G-SINC: Global Synchronization Infrastructure for Network Clocks

Marc Frei*, Jonghoon Kwon*, Seyedali Tabaeiaghdaei*, Marc Wyss*, Christoph Lenzen†, and Adrian Perrig*

*Network Security Group, Department of Computer Science, ETH Zurich, Switzerland
Email: {marc.frei, jong.kwon, seyedali.tabaeiaghdaei, marc.wyss, adrian.perrig}@inf.ethz.ch
†CISPA Helmholtz Center for Information Security, Germany
Email: lenzen@cispa.de

***Abstract*—Many critical computing applications rely on secure and dependable time which is reliably synchronized across large distributed systems. Today's time synchronization architectures are commonly based on global navigation satellite systems at the considerable risk of being exposed to outages, malfunction, or attacks against availability and accuracy. This paper describes a practical instantiation of a new global, Byzantine fault-tolerant clock synchronization approach that does not place trust in any single entity and is able to tolerate a fraction of faulty entities while still maintaining synchronization on a global scale among otherwise sovereign network topologies. Leveraging strong resilience and security properties provided by the path-aware SCION networking architecture, the presented design can be implemented as a backward compatible active standby solution for existing time synchronization deployments. Through extensive evaluation, we demonstrate that over 94% of time servers reliably minimize the offset of their local clocks to real-time in the presence of up to 20% malicious nodes, and all time servers remain synchronized with a skew of only 2 ms even after one year of reference clock outage.**

***Index Terms*—Byzantine fault tolerance; clock synchronization; multipath communication**

I. Introduction

Secure and dependable time synchronization is an essential prerequisite for many industries with applications in finance, telecommunication, electric power production and distribution, or environmental monitoring.

Current best practice to achieve large-scale time synchronization relies on Coordinated Universal Time (UTC) as a global time standard which is distributed within multiple, hierarchical synchronization networks from a set of primary time servers to every end host in the system. The Network Time Protocol (NTP) [43] is a commonly used protocol for this purpose. Global Navigation Satellite Systems (GNSSes) are the most practical and cost effective sources of UTC as a reference time in this architecture.

Alternatives to GNSSes as reference clocks for UTC exist, but these more specialized methods are primarily used in purpose-built solutions with very specific use-cases:

- Multi-Source Common-View Disciplined Clock (MSCVDC) is a recent design to support critical infrastructure systems that require fault-tolerant clock synchronization. In its current stage of development, MSCVDC makes use of GNSSes (and a central cloud service). Conceptually it would be feasible to augment the implementation with additional time sources, possibly even based on terrestrial transmitters. But it remains to be explored in how far the strict dependency on GNSSes could be removed for deployments beyond geographically restricted areas [52], [40].
- Two-Way Satellite Time and Frequency Transfer (TW-STFT) provides significantly better synchronization quality than one-way communication with GNSSes at much higher complexity and cost [65]. Like GNSSes, TW-STFT relies on satellite infrastructure which represents a potential source of disruption. The system also poses considerable unsolved scalability challenges to support more than a few tens of clients [53].
- Time distribution over dedicated fiber optic links does not come with a dependency on satellites. At the same time this technology is not flexible enough to synchronize a potentially large and evolving set of primary time servers around the globe [11], [27].
- Physical transport of high-accuracy clocks can be a viable synchronization method but it is even more limited in its application than dedicated fiber optic links [22].

We also note that installation of atomic clocks at every site to be synchronized is not sufficient to avoid dependencies on GNSSes. Atomic clocks can only serve as accurate and highly stable oscillators. As so-called frequency standards they cannot provide UTC by themselves but have to be continuously synchronized with external time sources [39]. Ongoing synchronization is required to compensate for nonidentical frequency drifts among separate clocks. Moreover, due to occasional (positive or negative) leap second insertions, UTC is not a fixed time scale that can be computed locally without external coordination.

Based on this short survey we conclude that there currently exists no practical alternative to GNSSes to receive UTC as a global reference time. In the context of critical infrastructure this means a significant risk since GNSSes exhibit a number of potentially severe vulnerabilities such as equipment failure [25], misconfiguration, malicious manipulation by attackers [51], and due to more localized spoofing or jamming attacks [69], [26]. Natural disasters like, for example, solar superstorms could also hit and impact GNSSes [30].

2575-8462/22/$31.00 ©2022 IEEE
DOI 10.1109/SRDS55811.2022.00021

Complete reliance on GNSSes in the traditional NTP architecture can be all the more serious in its consequences because primary time servers influenced by malfunctioning GNSSes will in turn affect the entire synchronization topology, and no fallback plan exists. It is apparent that time synchronization solely based on GNSSes does not fulfill fundamental dependability requirements for systems that serve indispensable functionalities in our society.

Related to these concerns, in 2020 NIST received a mandate to investigate possible approaches to the *deliberate, risk-informed use of positioning, navigation, and timing services* with the goal of supporting the needs of critical infrastructure owners and operators in the public and private sectors [73]. The resulting report [65] indicates an increasing awareness for this problem and highlights the importance of resilient time distribution. Manufacturers of time synchronization equipment are encouraged to further explore solutions in this space. In addition to the NIST work, there are a number of other publications that call attention to the high degree of dependence on GNSSes and the resulting economic impact in different industries [38], [49], [41], [72], [57], [14], [50], [40].

Aiming to address the issues raised above, this paper describes G-SINC: a novel, Byzantine fault-tolerant clock synchronization approach as a fully backward compatible extension of current time synchronization architectures. G-SINC is executed among primary time servers across previously separate synchronization hierarchies and allows these top-level servers to reliably detect inconsistencies between time measurements retrieved via GNSSes and the globally synchronized time maintained by G-SINC. In addition, the globally synchronized timing information can be used along the GNSS reference time as a redundant external time source for local clock corrections with the following qualities:

- formally proven synchronization with real-time (UTC) for non-faulty nodes under normal conditions, with the accuracy of an arbitrary unreliable reference, e.g., a GNSS; this includes full recovery after transient faults of references;
- formally proven upper bound on the clock offsets among non-faulty nodes even under extreme conditions, where GNSSes are unavailable or cannot be trusted.

Our approach thus constructs a two-tier structure, where the top-level time servers run the global G-SINC algorithm and all other intermediate time servers as well as end hosts then synchronize with their respective upstream providers as in currently deployed time synchronization hierarchies.

An additional contribution of this paper is the application of secure multipath communication in path-aware networking architectures [75], [19], [2], [5], [4] to improve fault tolerance and defend against on-path adversaries. It thereby helps to further advance methods to prevent time shifting attacks, orthogonal to approaches like Chronos [9], [63]. We demonstrate this in an extensive evaluation based on a simulator framework that supports multipath communication at Internet scale. Our experiments show that over 94% of time servers reliably minimize the offset of their local clocks to real-time even under the presence of 20% malicious nodes, and that all time servers remain internally synchronized with a skew of at most 2 ms after one year of reference clock outage.

To implement the system, we make use of the SCION next-generation Internet architecture [4], [7], which provides several mechanisms to realize the desired system properties. Besides the possibility to use multiple distinct paths in parallel, we highlight the fact that SCION paths are reversible and therefore symmetric. Hence, they help to increase time synchronization precision compared to clock offset measurements over the often asymmetric paths in today's Internet.

II. Background

A. Time Synchronization

The goal of time synchronization is to limit the relative offsets among clocks to an acceptable range. Due to the intrinsic error of clocks, computing devices usually cannot rely on an agreed upon clock configuration at a single point in time, but instead need to correct their clocks in a periodic manner.

Time synchronization algorithms can be divided into two categories: In *external* algorithms, nodes synchronize with an independent outside clock source, while in *internal* algorithms, nodes synchronize among themselves to an internally generated common time, e.g., to a leader or an averaged time.

Even though GNSS-based time synchronization provides high accuracy, it is not affordable to every system due to its relatively high cost, operational complexity, and energy consumption. As an alternative, nodes can use the network infrastructure to synchronize their clocks to a reference server. The most widely used clock synchronization protocol is NTP [43]. It approximates the relative clock offset between an NTP client and an NTP server by exchanging timestamps over the network. The underlying assumption is that the round-trip delay divided by two is close to the one-way-delay in each direction. Accurate time synchronization can therefore only be achieved when the one-way-delay of the packet's forward and backward path do not differ much, where the synchronization error amounts to half the difference between the forward and backward travel times.

