Exploiting Electrical Grid for Accurate and Secure Clock Synchronization

SREEJAYA VISWANATHAN, Illinois at Singapore Pte Ltd
RUI TAN, Nanyang Technological University
DAVID K. Y. YAU, Singapore University of Technology and Design

Desynchronized clocks among network nodes in critical infrastructures can degrade system performance and even lead to safety incidents. Clock synchronization protocols based on network message exchanges, though widely used in current network systems, are susceptible to delay attacks against the packet transmission. This vulnerability cannot be solved by conventional security measures, such as encryption, and remains an open problem. This article proposes to use the sine voltage waveform of a utility power grid to synchronize network nodes connected to the same grid. Our experiments demonstrate that minute fluctuations of the voltage's cycle length encode fine-grained global time information in Singapore's utility grid. Based on this key result, we develop a clock synchronization approach that achieves good accuracy and is provably secure against packet-delay attacks. Implementation results show that our approach achieves an average synchronization error of 0.1 ms between two network nodes that are deployed in office and residential buildings 10 km apart. When the proposed system is deployed within the same floor of an office building, the error reduces to 10 μs. When there are heavy industrial loads close to one of the two nodes 10 km apart, the system can still maintain subsecond accuracy. Moreover, when the two nodes are deployed within the same building floor with industrial loads nearby, the average synchronization error is 34 μs.

CCS Concepts: • **Computer systems organization** → **Sensor networks**; *Dependable and fault-tolerant systems and networks*;

Additional Key Words and Phrases: Clock synchronization, security, critical infrastructures, cyber-physical systems

ACM Reference format:
Sreejaya Viswanathan, Rui Tan, and David K. Y. Yau. 2018. Exploiting Electrical Grid for Accurate and Secure Clock Synchronization. *ACM Trans. Sen. Netw.* 14, 2, Article 12 (May 2018), 32 pages.
http://doi.org/10.1145/3195182

A preliminary version of this work appeared in the 37th IEEE Real-Time Systems Symposium (RTSS'16).
This research is supported in part by the National Research Foundation, Prime Minister's Office, Singapore under the Energy Programme and administrated by the Energy Market Authority (EP Award No. NRF2014EWT-EIRP002-026) an NTU Start-up Grant, and MOE grant number SUTDT12017004.
Authors' addresses: S. Viswanathan, Advanced Digital Sciences Center, Illinois at Singapore Pte Ltd, 1 Fusionopolis Way, #08-10 Connexis North Tower, Singapore 138632; email: sreejaya.v@adsc.com.sg; R. Tan (corresponding author), School of Computer Science and Engineering, Nanyang Technological University, N4-02C-85, 50 Nanyang Avenue, Singapore 639798; email: tanrui@ntu.edu.sg; D. K. Y. Yau, Information Systems Technology and Design, Singapore University of Technology and Design, 8 Somapah Road, Building 1, Level 5, Singapore 487372; email: david_yau@sutd.edu.sg.

http://doi.org/10.1145/3195182

1 INTRODUCTION

Critical infrastructures—for example, utility grids, manufacturing systems, and public transportation—are embracing the vision of the Internet of Things (IoT) [30], which provides a fabric that connects advanced sensing, computing, communications, and actuation [26]. For the "things" (i.e., network nodes) in a critical infrastructure, trustworthy time information can be critical. Accurate timestamps of data allow us to make sense of the data relative to extrinsic events. Tight clock synchronization enables punctual and coordinated real-time operations. Desynchronized clocks, on the other hand, can degrade system performance or cause expensive infrastructure damage. For instance, in an electrical grid, smart meters and other intelligent electronic devices (IEDs) installed at substations to monitor the grid's state and operate power instruments accordingly often require global clock synchronization of submillisecond accuracy. Stale measurements will result in erroneous control that endangers the grid's safety [25]. In car manufacturing, for example, desynchronized robots in a Roboteam [29] working on a same car can cause clashes of their arms and disrupt the production pipeline.

In existing critical infrastructures, Network Time Protocol (NTP) and Precision Time Protocol (PTP) are often used to synchronize distributed slave nodes to a master node, which may be equipped with a Global Positioning System (GPS) receiver for further global synchronization. However, as discussed in RFC 7384 [20], these protocols are susceptible to various cybersecurity threats. A simple packet-delay attack, in particular, is effective in desynchronizing the slave nodes. This attack cannot be prevented by conventional security measures, including cryptographic authentication and encryption [19, 28]. In the attack, a malicious intermediate node on the network path between the slave and master strategically delays the transmissions of the NTP or PTP packets in order to manipulate a slave's clock. For instance, if an NTP request or reply is delayed maliciously by τ, the slave's clock will have an extra drift of $\tau/2$ from the master's clock [19, 28]. To the best of our knowledge, there is no solution to completely mitigate this attack. The effectiveness of existing mitigation approaches [20], such as using redundant masters and network paths, varies significantly depending on the network topology and attack points. In particular, a dense deployment of GPS receivers in a critical infrastructure may harden its cyber network, but it may increase the physical attack surface because the GPS receivers are susceptible to wireless spoofing that can be launched remotely (e.g., 1.4 km away [2]) using low-cost (e.g., $300 [17]) hardware.

