The clock tapes of the individual ITMs (which encode the protocol) can be written to by the "adversary," which is another ITM that captures potential misbehavior. While this formalism drawn from the models prevalent in the cryptographic literature makes the model very expressive, its complexity inhibits simple proofs (even for trivial statements). Also, the exact formalization prohibits statements about the termination of protocols, which is one major reason for protocols to use time.

The *Timed Automata* in [LV96], [KLSV03] extend the state-transition model of automata by continuous trajectories on the states. While this allows to capture the influence of real-time in a detailed and concrete way, it seems fair to say that it results in a fairly complex model.

B. Contributions

The main contribution of this work is the rigorous and formal yet simple model. The definitions are based on ideas developed in [MR11]; clocks are considered as *resources* that are available to the parties, and a protocol is a *construction* between such resources.

The model developed here, however, is restricted in that it does not allow for resources being available in addition to the clocks. An extended model that is capable of also capturing reactive resources such as communication channels (while preserving the simplicity of the current model) is currently in preparation.

II. Preliminaries

A. Notation

For a number $n \in \mathbb{N}$, let $[n] \coloneqq \{1, \ldots, n\}$. If, for two functions f and g, we write $f(\cdot) \geq g(\cdot)$, we mean that the condition holds on the complete domain.

B. Systems: Resources and Converters

Both clocks and protocols are formalized as *systems*. At the highest level of abstraction, a *system* is an abstract object with *interfaces* through which it interacts with the environment and with other systems [MR11]; interfaces are labeled with elements of a label set $\mathcal{I}$. Two systems are composed into a single system by connecting one interface of each system.

We consider the setting in which n parties have access to (weakly) synchronous clocks. We distinguish two types of systems: The first type are *n-party clocks* that are resources according to [MR11] and provide one interface for each party; resources are generally denoted by upper case letters. The second type are *converters* that are denoted by small Greek letters and model local actions of a party. A converter provides two interfaces: one *inner* interface that connects to a clock and one *outer* interface provided to the party. If the party corresponding to interface $i \in \mathcal{I}$ accesses the resource R via the converter τ, this is denoted as $\tau^i R$.

For such a systems algebra, we usually require the following type of generalized associativity, which is explained in more detail in [MR11].

Definition 1. A set of systems with compositions is *composition-order independent* if for any system composed of several systems, the order in which the systems are composed does not matter.

A protocol for resources with $\mathcal{I}$ interfaces can then simply be defined as a tuple of $|\mathcal{I}|$ converters, one converter for each party that has access to the resource.

C. Constructive Cryptography

This paper is based on the construction paradigm introduced in [MR11], [Mau11]: The goal of a protocol or scheme is phrased as *constructing* a resource with the desired behavior from one or more given ones. As the given resources and the constructed ones are objects of the same type, such a definition is predestined for protocol composition: The resources constructed by one protocol are used by another one, which induces a serial composition operation for protocols.

Definition 2 (Construction)**.** A *construction*[1] for a resource set Ω and a constructor set Γ is a subset of $\Omega \times \Gamma \times \Omega$. A construction is often denoted as an arrow "$\longrightarrow$" as follows: If (R, α, S) is in the construction, then we write $R \xrightarrow{\alpha} S$ and say that S can be *constructed from* R (or that S can be *reduced to* R) by α.

In this paper, the resource set Ω is the set of all clocks, and a constructor in the set Γ corresponds to a scheme or protocol executed by the parties.

D. Specifications

A further notion derived from [MR11] is the concept of a *specification*, which technically is a *set* of resources and formalizes the guarantee that the parties have access to *any one* resource from the set. Construction notions for resources extend generically to specifications: For two specifications $\mathcal{R}, \mathcal{S} \subseteq \Omega$ and a constructor $\alpha \in \Gamma$,

$$\mathcal{R} \xrightarrow{\alpha} \mathcal{S} \quad :\Leftrightarrow \quad \forall R \in \mathcal{R}\ \exists S \in \mathcal{S} : R \xrightarrow{\alpha} S.$$

III. Time and Clocks

For protocols that make use of clocks, the absolute value of the time is not of interest.[2] Indeed, for a single isolated party, a local clock is merely a source of "activations" (such as a processor's clock); a party can of course assign to each activation some label or identifier.[3] In distributed systems, the main use of time is to derive information about the *order of events* at different locations [Lam78], and a guarantee on the synchrony of clocks states that the activations issued to different parties are ordered in a specific manner.