In today's Internet, routing asymmetry, i.e., the phenomenon that paths in the forward and backward direction are different, is a widely observed phenomenon [21], [8], [59]. But even on a symmetric path, messages can encounter different buffer times [15], [37], and random wire delay as well as software timestamping inaccuracies are another source of latency asymmetry [31]. While NTP is simple and easy to adopt, its accuracy in large-scale deployments is therefore relatively low, meaning that it is limited to synchronization on the order of milliseconds even under ideal conditions [48]. NTP's limiting factors (with network asymmetry as a dominant source) are discussed further in [54].

B. Network Stability and Security

The correct functioning of network-based clock synchronization algorithms depends on the properties of the network architecture. In case of the public Internet, the Border Gateway

Protocol (BGP) is responsible for exchanging routing information among autonomous systems (ASes). However, considering BGP's slow convergence after network failures [34], [24] as well as its lack of path stability and security, reliable and precise global clock synchronization is elusive in the current BGP-based Internet. From those shortcomings, the next-generation architecture SCION [4], [7] emerges.

As a clean-slate network architecture, SCION has been designed with a focus on reliability and security. In contrast to BGP, network failures or misconfigurations cannot result in global outages, route hijacking is prevented by design, and network-based DDoS attacks are efficiently mitigated. Furthermore, SCION allows for trust heterogeneity and supports multipath communication for end hosts.

A fundamental building block of SCION is the concept of Isolation Domains (ISDs). An ISD constitutes a logical grouping of ASes with a uniform trust environment or a common jurisdiction. Each ISD defines its own roots of trust and policies through a trust root configuration (TRC), combining a signed collection of certificates and policy specifications. Within an ISD, the routing process is isolated from external attacks and misconfigurations. A subset of ASes in an ISD, called core ASes, maintain connections to other ISDs. Path exploration inside an ISD is separate from the routing process between ISDs; an end-to-end path is hence assembled by the source through (up to) three path segments: an up-segment from a regular AS to a core AS, a core-segment between core ASes, and a down-segment from a core AS to a regular AS.

Besides the multitude of apparent advantages, we also designed our system on top of SCION due to its real-world deployment [1], [32], [66], [67], [68], its open-source implementation [64], and its global research testbed [33].

C. Byzantine Fault Tolerance

Achieving global clock synchronization is a difficult task notably also because of the questionable trustworthiness of different actors. Some synchronization peers may have malicious intentions and for example report wrong time information. But even benign peers can accidentally misconfigure their infrastructure, suffer unexpected faults, or get compromised.

The Lynch-Welch algorithm [74] allows initially synchronized nodes in a fully connected network to avoid drifting apart and thus to keep closely synchronized local times, while being able to tolerate up to just under one third Byzantine node faults. With n participating nodes, the algorithm can therefore sustain f Byzantine node faults, as long as $n \geq 3f + 1$. Byzantine faults refer to the most general type of faults, meaning that a Byzantine node can be in an arbitrary state and send arbitrary messages at any time. The algorithm executes in iterative rounds, where at the beginning of each round the nodes broadcast a message to all other nodes. Every node then waits for a certain time period that depends on the maximum message transmission delay, in order to collect the arrival times of all messages from its synchronization peers, which it stores as a sorted array A. Each node then applies a fault-tolerant midpoint calculation, whose output is subsequently used to correct its local clock. The function discards the f smallest and the f largest values of A and computes the arithmetic mean of the minimum and maximum of the remaining values. This whole procedure is then repeated after a certain time P, which marks the beginning of the next synchronization round.

III. Design Principles

A. Goals and Challenges

The fundamental goal of this work is for collaborative networks without complete mutual trust to introduce a reliable time synchronization infrastructure. However, our intention is not to introduce a new time synchronization protocol. Various such protocols achieving high accuracy have already been introduced. The main obstacle for them to achieve global time synchronization is the hierarchical synchronization structure stemming from a single global root of trust: a lack of resilience to abnormal input from the root. To overcome this, we seek the following architectural design: (1) multi-source time synchronization rather than a single trust root; (2) a distributed time synchronization structure that scales to the Internet; and (3) a flexible architecture where existing time synchronization protocols can be leveraged. This design raises the following research challenges.

Fault Tolerance. Multi-source time synchronization decreases the structural weakness of single root-driven time synchronization. That is, each node collects and analyzes various inputs from multiple time sources, calculates a consolidated clock offset, and performs local clock correction. This aims to guarantee resilience against arbitrary faults or malicious behavior by a subset of entities.

Load Distribution. In the context of Internet-scale multi-source time synchronization it is impractical for each node to synchronize with all other nodes. Therefore, for scalable multi-source time synchronization, (1) the network needs to be hierarchically layered according to the requirement of synchronization accuracy, and (2) network segmentation is needed to distribute the load required for the entire global time synchronization.

Backward Compatibility. In order for a new global time synchronization approach to be rapidly deployed and ready to use, architectural continuity with existing time synchronization systems is required. By utilizing already established time synchronization resources (e.g., GNSSes, servers, and protocols), architectural continuity with the existing system and the new goal, reliable global time synchronization, are simultaneously achieved.

B. System Model

Topology. We model a distributed system as a graph $G = (V, E)$, where V is the set of nodes, and E is the set of bidirectional communication links. An unknown subset $F \subset V$ of the nodes is (Byzantine) faulty. Faulty nodes may violate the protocol in an arbitrary manner. In the following, denote by $V_g := V \setminus F$ the subset of *correct* nodes. Each node v can send network packets to its neighbors $N_v := \{u \in V | \{v, u\} \in E\}$. For $u \in N_v$, v could have parallel links

that can be distinguished by means of an interface identifier or a port number. A *path* is a sequence of nodes and links which are all distinct, i.e., $p = v_0, e_1, ..., v_{k-1}, e_k, v_k$ where $e_i = \{v_{i-1}, v_i\}, 1 \leq i \leq k$. Non-adjacent nodes $v, w \in V$ need to rely on the internal nodes of one or more paths from v to w to communicate.

Latency. Communication suffers from delay, which might vary due to link congestion, queuing delay, and process scheduling, ranging in $[d-u, d]$. Precise bounds on the (maximum) *delay* d and the delay *uncertainty* u might be hard to determine and these parameters might be different for different links in practice. However, note that we just ask that delays are between $d-u$ and d, so overestimating these parameters is acceptable.

Hardware Clocks. To measure the progress of time, each node v is equipped with a hardware clock, which we model by an increasing function $H_v \colon \mathbb{R}_0^+ \to \mathbb{R}_0^+$ mapping the real time t to the *local time* $H_v(t)$ at v at time t:

$$H_v(t) = H_v(0) + \int_0^t h_v(\tau)\, d\tau, \tag{1}$$

where $H_v(0) \in \mathbb{R}_0^+$ is the *hardware offset* and $h_v(\tau)$ is the clock rate of v's hardware clock. We stress that v has no access to t; it can only measure the progress of time by querying and storing $H_v(t)$. Thus, the hardware clock rate $h_v(\tau)$ during a time interval $[t, t']$ determines how far measuring $t'-t$ via evaluating $H_v(t') - H_v(t)$ is off the mark. In general, h_v may vary over time depending on environmental conditions such as the ambient temperature, the stability of the supply voltage, or crystal quartz aging. We assume that the *clock drift* relative to real time is bounded by $\vartheta - 1 \ll 1$, i.e.,*

$$\forall v \in V\ \forall t \in \mathbb{R}_0^+\ : 1 \leq h_v(t) \leq \vartheta. \tag{2}$$

A relatively cheap quartz oscillator clock exhibits a drift of less than $20\,\mathrm{ppm}$ [70] which corresponds to around 1.7 seconds per day, while a higher-end rubidium clock may have a drift as small as $8\,\mathrm{ms}$ in a year [44].