Critical infrastructures have also resorted to cyber isolation to protect their networks. Such isolation is shaky, however [12]. Zero-day vulnerability exploits, insider attacks, and stepping stone attacks can render the isolation futile, as evidenced in recent high-profile intrusions, including Dragonfly [10] and Stuxnet [11] against power and nuclear plants. For the problem context of this article, insiders (e.g., disgruntled employees) who have access to network management may easily launch a packet-delay attack at a strategic router, thereby endangering the integrity of time among a large number of network nodes. Increased network connectivity due to adoption of the IoT will only further lower the barriers of penetrating critical infrastructures. Hence, providing secure clock synchronization in these systems, where connectivity is a defining characteristic, is an imperative research problem.

Recent research efforts have investigated clock synchronization in wireless sensor networks and mobile/pervasive computing. These approaches leverage the nodes' built-in radios [4, 5, 18] or external wireless time broadcasts or periodic signals found in AM/FM radios [3, 14], Wi-Fi beacons [9], power line electromagnetic radiation (EMR) [23], and fluorescent light flickering [16]. However, they are designed without security considerations. Moreover, reliance on wireless electromagnetic signals in a mission-/safety-critical context often raises reliability and security concerns owing to the possibility of wireless jamming and spoofing.

In this article, we advance the notion of *inherent security* in providing secure and accurate time in time-critical systems. We exploit the electric network voltage (ENV) signal of an alternating current (ac) utility grid to design an accurate clock synchronization approach with provable security against the packet-delay attack for a network system connected to the same grid. The following properties of the ENV make it an ideal extrinsic signal that serves our purposes. First, the ENV is a periodic signal with a nominal frequency of 50 or 60 Hz, and it is almost identical across all locations within a local area (e.g., a power substation or factory). Our measurements show that the phase difference between the ENVs at two locations 10 km apart is generally below 0.2 ms and it has a mean value of around 0.1 ms. This property enables us to achieve a submilliseconds average error in clock synchronization. Second, in critical infrastructures, many network nodes are connected to the utility grid for stable power supply and unattended long-term operations. This typical setup renders our ENV-based approach widely applicable. Third, the ENV is a highly available and almost unforgeable physical signal that is practically difficult for the attacker to tamper with or jam. Any high-frequency noise injected by the attacker into related power lines can be removed readily by a low-pass filter. On the other hand, injection of low-frequency disturbances that can distort the ENV waveform would require a considerable amount of energy and physical tampering to the power networks. For instance, our experiments in this article show that heavy industrial loads can affect the ENV waveform. However, the implementation of these physical attacks raises insurmountable barriers economically and logistically for would-be attackers. Moreover, if the attackers have obtained physical access to the power network supporting the targeted system, simply causing power outages by connecting large loads exceeding the network's power rating or introducing a short circuit would be more attractive than attacking the clock synchronization. Thus, although these physical attacks cannot be totally ruled out, we focus on the packet-delay attack in this article.

Subject to the gridwide ENV, different network nodes can count the ac cycles of the ENV signal to achieve *clock calibration* [14, 16]. Note that clock calibration ensures that different clocks advance at the same speed, whereas clock synchronization regulates the clocks to have the same value. Thus, continuous clock calibration keeps the clocks synchronized once they are initially synchronized. The initial synchronization, however, requires the exchange of network messages, which may be subverted by packet delay attacks. A major contribution of this article is the identification, validation, and exploitation of a physical fingerprint embedded in the ENV signal, which we call a *time fingerprint* (TiF), which provides resilience against delay attacks. A TiF is a vector of successive ac cycle lengths of the ENV signal. In a power grid, although the system frequency is regulated at 50 or 60 Hz by a control system [13], the frequency fluctuates continuously because of inevitable transient imbalance between generation and load. Accordingly, the ac cycle length fluctuates around its nominal value as well. Our extensive measurements show that, using a similarity-based matching algorithm, a TiF of sufficient length captured by a network node—say, *A*, can be correctly *time aligned* within a trace of ac cycle lengths captured by another network node—say, *B*—at the granularity of an ac cycle.

These key observations enable a novel cyber-physical approach to achieving the objectives of accurate and secure clock synchronization simultaneously for a network system served by the same utility grid. Specifically, in a synchronization session, we ensure the integrity (e.g., using cryptographic signature) of a packet from node *A* to node *B* that contains a TiF captured by *A* and the corresponding *A*'s clock value. Based on that, *B* will be able to time align the received TiF within its own historical ac cycle lengths timestamped with its clock. As a result, *B* will be able to compute the offset between *A*'s and *B*'s clocks. If this offset is communicated back to *A* with guaranteed integrity, *A* can then calibrate its clock to synchronize with that of *B*. This new approach is immune to any malicious delays introduced in the communications between *A* and *B*

since it does not depend on any explicit measurements of the network transmission delays. This principle, which applies for a pair of nodes, underlines a complete clock synchronization system among all the grid-connected nodes.