These two purposes of using clocks—activations and some global ordering—are captured by specifying the availability of (weakly) synchronous clocks as an n-party resource. Restricted to the local view of each party, a clock is merely an (a priori infinite) sequence of activations, that is, unary outputs of the resource. The "global" guarantee of the clocks for multiple parties is described exactly by how the activations to the individual parties are interleaved.

Definition 3 (Clock)**.** An *n-party clock* is a sequence $C = (C_k)_{k \geq 1}$ of sets with $C_k \subseteq [n]$.

The interpretation of a clock C is that it proceeds in steps, and in the k-th step it issues an activation at the interfaces of all parties i with $i \in C_k$. The setting

[1] This concept was called *reduction* in [MR11]; however, we prefer the more descriptive term *construction* from [Mau11].

[2] Some run-time environments use the clock register as a source of entropy, though.

[3] A party can in particular count the activations and call some number of such activations a second, a minute, and so on.

where $|C_k| > 1$ corresponds to the situation where two parties obtain activations in such a way that no "real-time" difference can be measured between these events; they are (essentially) simultaneous. Intuitively, the global index corresponds to the most fine-grained resolution that can be observed with respect to time.

The knowledge about the ordering of events is often not complete; the parties might only know that their clocks advance at similar speed. Such a weaker guarantee is specified by a *clock specification*, a set of clocks.

Definition 4 (Clock specification)**.** An *n-party clock specification* is a set $\mathcal{C}$ of n-party clocks.

The *local time* of a party $i \in [n]$ connected to a clock $C = (C_k)_{k\geq 1}$ with respect to the global index k can, without loss of generality, be defined to be the number of activations that the party obtained up to k.

Definition 5 (Local time)**.** The *local time* at the k-th index at party $i \in [n]$ is defined as

$$\mathrm{loc}_C^i(k) \coloneqq |\{k' \leq k \mid i \in C_{k'}\}| .$$

The speed of a clock is measured by its "rate" with respect to the global index k: the number of activations issued up to k. The rate can of course be defined for each individual party or for the complete clock (as the rate of the "slowest" party). Since we will be interested in comparing how the rates of clocks evolve over time, we consider the rate of a clock as a function of the index.

Definition 6 (Rate)**.** For a clock C and an index $k \in \mathbb{N}$, the *rate of C at party i up to index k* is the value $\mathrm{rat}_C^i(k) \coloneqq \frac{\mathrm{loc}_C^i(k)}{k}$. For brevity, we define $\mathrm{rat}_C(k) \coloneqq \min_{i\in[n]} \mathrm{rat}_C^i(k)$ and the *asymptotic rate* $\mathrm{rat}_C \coloneqq \lim_{k\to\infty} \mathrm{rat}_C(k)$ (if defined).

An important measure of the synchrony of a clock is the offset, that is, the maximum difference between the local times at two different parties.

Definition 7 (Offset)**.** For a clock C, the *offset* at index $k \in \mathbb{N}$ is defined as

$$\mathrm{off}_C(k) \coloneqq \max_{i,j\in[n]} \left|\mathrm{loc}_C^i(k) - \mathrm{loc}_C^j(k)\right| .$$

For a function $\delta : \mathbb{N} \to \mathbb{N}$, a clock C is called a *δ-bounded offset clock* if $\mathrm{off}_C(\cdot) \leq \delta(\cdot)$. The specification $\mathcal{C}_\delta^{\mathsf{off}}$ is defined as $\mathcal{C}_\delta^{\mathsf{off}} \coloneqq \{C | \, \mathrm{off}_C(\cdot) \leq \delta(\cdot)\}$.

Bounded-offset clocks guarantee that the clocks of all parties advance at (essentially) the same speed. This is a strong assumption, which is not justifiable for many clocks used in practice where one might only want to make the assumption that the difference in speed is bounded; that is, the clocks have a bounded *drift*.