Synchronization Requirements. A clock synchronization algorithm computes at each correct node $v \in V_g$ a logical clock $L_v := \mathbb{R}_0^+ \to \mathbb{R}_0^+$. "Good" logical clocks should behave as closely as possible to ideal clocks.† However, since it is impossible to track the real time t perfectly, different features of such ideal clocks result in different, competing requirements:

- **accuracy:** minimize $\mathcal{A}(t) := \max_{v \in V_g}\{|L_v(t) - t|\}$;
- **skew:** minimize $\mathcal{G}(t) := \max_{v,w \in V_g}\{L_v(t) - L_w(t)\}$; and
- **bounded rates:** minimize μ s.t. $\forall t \in \mathbb{R}_0^+, \forall v \in V_g$: $(1-\mu)h_v(t) \leq \ell_v(t) \leq (1+\mu)h_v(t)$, where $\ell_v := \frac{dL_v}{dt}$.

*For notational convenience, we assume a one-sided error here. If H_v satisfies $1-\rho \leq h_v(\tau) \leq 1+\rho$ for some $\rho \ll 1$ and all τ, then $H'_v(t) := H_v(0) + \int_0^t h_v(\tau)/(1-\rho)\, d\tau$ meets our requirements for $\vartheta = (1+\rho)/(1-\rho) \approx 1+2\rho$.

†The term "logical clock" is used here as a common term in the clock synchronization literature. Introduced, e.g., in [35], it is defined as the value of a hardware clock plus some correction value.

Reference Time. To achieve bounded accuracy, the nodes need some access to a reference tracking real time with bounded error. While, both from the viewpoint of philosophy and physics, it is not clear how "real" time should be defined, in the context of our work we consider UTC to equal the real time t. The nodes learn about t via oracle access to a clock reference, which could be implemented by receiving time information from one or multiple GNSS services. Under regular conditions, the oracle function r_v at v evaluated at time t satisfies that $|r_v(t) - t| \leq \varepsilon$, where $\varepsilon > 0$ can be expected to be much smaller than u. In this case, simply regularly querying the oracle and interpolating between the returned values yields a logical clock with small μ, where accuracy and skew are in $O(\varepsilon)$, because $L_v(t) - L_w(t) \leq |L_v(t) - t| + |L_w(t) - t| \leq 2\mathcal{A}(t)$ for all $v, w \in V_g$ and t. However, there is no guarantee that r_v behaves this way at all times.

Objective. The task of the clock synchronization algorithm is to make a best effort in leveraging r_v, in that we want to achieve $\mathcal{A}(t) = O(\varepsilon) \ll u$ when the oracle is reliable, while $\mathcal{G}(t)$ remains bounded even if r_v misbehaves arbitrarily. Moreover, we want $\mathcal{A}(t)$ to become small again after a temporary failure of the oracle. Note that, on the upside, the requirement of bounded rates ensures that a temporary failure of the oracle does not result in a large deviation of logical clocks from UTC. However, it also means that the duration to return to the correct time once the oracle functions correctly again must be proportional to the accumulated error as well. This is a deliberate design decision: applications that rely on a trusted time reference are likely to expect the reference time to not deviate significantly from its nominal rate.

C. Threat Model

We assume that the adversary can compromise fewer than one-third of primary time servers. The adversary has full control over its territory (i.e., compromised entities and corresponding links) and can eavesdrop, inject, intercept, delay, and alter the on-path packets with negligible latency inflation. Besides the presence of an active attacker, we also consider network failures due to, e.g., link congestion, misconfiguration, and physical errors that hamper reliable clock synchronization. A detailed discussion of the implications of the SCION multipath architecture in our threat model is presented in Section IV-F.

IV. Architecture Design

To achieve reliable time synchronization at global scale, we propose G-SINC. This section describes architectural design details, including topology layout, group membership, and the novel multi-source synchronization algorithm.

A. Architecture Overview

G-SINC augments the traditional NTP architecture with the core concept of a Byzantine fault-tolerant algorithm to synchronize independent NTP networks. That is, G-SINC ensures reliable and accurate time synchronization among primary time servers of each NTP network while the existing

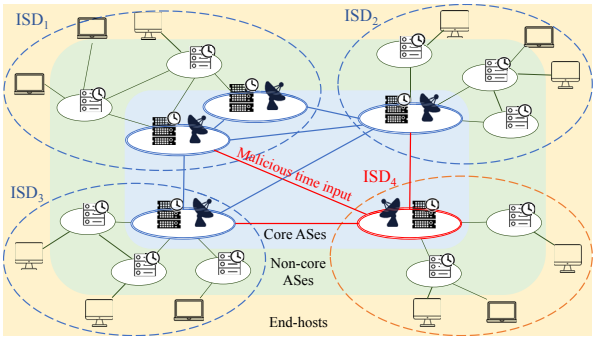


Figure 1: G-SINC architecture overview. Core ASes construct a core time synchronization network while each network cluster (ISD) operates in the traditional NTP structure.

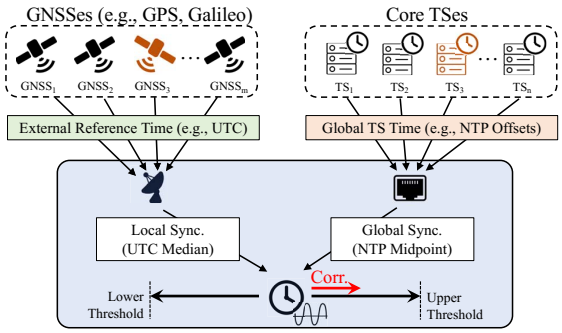


Figure 2: A core TS synchronizes its local clock with two types of time sources: external reference times from GNSSes and NTP offsets to other core TSes.

NTP network structure follows the established design. To this end, we summarize the key characteristics of our topology design as follows:

- *Hierarchical Partitioning*: We take advantage of the AS layering in SCION to form a two-tier structure. Core ASes run a global peer-to-peer synchronization algorithm to achieve Byzantine fault tolerance. All other ASes synchronize their time with the upstream providers to ensure scalability.
- *Clustering*: The collaborative networks are clustered into ISDs based on their trust relationship. The network clustering maintains the sovereign operation of ISDs, even if external entities (e.g., GNSS providers) supply erroneous values, experience outages, or attempt to interfere. An ISD can still maintain internal time synchronization among its ASes and other ISDs operating correctly, as long as paths through correctly operating ISDs exist.
- *Logical N-to-N Peering*: Each core AS is virtually peered with all other core ASes, securing multiple network-based time sources.

In G-SINC, each core AS operates one or more primary time servers (TSes). Figure 1 illustrates the basic topology. These core TSes also act as primary servers in the traditional NTP architecture and they typically use one or more GNSSes as their external time reference providing time values in UTC. We assume that core TSes are equipped with local clocks driven by high-quality oscillators with an autonomous free run accuracy of $\pm 10\,\mathrm{ms}$ (ca. $0.3\,\mathrm{ppb}$) or better after one year.

B. Mutli-Source Time Synchronization

The general strategy of our synchronization algorithm is to periodically compute an approximate agreement on clock correction values at each core TS based on the relative clock offsets to its peers. Core TSes then correct their local clocks towards the approximate global clock value. In addition to these periodic global synchronizations, each core TS synchronizes the local clock with its local reference clocks. These local corrections are applied more frequently than the global clock corrections. By encapsulating the local synchronization in a separate process, we get an abstracted local clock that follows its reference clocks as long as these exhibit benign behavior. The abstraction also guarantees a maximum drift that is always within a constant factor of what the underlying hardware clock guarantees. Figure 2 illustrates this design.

The combination of global clock synchronization with the local synchronization process at each core TS yields a globally synchronized time that is not only internally synchronized across all core ASes, but also externally synchronized to UTC as long as the UTC time values provided by the reference clocks are consistent with the globally synchronized time. If the offset between a local clock and the globally synchronized time exceeds a predefined threshold (on the order of the expected measurement error over the network) the TS follows the globally agreed upon time to a larger degree than its reference clocks until the discrepancy is resolved.

Byzantine fault tolerance properties are achieved by deriving the algorithm from the extensively studied clock synchronization algorithm by Lynch and Welch [74], see also Section II-C. The key idea is that in each round every participating node collects an array of relative clock offsets to each peer. Based on this data, a fault-tolerant midpoint (or averaging) function is computed, resulting in a global approximate agreement on the relative time differences among the core TSes [10]. With this algorithm, the system is in theory able to tolerate faults or malicious behavior of up to one third of the nodes in the set of participating core TSes.

C. Consensus on Membership

A requirement for the core synchronization algorithm is that all core TSes agree on the same set of core TSes to synchronize with. In SCION it is possible to provide TSes in the core network with a consistent enough view of all core ASes in the network to satisfy this requirement in practice. So-called trust root configurations (TRCs) are disseminated among all ISDs as part of the core beaconing (a fundamental network functionality in SCION to construct path segments among core ASes within an ISD and across ISDs). Each TRC consists of a signed collection of cryptographic information entries,

Algorithm 1 Local Clock Synchronization

```
Input
    I   Interval between local clock synchronizations
    X   Local sync. impact factor (X > 1)
Algorithm
    maxCorr ← X · LocalClock.MaxDrift(I)
    refTime ← ReferenceClock.Time()
    LocalClock.SetTime(refTime)
    while true do
        refTime ← ReferenceClock.Time()
        locTime ← LocalClock.Time()
        loff ← refTime - locTime
        if |loff| > 0 then
            corr ← sgn(loff) · min(|loff|, maxCorr)
            LocalClock.AdjustTime(corr, I)
        end if
        LocalClock.Sleep(I)
    end while
```

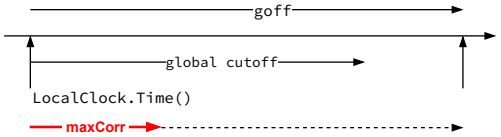


Figure 3: Global Clock Correction

including a list of the core ASes in a given ISD. The public-key infrastructure (PKI) of SCION precisely defines policies, roles, and procedures covering verification, update, and revocation of TRCs as well as recovery from catastrophic events. The set of all core ASes can therefore be maintained based on TRCs without introducing additional mechanisms besides what is already provided by SCION's control-plane PKI. Since every core AS has to go through an official approval process to be included in a TRC, we also substantially reduce the risk of possible Sybil attacks [12] as a common threat in peer-to-peer settings.

D. Local Clock Synchronization

Each core TS internally runs a process that synchronizes the local clock with the connected external reference clocks, see Algorithm 1. The local clock and the external reference clocks are abstracted into separate modules: *LocalClock* and *ReferenceClock*. Every synchronization round begins by measuring the accumulated local offset *loff* as the difference between *ReferenceClock.Time()* and *LocalClock.Time()*. If $|loff| > 0$, the corresponding correction value is computed and applied to *LocalClock*. However, this correction is limited to *maxCorr* in each round, so that a faulty or malicious reference cannot manipulate the local clock arbitrarily. We set *maxCorr* to the maximum drift the local clock may experience, scaled by a constant coefficient $X > 1$, to make sure that the reference clock is able to pull the local clock towards itself even if the local clock drifts maximally in the opposite direction. Corrections are thus capped by a function of the maximum expected time drift of the local hardware clocks. This important parameter is well-documented by the manufacturers of the intended clock sources and can also be independently tested. The local clock synchronization process repeats at interval I.

Algorithm 2 Global Clock Synchronization

```
Input
    N   Number of nodes
    F   Number of faulty nodes (N >= 3F + 1)
    P   Set of synchronization peers
    J   Interval between global clock
        synchronizations (J >= I)
    G   Global cutoff
    Y   Global sync. impact factor (Y > X + 1)
Algorithm
    maxCorr ← Y · LocalClock.MaxDrift(J)
    while true do
        M ← [0]                    ▷ Array of NTP offset measurements
        for p ∈ P do
            m ← PathAwareNTPOffset(p)
            M ← M + [m]
        end for
        M ← sort(M)
        goff ← (M[F] + M[N − 1 − F]) / 2
        if |goff| > G then
            corr ← sgn(goff) · min(|goff|, maxCorr)
            LocalClock.AdjustTime(corr, J)
        end if
        LocalClock.Sleep(J)
    end while
```

E. Global Clock Synchronization

In parallel to the local synchronization process, a global synchronization algorithm is executed, which ensures that the clocks of all core TSes stay internally synchronized. Essential parameters of Algorithm 2 are defined as follows: N is the number of core TS nodes participating in the synchronization and F is the assumed maximum number of faulty nodes on which the Byzantine fault tolerance argument is grounded. Clock synchronization is performed in rounds. The constant J specifies the time interval between global synchronizations. As for the local synchronization, *maxCorr* stores the maximum correction value that we are willing to apply within one synchronization round. This value has to be chosen large enough so that it can compensate for the entire offset that the local synchronization may introduce over the interval J. This amounts to the maximum expected clock drift over the interval J plus the term $X \cdot LocalClock.MaxDrift(J)$ which is the maximum correction towards the reference clock that can accumulate on top of the local clock's intrinsic drift during the given interval J. Taken together this results in the scale factor Y $(> X + 1)$ for the maximum clock drift.

The array *off* collects the relative clock offsets to the local TS itself and to every peer TS as measured by the the function *PathAwareNTPOffset*, see Algorithm 3 and Section IV-F. Since the offset of any given local clock to itself is 0, *off* is initialized with the value 0. The relative clock offsets to the $N - 1$ peers are appended subsequently. After sorting *off*, the approximate agreement for the global clock offset *goff* is

Algorithm 3 Path-Aware NTP Offset Computation (PATHAWARENTPOFFSET)

```
Input
    p    Synchronization peer
Algorithm
    D ← SCION.GETDISJOINTPATHS(p)
    M ← []                          ▷ Array of NTP offset measurements
    for d ∈ D do
        t0, t1, t2, t3, ok ← NTP.MEASURE(p, d)
        if ok then
            m ← ((t1 − t0) + (t2 − t3)) / 2
            M ← M + [m]
        end if
    end for
    off ← 0
    if |M| > 0 then
        off ← median(M)
    end if
    return off
```

computed by applying the Byzantine fault-tolerant midpoint calculation proposed by Lynch and Welch [74].

The resulting clock adjustment is controlled based on *goff*. If $|goff|$ lies beyond the global cutoff value G, a clock correction towards *goff* is computed so that the correction keeps the previously computed direction of *goff* but never exceeds *maxCorr*, see Figure 3. Otherwise, we don't apply the global clock correction. G is assigned a value on the order of the NTP measurement accuracy.

LocalClock.Sleep at the end of the loop suspends execution for the specified duration taking possible time adjustments issued previously via *LocalClock.AdjustTime* into account.

Formal Analysis. An in-depth mathematical analysis of the synchronization routines is provided in [17]. The first key result states that even in a worst-case scenario where the reference clocks can behave arbitrarily, the G-SINC clock skew is upper-bounded by 4δ (plus negligible terms), where δ denotes the maximum expected offset measurement error over the network. A second theorem in the analysis guarantees that under normal conditions, the skew is bounded by the fourfold of the local clock's maximum drift during the synchronization interval I.

F. Path-Aware Networking

A central concept of the SCION networking architecture is comprehensive path transparency and control that enables senders to simultaneously select multiple paths to carry packets towards the destination. In general, this multipath communication capability can be used to optimize bandwidth and latency as well as to enhance overall availability due to increased resilience against link failures, or to avoid untrusted infrastructure along the way. In the specific application of global time synchronization, multipath communication enables designing a system that is able to closely approximate optimal accuracy while also improving security and fault tolerance of the synchronization process even over the public Internet.

Path Selection and Symmetry. When a sender is creating a packet to be sent over the network, it first queries a set of paths to the target end host. These paths are discovered and disseminated by the SCION control plane based on individual path segments at the level of ASes. At the end host path segments are then combined into actual end-to-end paths. For typical network topologies it is expected that the result set for a path query to a given target end host will consist of up to a few dozens of paths, especially between core ASes. The sender will thus select one or more paths out of the set of available paths, which can be used simultaneously even from endpoints connected by a single link. For each selected path, the sender creates packets that include path information in their headers as packet-carried forwarding state (PCFS) [4], [71]. The destination host receiving those packets can either reply back to the source by fetching its own set of paths, or by reversing the PCFS information of the received packets. The latter approach allows the response packets to traverse the same path but in the backwards direction. This path symmetry can help reducing the latency variance between NTP requests and responses, leading to potentially higher measurement accuracy.