Definition 8 (Drift)**.** For $k \in \mathbb{N}$, the *drift* of a clock C up to index k is defined as

$$\mathrm{drf}_C(k) \coloneqq \max_{i,j\in[n]} \frac{\mathrm{loc}_C^i(k)}{\mathrm{loc}_C^j(k)}.$$

We set $\mathrm{drf}_C(k) \coloneqq 1$ for all k with $\min_{i\in[n]} \mathrm{loc}_C^i(k) = 0$. For a function $\rho : \mathbb{N} \to \mathbb{R}$, a clock C is called a *ρ-bounded drift clock* if $\mathrm{drf}_C(\cdot) \leq \rho(\cdot)$. The specification $\mathcal{C}_\rho^{\mathsf{drf}}$ is defined as $\mathcal{C}_\rho^{\mathsf{drf}} \coloneqq \{C | \, \mathrm{drf}_C(\cdot) \leq \rho(\cdot)\}$.

Note that our definition of drift differs from that used, for instance, in [KLP05] in that they require that the difference in speed is bounded for every time interval. Such a notion of "smoothness" is stricter, but our result in Theorem 8 extends to this case.

IV. Converters, Protocols, and Constructions

A. Converters

If the only resources available to the parties are clocks and they cannot make use of other resources such as communication channels, then a converter that a party uses to "transform" clocks can only ignore certain activations and forward others. Consequently, the converter can be specified as a sequence of integers that specify the activations that are forwarded.

Definition 9 (Converter)**.** A *(clock) converter* τ is a strictly increasing (finite or infinite) sequence $(t_l)_{l\geq 1}$ of integers $t_l \in \mathbb{N}$.

We describe the clock that is produced by applying a converter at some interface of the original clock.

Definition 10. Let $C = (C_k)_{k\geq 1}$ be a clock, i be an interface of the clock, and $\tau = (t_l)_{l\geq 1}$ be a converter. The converted clock $C' \coloneqq \tau^i C$ obtained by attaching the converter τ at the interface of party i is described by the sequence $C' = (C'_k)_{k\geq 1}$ with

$$C'_k = \begin{cases} C_k, & \text{if } \exists l \in \mathbb{N} : \mathrm{loc}_C^i(k) = t_l, \\ C_k \setminus \{i\} & \text{otherwise.} \end{cases}$$

For a clock specification $\mathcal{C}$, we define

$$\tau^i \mathcal{C} \coloneqq \left\{\tau^i C : C \in \mathcal{C}\right\} .$$

Note that the algebra of clocks and converters fulfills the notion of composition order independence from Definition 1. This is obvious because applying a converter at some interface i "transforms" the clock in a way irrespective of what happens at the other interfaces.

B. Protocols

A protocol is a tuple of converters; one converter for each party.

Definition 11. A *(clock transformation) protocol* is a tuple $\boldsymbol{\tau} = (\tau_i)_{i\in[n]}$ of converters $\tau_i = (t_l^i)_{l\geq 1}$. Applying of a protocol to a clock is defined as $\boldsymbol{\tau}C = \tau_1^1 \dots \tau_n^n C$. This notion extends naturally to clock specifications.

By definition, a protocol can only decrease the rate of a clock (converters cannot generate activations). In practical applications, one would be interested in protocols with a minimal such slackness.

Definition 12 (Efficiency)**.** Let $\boldsymbol{\tau}$ be a protocol, and $l \in \mathbb{N}$. The *efficiency* of $\boldsymbol{\tau}$ at index l is defined as $\mathrm{eff}_{\boldsymbol{\tau}}(l) \coloneqq l/\max_{i\in[n]} t_l^i$. For finite protocols, the efficiency after the last defined index is 0.

Symmetric protocols in which all parties use the same converter are an important class of protocols: For constructions where both the given and the desired specifications are symmetric with respect to the parties, the interesting protocols are of this form. For this type of protocol, we show that the rate of the clocks is transformed in the expected sense.

Lemma 1. *Let C be a clock and $\boldsymbol{\tau}$ be a symmetric protocol, that is, $\boldsymbol{\tau} = (\tau, \dots, \tau)$ with $\tau = (t_l)_{l\geq 1}$. Then,*

$$\mathrm{rat}_{\boldsymbol{\tau}C}(k) = \mathrm{eff}_{\boldsymbol{\tau}}(t_v) \cdot \mathrm{rat}_C(k),$$

for all k with $\exists v \in \mathbb{N} :\ t_v = \min_{i\in[n]} \mathrm{loc}_C^i(k)$, that is, indices at which the "slowest party" is activated.

Proof: By Definitions 6 and 10, $\mathrm{rat}_{\boldsymbol{\tau}C}(k) = \min_{i\in[n]} \frac{\mathrm{loc}^i_{\boldsymbol{\tau}C}(k)}{k}$, and $\mathrm{loc}^i_{\boldsymbol{\tau}C}(k) = \min\{l \mid t_l \geq \mathrm{loc}_C^i(k)\}$. By the condition on k, $\mathrm{rat}_C(k) = t_v/k$ and $\mathrm{eff}_{\boldsymbol{\tau}}(v) = v/t_v$ for v as above. By monotonicity,

$$\min_{i\in[n]} \min\{l \mid t_l \geq \mathrm{loc}_C^i(k)\} = \min\{l \mid t_l \geq \min_{i\in[n]} \mathrm{loc}_C^i(k)\} = v,$$

which concludes the proof. ∎

For the indices which do not correspond to activations of $\boldsymbol{\tau}C$, we can still prove bounds for the rate.

Lemma 2. *For C and $\boldsymbol{\tau}$ as in Lemma 1,*

$$\frac{t_{l_k}}{t_{l_k+1}-1} \cdot \mathrm{eff}_{\boldsymbol{\tau}}(l_k) \cdot \mathrm{rat}_C(k) \leq \mathrm{rat}_{\boldsymbol{\tau}C}(k) \leq \mathrm{eff}_{\boldsymbol{\tau}}(l_k) \cdot \mathrm{rat}_C(k),$$

with $l_k \coloneqq \min\{v \mid t_v \geq \mathrm{loc}_C^i(k)\} = \mathrm{loc}^i_{\boldsymbol{\tau}C}(k)$.

Proof: By the definition of l_k,

$$\frac{\mathrm{loc}_C^i(k)}{t_{l_k+1}-1} \leq \frac{\mathrm{loc}^i_{\boldsymbol{\tau}C}(k)}{l_k} \leq \frac{\mathrm{loc}_C^i(k)}{t_{l_k}},$$

so $\mathrm{rat}_{\boldsymbol{\tau}C}(k) \leq \min_{i\in[n]} \frac{l_k}{t_{l_k}} \cdot \frac{\mathrm{loc}_C^i(k)}{k}$. The other inequality follows analogously. ∎

Corollary 3. *Let C be a clock and $\boldsymbol{\tau}$ be a symmetric protocol with constant efficiency, that is, $\mathrm{eff}_{\boldsymbol{\tau}}(\cdot) = \eta \in (0, 1]$. Then, $\mathrm{rat}_{\boldsymbol{\tau}C} = \eta \cdot \mathrm{rat}_C$.*

C. Constructions

A (clock transformation) protocol induces a natural notion of construction:[4] Denote by Ω the set of all clocks, and by Γ the set of all (clock transformation) protocols. Then we obtain, for $C_1, C_2 \in \Omega$ and $\boldsymbol{\tau} \in \Gamma$,

$$C_1 \xrightarrow{\boldsymbol{\tau}} C_2 \quad \Leftrightarrow \quad \boldsymbol{\tau}C_1 = C_2.$$

This definition extends to clock specifications as described in Section II: For each $C \in \mathcal{C}_1$, the constructed $\boldsymbol{\tau}C$ must be in $\mathcal{C}_2$. For specifications, however, a further notion of construction, denoted by $\longmapsto$, is interesting: For two specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, we can ask whether there is a protocol with at least efficiency η that constructs $\mathcal{C}_2$ from $\mathcal{C}_1$. More precisely, for $\mathcal{C}_1, \mathcal{C}_2 \subseteq \Omega$ and $\eta : \mathbb{N} \to [0, 1]$,

$$\mathcal{C}_1 \xmapsto{\eta} \mathcal{C}_2 \ \Leftrightarrow\ \exists \boldsymbol{\tau} \in \Gamma : \mathrm{eff}_{\boldsymbol{\tau}}(\cdot) \geq \eta(\cdot) \ \wedge\ \mathcal{C}_1 \xrightarrow{\boldsymbol{\tau}} \mathcal{C}_2.$$

V. Synchrony Amplification

In order to use synchronous protocols, one needs synchronized clocks that determine the round switches. This type of synchrony is described by the specification $\mathcal{C}^{\mathsf{sync}} = \mathcal{C}_1^{\mathsf{off}}$, and this section describes from which type of assumption such synchrony can be achieved.