On-Path Adversaries. The clock synchronization algorithm proposed by Lynch and Welch (Section II-C) allows to tolerate up to just under one-third arbitrary, i.e., Byzantine, faulty processes. This holds under the assumption that the communication network is fully connected, and hence every process can communicate directly with all other processes. This assumption is not met for the Internet however, where ASes are connected to others via many further (potentially malicious) ASes. In order to account for this fact, we have to consider one of two endpoint ASes as malicious when the path between them is at least partially controlled by an attacker.

Threat Mitigations. By leveraging a multipath-aware networking substrate, we strive to approximate a fully connected network by decreasing the impact that on-path adversaries can have on offset measurements. First, paths via ASes considered untrustworthy can be filtered out and not be used for the NTP measurements. Second, by only considering the median of NTP measurements conducted over multiple paths in Algorithm 3, many paths need to be compromised in order for an attacker to shift the resulting NTP offset. To avoid letting a single malicious AS influence multiple paths, disjoint, i.e., non-overlapping, paths are preferred. In some cases, choosing a single highly trusted path or a single path with the least number of ASes, or random selection might serve as an alternative strategy. Given the topology dependence, we can only provide average-case numbers obtained though large-scale simulations to quantify the effect of faulty or adversarial network entities in the threat model. Our simulations show that, in realistic topologies, over 99% of nodes in the core network maintain close synchronization with real-time in the presence of up to 10% faulty nodes; with 20% faulty nodes, over 94% stay closely synchronized, see Section V-B. A thorough theoretical understanding of how general network topologies map to the best achievable resilience is an open research problem. Our approach allows to tackle this subproblem in a modular way as future work.

Synchronization Within a NTP Cluster. With its multi-source synchronization algorithm among primary TSes G-SINC avoids complete reliance on GNSSes at the top-tier of the traditional NTP architecture. To achieve reliable end-to-end synchronization, intermediate (i.e., non-primary) TSes and end-hosts also need to defend against faults and malicious actors on the path to upstream TSes. In particular, the architecture also has to cope with malicious primary TSes at the top of a hierarchy which could break synchronization of downstream nodes. We approach this part of the problem by combining the previously proposed Chronos mechanism [9], [63] with path-aware networking concepts introduced in this section. Chronos is specifically designed to prevent time-shifting attacks in the presence of Byzantine faults in a NTP network. Enhanced with multipath support, this strategy is expected to result in significantly improved fault-tolerance compared to today's synchronization hierarchies based on unmodified NTP clients.

NTP Message Authentication. To guarantee that only messages from verified members of the core synchronization group are processed, it is necessary to authenticate the request and response packets exchanged during the NTP-based offset measurements. NTP provides Network Time Security (NTS) as a cryptographic security mechanism for NTPv4 via extension fields in the NTP packet format and the NTS Key Establishment protocol (NTS-KE) to create and manage the corresponding key material between NTP clients and servers [16]. It is entirely possible to use NTS over SCION and we successfully tested this in prototype deployments. The SCION architecture offers however a more lightweight and therefore preferred alternative in the form of DRKey [62], which enables highly efficient authentication of all message exchanges on the network layer.

V. Evaluation

In this section, we evaluate the reliability of G-SINC under various proportions of malicious in-network attackers. We then compare different path selection strategies and the number of paths being used in the core time synchronization. In addition, we investigate the availability of G-SINC without GNSSes, demonstrating resilience to long-term failure of the reference clocks. At last, we analyze the additional scalability requirements for core time servers participating in G-SINC.

A. Experimental Setup

Despite the ever-expanding commercial deployment of SCION in the past few years [32], the global production network is not yet sufficiently diverse for our experiments. Therefore, we conduct our experiments by simulating G-SINC on realistic inter-domain topologies consisting of either the 2000 or the 500 highest-degree Tier-1 and Tier-2 ASes, extracted from the CAIDA AS relationships with geographic locations data set [6]. To this end, we have developed an ns-3-based simulator [55] that supports the SCION control and data plane. We conduct all experiments on the topology of 2000 ASes except long-term (one year) experiments, where we conduct them on the topology of 500 ASes due to execution time constraints.

The control plane provides ASes with diverse paths to any destination AS. Since the topology is massively interconnected, the longest distance between any pair of ASes is five. Each AS stores at most 60 different paths per destination AS. In the data plane, we simulate propagation delay by calculating the great circle latency between border routers of an AS using their locations specified by the CAIDA AS data set. For the propagation delay between two adjacent routers at the same location we assume a 1 meter long optical fiber between them. Furthermore, we simulate the transmission delay by assuming 400 Gbps links between all border routers, and queuing delay by assuming each router can process 5 Gpps. We run the simulations in a compute cluster on 64 CPU cores with 32 or 120 GB of memory, depending on problem size.

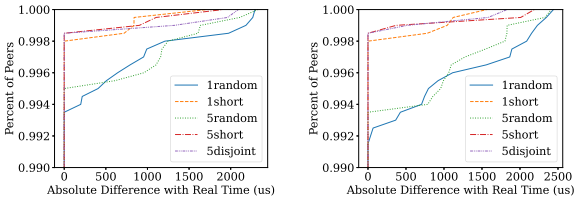
B. Reliability Analysis

To evaluate the reliability of G-SINC, we specifically consider an on-path delay attacker model among the various attack methods, because a simple authentication scheme can mitigate other attacks. Furthermore, an off-path delay attack requires complex preceding attacks (e.g., BGP hijacking) manipulating the routing infrastructure to introduce arbitrary asymmetric delays in packet exchanges, which is impossible in path-aware networking environments such as SCION.

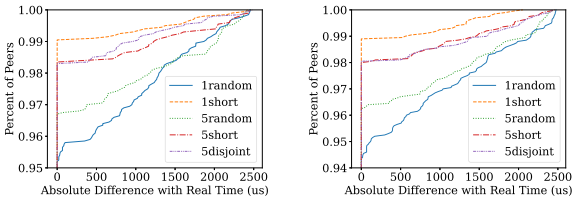
An on-path attacker, however, can impact the packets passing through nodes under its control. It is possible to enlarge the attack influence by strategically controlling core ASes with the highest degree of connectivity, but compromising such ASes is assumed to be very difficult in realistic scenarios. In addition, ASes in SCION can quickly switch the forwarding path over a different route as soon as they observe any suspicious behavior of the compromised ASes. Driven by this, we consider a botnet-size attacker distributed uniformly at random throughout the network.

We conduct 20 different experiments on the topology of 2000 ASes with a combination of four different attacker populations (i.e., 5, 10, 15, and 20%) and five different path selection strategies for non-neighboring ASes (i.e., k shortest, disjoint, and random paths). In all the experiments, each AS is assigned with a uniformly and independently random drift in a range of $\pm$ 27 μs per day. The global cutoff is $1\,\mathrm{ms}$, *LocalClock.MaxDrift* is $27\,\mu\mathrm{s}$ per day, and the coefficients X and Y are 1.25 and 2.5, respectively. The attacker ASes introduce a random asymmetry in the range of 50 ms to 300 ms between their own border router pairs.

Overall Performance. Figure 4 shows cumulative distribution function (CDF) results of local clock offsets at each AS to real-time over 40 days. The majority of ASes achieve highly reliable global time synchronization with only a few microseconds of deviation from real-time; more precisely, even in the extreme case where 20% of entities in the network act maliciously, over 94% of ASes are closely synchronized with real-time. In comparison, 99.3% of ASes are successfully synchronized with real-time when 5% of nodes are malicious.

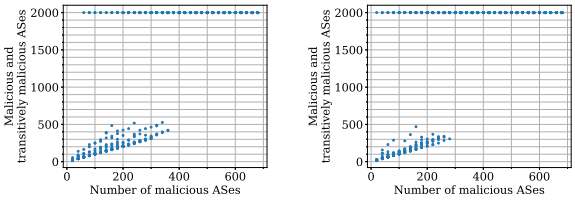


(a) 5% of attacker nodes. (b) 10% of attacker nodes.