Lemma 4. *Let $\delta_1, \delta_2 \in \mathbb{N}$ with $\delta_1 \geq \delta_2$.[5] From a δ_1-offset bounded clock C_1, we can construct a δ_2-offset bounded clock C_2 using a protocol τ with*

$$\mathrm{eff}_\tau(l) = \frac{l}{\lfloor (l-1) \cdot \delta_1/\delta_2 \rfloor + 1}.$$

Proof: We consider the symmetric protocol $\boldsymbol{\tau}_{\delta_1/\delta_2}$ in which each converter τ_i is defined by the (same) sequence $(t_l)_{l\geq 1}$ with $t_l = \lfloor (l-1) \cdot \delta_1/\delta_2 \rfloor + 1$. We

[4] For a cryptographic treatment in the spirit of [MR11], we also have to specify the behavior of clocks with respect to dishonest parties. Yet, if we specify that dishonest parties are activated by a clock for every index, then with simulators $\sigma = (1, 2, \dots)$ the construction $\longrightarrow$ is the same as that considered in [MR11].

[5] Of course, $\delta \in \mathbb{N}$ can be seen as the constant function $n \mapsto \delta$.

set $C_2 = \boldsymbol{\tau}_{\delta_1/\delta_2} C_1$. Note that this in particular means that

$$\lfloor (\mathrm{loc}^i_{C_2}(k) - 1) \cdot \delta_1/\delta_2 \rfloor + 1 \leq \mathrm{loc}^i_{C_1}(k) \tag{1}$$

$$\leq \lfloor \mathrm{loc}^i_{C_2}(k) \cdot \delta_1/\delta_2 \rfloor, \tag{2}$$

where both (1) and (2) use Definition 10 and (2) also uses $\delta_1 \geq \delta_2$.

Assume that there exists a $k \in \mathbb{N}$ with $\mathrm{off}_{C_2}(k) > \delta_2$. This means that there exist $i \neq j$ with

$$\mathrm{loc}^i_{C_2}(k) - \mathrm{loc}^j_{C_2}(k) = \delta_2 + 1. \tag{3}$$

Together, this means that

$$\mathrm{loc}^i_{C_1}(k) - \mathrm{loc}^j_{C_1}(k) - 1$$

$$\geq \lfloor (\mathrm{loc}^i_{C_2}(k) - 1) \cdot \delta_1/\delta_2 \rfloor - \lfloor \mathrm{loc}^j_{C_2}(k) \cdot \delta_1/\delta_2 \rfloor \tag{4}$$

$$= \lfloor \delta_2 \cdot \delta_1/\delta_2 \rfloor = \delta_1, \tag{5}$$

where (4) follows from (1) and (2), and (5) follows from (3): $\mathrm{loc}^i_{C_2}(k) - 1$ and $\mathrm{loc}^j_{C_2}(k)$ have the same remainder modulo δ_2. This equation contradicts the assumption about C_1, and the efficiency statement is obvious. ∎

Indeed, we can prove that the rate of the clocks is transformed as expected, which follows from Lemma 2.

Lemma 5. *For δ_1, δ_2, C_1, and $\boldsymbol{\tau}$ as in Lemma 4,*

$$\mathrm{rat}_{\boldsymbol{\tau} C_1}(\cdot) \geq \frac{\delta_2}{\delta_1} \cdot \mathrm{rat}_{C_1}(\cdot).$$

Lemmas 4 and 5 imply that we can obtain a round-synchronous clock from any bounded-offset clock, the rate degrades by a factor depending on the given clock.

Theorem 6. *For $\delta_1, \delta_2 \in \mathbb{N}$ with $\delta_1 \geq \delta_2$,*

$$\mathcal{C}^{\mathsf{off}}_{\delta_1} \overset{\delta_2/\delta_1}{\longmapsto} \mathcal{C}^{\mathsf{off}}_{\delta_2}.$$

In particular, $\mathcal{C}^{\mathsf{off}}_{\delta_1} \overset{\delta_1^{-1}}{\longmapsto} \mathcal{C}^{\mathsf{sync}}$.

We also prove a converse statement: If we only have a guarantee for the drift, the offset of the clock can grow exponentially (as a function of k). The following lemma states that one can obtain a round-synchronous clock only at the cost of a rapidly decreasing rate.

Lemma 7. *Let $\rho = \frac{p}{q} \in \mathbb{Q}$, $\rho > 1$, and let $\boldsymbol{\tau} = ((t^1_l)_l, \ldots, (t^n_l)_l)$ be a protocol that achieves $\mathcal{C}^{\mathsf{drf}}_\rho \xrightarrow{\boldsymbol{\tau}} \mathcal{C}^{\mathsf{sync}}$. Then, for $t^i_v \geq 2 \cdot q^2/p$, $t^i_{v+2} \geq \rho/2 \cdot t^i_v$.*

Proof: The specification $\mathcal{C}^{\mathsf{drf}}_\rho$ contains the clock C with $2 \in C_k$ for all $k \in \mathbb{N}$ and $1 \in C_k$ for $k = \lfloor j \cdot \rho \rfloor$ with some $j \in \mathbb{N}$. This clock has $\mathrm{drf}_C(k) \leq \rho$.

Assume that the protocol $\boldsymbol{\tau}$ with $\tau_1 = (t^1_l)_{l \geq 1}$ and $\tau_2 = (t^2_l)_{l \geq 1}$ is such that $\boldsymbol{\tau} C$ is synchronous. Hence, for each value t^2_v it must hold that

$$\min\left\{k \,\middle|\, \mathrm{loc}^1_C(k) = t^1_{v+1}\right\} \leq \min\left\{k \,\middle|\, \mathrm{loc}^2_C(k) = t^2_{v+2}\right\},$$

that is, party 1 switches to $v+1$ before party 2 switches to $v+2$. Hence, $t^2_{v+2} \geq \lfloor t^1_{v+1} \cdot \rho \rfloor$ for the clock C.

Of course, using the conversely defined clock $C' \in \mathcal{C}^{\mathsf{drf}}_\rho$, we can conclude that $t^1_{v+1} \geq \lfloor t^2_v \cdot \rho \rfloor$. Plugging the two equations together, we obtain that $t^2_{v+2} \geq \lfloor \lfloor t^2_v \rho \rfloor \rho \rfloor$. From this, the statement for the t^i_v can be easily computed. ∎

By the lemma, for a clock C with constant rate, $\boldsymbol{\tau} C$ has asymptotical rate $\log k/k$. The following theorem is a consequence of Lemma 7.

Theorem 8. *$\mathcal{C}^{\mathsf{drf}}_\rho \overset{\eta}{\longmapsto} \mathcal{C}^{\mathsf{sync}}$ only if $\eta(\cdot)$ vanishes exponentially.*

Together, Theorems 6 and 8 imply that it is also impossible to efficiently construct offset-bounded clocks for any constant from an arbitrary bounded-drift clock.

VI. Conclusion and Future Work

We have developed a formal model that allows to make statements about the synchrony of clocks in a simple yet rigorous manner, which we demonstrated by proving one "positive" and one "negative" statement about amplifying synchrony.

The current model does not allow for reactive resources such as communication channels; a full-fledged model that also covers this type of resource and can be seen as an extension of the model presented here is currently being developed. Yet, the statements proven here will be maintained.

References

[KLP05] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent composition of secure protocols in the timing model. In *STOC*, pages 644–653. ACM, 2005.

[KLSV03] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. Timed I/O automata: A mathematical framework for modeling and analyzing real-time systems. In *24th IEEE RTSS*, pages 166–177. IEEE, December 2003.

[Lam78] L. Lamport. Time, clocks, and the ordering of events in distributed systems. *C. ACM*, 21(7):558–565, July 1978.

[LV96] N. Lynch and F. Vaandrager. Forward and backward simulations—Part II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996.

[Mau11] U. Maurer. Constructive cryptography: A new paradigm for security definitions and proofs. In *TOSCA*, LNCS. Springer-Verlag, 2011.

[MR11] U. Maurer and R. Renner. Abstract cryptography. In *ICS*. Tsinghua University Press, 2011.