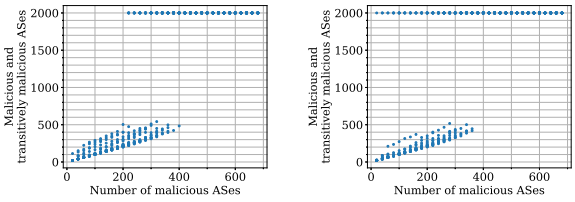


(c) 15% of attacker nodes. (d) 20% of attacker nodes.

Figure 4: CDF of local clock offsets to real-time according to different proportions of attackers in a network of 2000 ASes.



(a) The shortest path (b) One random path



(c) Five shortest paths (d) Five random paths

Figure 5: Number of maliciously affected ASes for different path selection strategies in a topology of 2000 ASes.

Although some ASes exhibit desynchronization by the delay attack, thanks to *maxCorr*, they avoid substantial time-shifts, resulting in a maximum of about $2500\,\mu$s after 40 days. An interesting observation is that multipath does not always guarantee better performance than the shortest path. However, in realistic cases where only small fractions of the network are compromised (e.g., $\leq$ 10%), attackers can desynchronize fewer ASes when a multipath strategy is used.

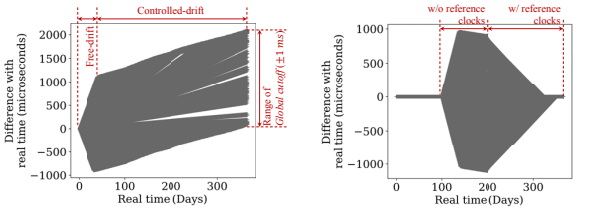
Path Selection Analysis. Another interesting point to highlight is that path selection strategies can be a key factor in overall performance. Each AS uses the single shortest path towards its neighboring ASes, while it applies three different path selection strategies for non-neighboring ASes: shortest path, disjoint path, and random path. In the shortest path selection, each AS evaluates the number of hops of the forwarding paths towards a destination AS provided by the control plane and selects k shortest paths (i.e., $k = 5$ for multipath, otherwise $k = 1$). In the case of disjoint path selection, similar to the shortest path selection, each AS evaluates the set of paths provided by the control plane and selects the shortest paths first. For the second path, however, it selects the most disjoint path with respect to previously selected paths and continues until all k shortest disjoint paths are determined. Finally, the random path selection performs uniform sampling from the set of paths provided by SCION.

Figure 4 demonstrates the benefit of multipath strategies; using multiple short or disjoint paths, fewer ASes are affected by an attacker population of up to 10%. However, the longer tail of the curves for the multipath strategies indicate that *affected ASes* could deviate more from real-time when they use multiple paths. Furthermore, although random path selection strategies (multi- or singlepath) indicate inferior performance compared to other strategies, in all attack scenarios using multiple random paths significantly reduces the number of affected ASes in comparison to using one random path. Despite the decent performance of multipath strategies when the attacker percentage is 5% or 10%, increasing this proportion to 15% and 20% deteriorates their performance relative to the (single) shortest path strategy. That is because, in a network with sparse attacker distribution, the probability of avoiding the attacker nodes through multiple completely different paths is high, whereas this probability decreases in a network with densely distributed attacker nodes.

Worst-Case Behavior of Compromised Nodes. Furthermore, we conduct a reliability analysis only based on the presence of attackers on the paths selected by peers. In order to bound the effect the attacker can have on the system in the worst case, we consider an AS transitively corrupted if the attacker might effectively gain control of its output clock. Once the attacker can arbitrarily pull the clock of an AS, this might in turn affect other ASes, leading to a domino effect.

To reflect this, we repeatedly uniformly sample sets of attacker ASes, ranging from none to one third of the nodes. In each iteration, we determine which ASes might be transitively affected. An AS is affected (i.e., desynchronized) by attackers or transitively corrupted ASes if the fault-tolerant midpoint calculation of the algorithm uses one third or more corrupt input values. Each input is given by measuring the offset to another AS. A measurement is corrupt if the majority of paths used contain a node controlled by the attacker, or if the AS to which the offset is measured is itself (transitively) corrupted. Figure 5 shows the relation between the number of malicious ASes (x-axis) and the number of malicious and transitively corrupted ASes for different path selection policies.

The plots show essentially three regions: below a certain threshold only a few ASes are transitively corrupted, above



(a) No references over a year (b) No references from day 100 to 200

Figure 6: G-SINC results with and without external reference clocks in a network of 500 ASes.

a larger threshold the system globally fails, and between the two thresholds either case might apply. Using five shortest paths, up to 400, i.e., about 20% of primary corrupted nodes can be sustained reliably. Moreover, this figure indicates that using more than one path increases the number of required attackers to affect the whole network.

C. Availability Analysis

We now evaluate the long-term availability of G-SINC with no external reference clocks, simulating an extended GNSS outage caused by, e.g., a solar superstorm [30]; we assume that some underground cables may also be damaged, but thanks to the multipath infrastructure, connectivity between all ASes is assumed to be intact. In this simulation scenario, each local clock solely depends on the global clock synchronization, and no local clock adjustment with the reference clock is available. Our objective is thus to tie all the peers within a synchronization threshold to each other, not to real-time. We set *LocalClock.MaxDrift* to $27\,\mu$s per day and global cutoff $= 1\,$ms. Figure 6a depicts the G-SINC result for a year, and Figure 6b depicts the results for a scenario where reference clocks disappear at some point and come back online again after some time.

From the results we observe the following: (i) For the first month, ASes drift apart following their local drifts (free-drift period). Since each AS has a different local clock drift, their local times are freely distributed (in a radial pattern) within the global cutoff range. (ii) ASes that reached global cutoff do not drift further apart, and as a result, all ASes remain within a range of $\pm\,1\,$ms (controlled-drift period). (iii) The global time inevitably biases because the global clock synchronization process is affected by the delay asymmetry in packet switching, different local clock drifts, and different synchronization initiation times. Finally, ASes are eventually classified into upper-bound group (local drift > global drift), lower bound group (local drift < global drift), and centerline group (local drift = global drift). Nevertheless, G-SINC successfully ties together all the core ASes within $2\,$ms of time drift from real-time for a year of reference clock outage.

As shown in Figure 6b, once the references are available again, the system converges back to UTC within a time span comparable to the outage. The speed of convergence is governed by the restricted corrections that are applied in the clock synchronization algorithm. Hence, the system reconverges more quickly after short outages.

D. Scalability Analysis

For the current design iteration, we assume that the set of core TSes will eventually encompass about 2'000 nodes. A critical question, therefore, is whether the core TSes will be able to (vertically) scale with the expected NTP traffic for the global synchronization as well as the traffic caused by intra-ISD synchronization requests. Computing a Byzantine fault-tolerant approximate agreement on clock corrections based on Lynch and Welch requires a single message exchange from each core TS to all its peers. To make optimal use of the path-aware networking infrastructure, we anticipate conducting offset measurements between any pair of peers in parallel over up to 5 different paths. This amounts to about 10'000 message exchanges that every core TS has to handle in each round. Based on the formal analysis in [17], it will be sufficient to schedule global synchronizations with an interval on the order of hours, e.g., once every hour.

We investigate the scalability question in a simple micro-benchmark between a server machine running our implementation of a SCION-based NTP server process and a load generator machine for NTP client requests via SCION. Both machines were running the operating system Ubuntu Server 20.04 LTS on Amazon's Elastic Compute Cloud instances equipped with 64 (virtual) CPUs (64-bit ARM), 256 GiB of memory, and 25 Gbps of available network bandwidth. Repeatable tests show that our implementation is able to sustain an average load of at least 370'000 NTP requests per second in one process without packet loss. Peak server performance was measured with values of over 400'000 NTP requests per second using the network monitoring tool bwm-ng [20]. These results indicate that our implementation is practical despite the additional SCION packet processing overhead.

VI. Related Work

Extensive studies have been performed on the fundamental problem of clock synchronization in distributed systems to improve accuracy, security, and reliability.

Accuracy. The *Precision Time Protocol (PTP)* uses hardware timestamping to eliminate network stack delays [28]. Thanks to the extensive hardware support at switches, PTP is known to be able to achieve sub-microsecond on a LAN and even sub-nanosecond precision in a well-provisioned datacenter [47]. Nevertheless, to guarantee the high precision, the network needs to be fully PTP-enabled; otherwise, the precision will significantly degrade [76], [36]. Both *Datacenter Time Protocol (DTP)* [36] and HUYGENS [18] can achieve synchronization accuracy of a few 10s of nanoseconds but both systems are targeted only at deployments in datacenters.

Security. Although a large volume of work is carried out on clock synchronization to improve accuracy, its security has only recently gotten attention [42], [61], [29].

Early NTP had no security design aspect. *NTPv3* had first adopted packet authentication with a pre-shared symmetric key, which needs to be established out-of-band [45].

In *NTPv4*, a PKI-based authentication mechanism has been introduced [43]. *IEEE 1588*'s experimental security extension describes an HMAC-based authentication method for PTP, and later several variants (e.g., GMAC or CMAC) were implemented and tested [23], [58]. Unfortunately, they have shown only limited adoption in practice due to the overhead of server-side public key operations. *Authenticated Network Time Synchronization (ANTP)* aims at large-scale deployments [13]. It minimizes server-side cryptographic operations using symmetric cryptography for subsequent synchronization processes while enabling the servers to be stateless. *SecureTime* employs high-performance digital signature schemes to secure multicast time synchronization [3]. The *Secure Time Synchronization (STS)* protocol offloads the authorization to a third party (i.e., Authorization Server), resolving the circular dependency between time synchronization authentication and certificate validation [46].

The *NTP Pool Project* provides more than 4'500 NTP servers (as of April 2022), with which hundreds of millions of systems are able to sync, mitigating individual server failures. Each NTP client gathers clock samples from multiple NTP servers and selects the best clock samples to update the local clock [56]. Perry et al. demonstrate that injecting malicious timeservers into the NTP pool is alarmingly feasible, influencing a significantly large number of systems [60]. *Chronos* applies an approximate agreement algorithm to a large number of NTP servers to guarantee reliable time synchronization even in case many time servers are faulty or under an attacker's control [9].

Reliability. A number of projects and publications with the specific goal of assessing and increasing the robustness of GNSS-based clock synchronization have already been discussed in Section I. The experimental *PTP ring method* outlined in [65] enables time servers with disrupted GNSS clocks to access a redundant source of frequency from clocks located elsewhere in the network. The underlying motivation is related to the contributions of G-SINC despite the fact that direct synchronization via individual PTP links is unable to provide the same degree of (Byzantine) fault tolerance.

VII. Conclusion

It is challenging to design a system with the ambition of going beyond what established time synchronization architectures provide to support critical infrastructure. This is a testimony showing how well for example NTP has been developed over many iterations into an architecture that fulfills its basic requirements to a large degree. Yet G-SINC demonstrates that it is possible to enhance the current state of the art in clock synchronization in particular when it comes to essential qualities like global scalability without a single root of trust, mitigation of network-level attacks, and in general availability even under extreme conditions. This is achieved by building on the solid body of fault-tolerant clock synchronization research and the SCION Internet architecture providing strong resilience and security properties as an intrinsic consequence of its underlying design principles.

VIII. Acknowledgements

We would like to thank the anonymous reviewers for their insightful feedback and valuable suggestions. In particular, we would like to express our gratitude to this paper's shepherd, Alysson Bessani, who helped improve it substantially. Sincere thanks go also to Attila Kinali, David Kleymann, Filip Meier, and Zilin Wang for their generous contributions to the project at various stages of development. We gratefully acknowledge support from ETH Zurich, from the Zurich Information Security and Privacy Center (ZISC), and from the Werner Siemens Stiftung (WSS) Centre for Cyber Trust at ETH Zurich. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 716562). We extend our profound thanks for this contribution.

References

[1] Anapaya Systems, "SCION-Internet: The new way to connect," https://www.anapaya.net/scion-the-new-way-to-connect, 2022.

[2] T. Anderson, K. Birman, R. M. Broberg, M. Caesar, D. Comer, C. Cotton, M. J. Freedman, A. Haeberlen, Z. G. Ives, A. Krishnamurthy, W. H. Lehr, B. T. Loo, D. Mazières, A. Nicolosi, J. M. Smith, I. Stoica, R. van Renesse, M. Walfish, H. Weatherspoon, and C. S. Yoo, "The NEBULA future Internet architecture," in *Future Internet Assembly*, ser. Lecture Notes in Computer Science, vol. 7858. Springer, 2013, pp. 16–26.

[3] R. Annessi, J. Fabini, and T. Zseby, "It's about time: Securing broadcast time synchronization with data origin authentication," in *Proceeings of the International Conference on Computer Communication and Networks (ICCCN)*, 2017.

[4] D. Barrera, L. Chuat, A. Perrig, R. M. Reischuk, and P. Szalachowski, "The SCION Internet architecture," *Communications of the ACM*, vol. 60, no. 6, pp. 56–65, 2017.

[5] I. Castro, A. Panda, B. Raghavan, S. Shenker, and S. Gorinsky, "Route Bazaar: Automatic interdomain contract negotiation," in *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*. USENIX Association, 2015.

[6] Center for Applied Internet Data Analysis, "CAIDA geolocation data," https://www.caida.org/data/as-relationships-geo/, 2022.

[7] L. Chuat, M. Legner, D. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, *The Complete Guide to SCION. From Design Principles to Formal Verification*. Springer International Publishing AG, 2022.

[8] W. de Vries, J. J. Santanna, A. Sperotto, and A. Pras, "How asymmetric is the Internet?" in *Proceedings of the IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2015.

[9] O. Deutsch, N. R. Schiff, D. Dolev, and M. Schapira, "Preventing (network) time travel with Chronos," in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2018.

[10] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM*, vol. 33, no. 3, p. 499–516, 1986.

[11] E. Donley, "Time & frequency activities at NIST," https://www.gps.gov/cgsic/meetings/2020/donley.pdf, pp. 17–24, 2020.

[12] J. J. Douceur, "The sybil attack," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[13] B. Dowling, D. Stebila, and G. Zaverucha, "Authenticated network time synchronization," in *Proceedings of the USENIX Security Symposium*, 2016.

[14] European Union Agency for Cybersecurity, "Power sector dependency on time service: attacks against time sensitive services," https://www.enisa.europa.eu/publications/power-sector-dependency, 2020.

[15] R. Exel, "Mitigation of asymmetric link delays in IEEE 1588 clock synchronization systems," *IEEE Communications Letters*, vol. 18, no. 3, pp. 507–510, 2014.

[16] D. F. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "Network Time Security for the Network Time Protocol," https://www.rfc-editor.org/rfc/rfc8915, 2020.

[17] M. Frei, J. Kwon, S. Tabaeiaghdaei, M. Wyss, C. Lenzen, and A. Perrig, "G-SINC: Global Synchronization Infrastructure for Network Clocks," https://arxiv.org/abs/2207.06116, 2022.

[18] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosunblum, and A. Vahdat, "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2018.

[19] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *Proceedings of the ACM SIGCOMM Conference*, 2009.

[20] V. Gropp, "Bandwidth monitor NG," https://github.com/vgropp/bwm-ng, 2004-2021.

[21] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the Internet," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2005.

[22] H. Hellwig and A. E. Wainwright, "A portable rubidium clock for precision time transport," in *Proc. 7th PTTI Planning Meeting, Washington, D.C.*, 1975, pp. 143–159.

[23] B. Hirschler and A. Treytl, "Validation and verification of IEEE 1588 Annex K," in *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011.

[24] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever, "Blink: Fast connectivity recovery entirely in the data plane," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

[25] B. Hubert, "GPS, Galileo & More: How do they work & what happened during the big outage?" https://berthub.eu/articles/posts/gps-gnss-how-do-they-work/, 2019.

[26] ——, "GNSS jamming and spoofing," https://berthub.eu/articles/posts/galileos-authentication-algorithm-part-3/, 2020.

[27] D. Husmann, L.-G. Bernier, M. Bertrand, D. Calonico, K. Chaloulos, G. Clausen, C. Clivati, J. Faist, E. Heiri, U. Hollenstein, A. Johnson, F. Mauchle, Z. Meir, F. Merkt, A. Mura, G. Scalari, S. Scheidegger, H. Schmutz, M. Sinhal, S. Willitsch, and J. Morel, "Si-traceable frequency dissemination at 1572.06 nm in a stabilized fiber network with ring topology," *Opt. Express*, vol. 29, no. 16, pp. 24 592–24 605, 2021.

[28] IEEE Std 1588-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," 2020.

[29] P. Jeitner, H. Shulman, and M. Waidner, "The impact of DNS insecurity on time," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE Computer Society, 2020.

[30] S. A. Jyothi, "Solar superstorms: Planning for an Internet apocalypse," in *Proceedings of the ACM SIGCOMM Conference*, 2021.

[31] P. G. Kannan, R. Joshi, and M. C. Chan, "Precise time-synchronization in the data-plane using programmable switching ASICs," in *Proceedings of the ACM Symposium on SDN Research*, 2019, pp. 8–20.

[32] C. Krähenbühl, S. Tabaeiaghdaei, C. Gloor, J. Kwon, A. Perrig, D. Hausheer, and D. Roos, "Deployment and scalability of an inter-domain multi-path routing infrastructure," in *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2021.

[33] J. Kwon, J. A. García-Pardo, M. Legner, F. Wirz, M. Frei, D. Hausheer, and A. Perrig, "SCIONLab: A next-generation Internet testbed," in *Proceedings of the IEEE Conference on Network Protocols (ICNP)*, 2020.

[34] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," in *Proceedings of the ACM SIGCOMM Conference*, 2000.

[35] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, p. 52–78, 1985.

[36] K. S. Lee, H. Wang, V. Shrivastav, and H. Weatherspoon, "Globally synchronized time via datacenter networks," in *Proceedings of the ACM SIGCOMM Conference*, 2016.

[37] M. Lévesque and D. Tipper, "Improving the PTP synchronization accuracy under asymmetric delay conditions," in *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, 2015.

[38] M. Lombardi, M. Narins, P. Enge, B. Peterson, S. Lo, Y.-H. Chen, and D. Akos, "The need for a robust, precise time and frequency alternative to GNSS," no. 23, 2012-11-01 2012.

[39] M. A. Lombardi, "Microsecond accuracy at multiple sites: is it possible without GPS?" *IEEE Instrum. Meas. Mag.*, vol. 15, no. 5, pp. 14–21, 2012.

[40] ——, "Multi-source common-view disciplined clock: A fail-safe clock for critical infrastructure systems," *Journal of Research of the National Institute of Standards and Technology*, vol. 126, 2021.

[41] London Economics (LE), "The economic impact on the UK of a disruption to GNSS," https://www.gov.uk/government/publications/the-economic-impact-on-the-uk-of-a-disruption-to-gnss, 2017.

[42] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the network time protocol." in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2016.

[43] J. Martin, J. Burbank, W. Kasch, and P. D. L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," https://www.rfc-editor.org/rfc/rfc5905, 2010.

[44] Meinberg, "Oscillators available for Meinberg GPS receivers / time servers: TCXO, OCXO, Rubidium," https://www.meinbergglobal.com/download/docs/other/oscillator-options-en.pdf, 2017.

[45] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," https://www.rfc-editor.org/rfc/rfc1305, 1992.

[46] F. Mkacher, X. Bestel, and A. Duda, "Secure time synchronization protocol," in *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, 2018.

[47] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over Ethernet," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2009, pp. 1–5.

[48] C. D. Murta, P. R. Torres Jr, and P. Mohapatra, "QRPp1-4: Characterizing quality of time and topology in a time synchronization network," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2006.

[49] NASPI Time Synchronization Task Force, "Time synchronization in the electric power system," https://www.naspi.org/node/608, 2017.

[50] Netnod, "Why accurate and secure time is the key to 21st century energy," https://www.netnod.se/blog/why-accurate-and-secure-time-key-21st-century-energy, 2021.

[51] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "GPS software attacks," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 450–461.

[52] NIST, "Common view GPS time transfer," https://www.nist.gov/pml/time-and-frequency-division/time-services/common-view-gps-time-transfer, 2022.

[53] ——, "Two-way satellite time and frequency transfer (TWSTFT)," https://www.nist.gov/pml/time-and-frequency-division/time-distribution/two-way-satellite-time-and-frequency-transfer, 2022.

[54] A. N. Novick and M. A. Lombardi, "Practical limitations of NTP time transfer," in *2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum*, 2015, pp. 570–574.

[55] nsnam, "ns-3," https://www.nsnam.org/, 2021.

[56] NTP Pool Project, "NTP Pool," https://www.ntppool.org/, 2022.

[57] A. C. O'Connor, M. P. Gallaher, K. Clark-Sutton, D. Lapidus, Z. T. Oliver, T. Scott, M. A. Gonzalez, E. G. Brown, and J. Fletcher, "Economic benefits of the global positioning system (GPS)," 2019.

[58] C. Önal and H. Kirrmann, "Security improvements for IEEE 1588 Annex K: Implementation and comparison of authentication codes," in *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2012.

[59] A. Pathak, H. Pucha, Y. Zhang, Y. C. Hu, and Z. M. Mao, "A measurement study of Internet delay asymmetry," in *Proceedings of the International Conference on Passive and Active Network Measurement*, 2008.

[60] Y. Perry, N. Rozen-Schiff, and M. Schapira, "A devil of a time: How vulnerable is NTP to malicious timeservers?" in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2021.

[61] D. Reilly, H. Stenn, and D. Sibold, "Network Time Protocol Best Current Practices," https://www.rfc-editor.org/rfc/rfc8633, 2019.

[62] B. Rothenberger, D. Roos, M. Legner, and A. Perrig, "PISKES: Pragmatic Internet-Scale Key-Establishment System," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2020.

[63] N. R. Schiff, D. Dolev, T. Mizrahi, and M. Schapira, "A secure selection and filtering mechanism for the Network Time Protocol with Chronos," https://datatracker.ietf.org/doc/html/draft-ietf-ntp-chronos-05, 2022.
[64] SCION Project, "SCION open-source implementation," https://github.com/scionproto/scion, 2022.
[65] J. A. Sherman, L. Arissian, R. C. Brown, M. J. Deutch, E. A. Donley, V. Gerginov, J. Levine, G. K. Nelson, A. N. Novick, B. R. Patla, T. E. Parker, B. K. Stuhl, D. D. Sutton, J. Yao, W. C. Yates, V. Zhang, and M. A. Lombardi, "A resilient architecture for the realization and distribution of Coordinated Universal Time to critical infrastructure systems in the United States: Methodologies and recommendations from the National Institute of Standards and Technology (NIST)," https://doi.org/10.6028/nist.tn.2187, 2021.
[66] Sunrise, "SCION – Secure Connectivity At Multiple Locations," https://www.sunrise.ch/business/en/enterprise/internet-networking/business-wan/scion, 2022.
[67] Swisscom, "The secure, high-speed Internet under your control," https://www.swisscom.ch/scion, 2022.
[68] SWITCH, "SWITCHlan SCION Access," https://www.switch.ch/scion/, 2022.
[69] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 75–86.
[70] F. Tirado-Andrés and A. Araujo, "Performance of clock sources and their influence on time synchronization in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 9, 2019.
[71] B. Trammell, J.-P. Smith, and A. Perrig, "Adding path awareness to the Internet architecture," *IEEE Internet Computing*, vol. 22, no. 2, pp. 96–102, 2018.
[72] UK Government Office for Science, "Satellite-derived time and position: a study of critical dependencies," https://www.gov.uk/government/publications/satellite-derived-time-and-position-blackett-review, 2018.
[73] US Executive Office of the President, "Strengthening national resilience through responsible use of positioning, navigation, and timing services," https://www.federalregister.gov/d/2020-03337, 2020.
[74] J. L. Welch and N. A. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," *Information and Computation*, vol. 77, no. 1, pp. 1–36, 1988.
[75] X. Yang, D. Clark, and A. W. Berger, "NIRA: A new inter-domain routing architecture," *IEEE/ACM Transactions on Networking*, 2007.
[76] R. Zarick, M. Hagen, and R. Bartoš, "Transparent clocks vs. enterprise Ethernet switches," in *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011.