This article presents a prototype implementation of our system and discusses its performance based on extensive empirical evaluations. While our approach is immune to the packet-delay attack, owing to random noises in TiFs, the TiF matching may produce errors, albeit with very low probabilities. This article develops an error-correction algorithm by cross-checking the synchronization results among multiple nodes. Our analysis shows that a four-node system can correct one erroneous synchronization session out of a total of six synchronization sessions during a cross-check round. This significantly improves the robustness of our approach against rare synchronization errors owing to noises in TiFs. Extensive evaluation shows that our approach achieves an average synchronization error of 0.1 ms between two network nodes that are deployed in office and residential buildings 10 km apart. When the proposed system is deployed within the same floor of an office building, the error reduces to 10 μs. We also evaluate the impact of industrial loads on the synchronization accuracy of our approach. The evaluation shows that, if the two network nodes are affected by the same industrial loads, the average synchronization error is 34 μs. When a node is affected by heavy industrial loads and the other is not, our synchronization approach still maintains subsecond accuracy.

The remainder of the article is organized as follows. Section 2 reviews related work. Section 3 presents the design and implementation of the TiF capture hardware. Section 4 presents extensive measurements that characterize key properties of the TiF under different deployment environments. Section 5 discusses the design and implementation of the proposed secure clock synchronization approach. Section 6 presents the synchronization error-correction algorithm and its performance analysis. Section 7 analyses evaluation results of the system prototype. Section 8 discusses several issues that are not addressed in this article. Section 9 contains our conclusions.

2 RELATED WORK

Various clock calibration and synchronization approaches have been proposed for wireless sensor networks and mobile/pervasive computing applications. They can be classified broadly into two categories. The first category (e.g., RBS [4], TPSN [5], and FTSP [18]) achieves clock synchronization by exchanging radio messages among the nodes in question. The second category [3, 9, 14, 16, 23] exploits external wireless time broadcasts and periodic signals. Chen et al. [3] have designed a low-power mote peripheral that can decode time broadcasts from timekeeping radio stations (WWVB and DCF77) to achieve global time synchronization. Li et al. [14] exploit the Radio Data System (RDS) of FM radios, which broadcasts data blocks periodically, to calibrate the clocks of motes. Similarly, ZigBee nodes have used detection of periodic Wi-Fi beacons for clock calibration [9]. Rowe et al. [23] have designed a mote peripheral to receive periodic EMR from utility power lines and calibrate the clocks of motes based on the detected ac cycles. Li et al. [16] leverage periodic fluorescent light flickering to calibrate the clocks of nodes equipped with light sensors. Our approach belongs to the second category, but it is also fundamentally different from all the prior work. They leverage the periodicity of the external signals to achieve clock calibration, whereas we exploit gridwide *imperfections* of the ENV's periodicity to achieve clock synchronization. They do not address security, whereas we address it as a principal concern. Their use of wireless signals often raises reliability and security concerns for mission- and safety-critical systems. We do not use wireless signals.

Moreover, the approaches based on powerline EMR [23], FM RDS [14], Wi-Fi beacons [9], and light flickering [16] achieve clock calibration by counting cycles. Note that *clock calibration* is related to, but different from, *clock synchronization*, as discussed in Section 1. Clock calibration

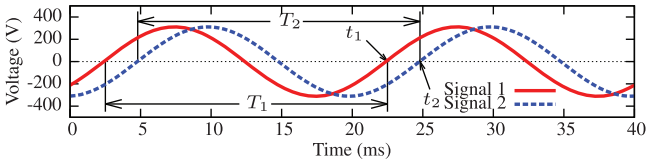


Fig. 1. Illustration of ENV ac cycle length. T_1 and T_2 are two ac cycle length measurements.

ensures that different clocks advance at the same speed, whereas clock synchronization regulates the clocks to have the same value. Continuous clock calibration keeps the clocks synchronized once they are initially synchronized. However, these existing studies [9, 14, 16, 23] do not devise new approaches for the initial synchronization or any resynchronization needed should hardware/software faults occur. They use the conventional clock synchronization approaches based on network message exchanges, which may be subverted by packet-delay attacks. Thus, our synchronization approach is complementary to existing work in clock calibration [9, 14, 16, 23] in that we provide the secure initial synchronization and resynchronization needed to bootstrap their systems.

ENV has been exploited to tell time for decades. Some electric clocks connected to utility grids (e.g., those found in home appliances) advance by counting the ac cycles. In some power grids, the grid operators regulate the *grid time*, that is, the product of the number of ac cycles and the nominal cycle length (e.g., 20 ms for a 50 Hz grid) based on Coordinated Universal Time (UTC) by correcting the grid frequency. However, the regulation often has errors on the order of seconds. For instance, by controlling generators, the grid operator in Texas increases/decreases the grid frequency whenever the error of grid time exceeds 2 s [21], thus keeping the maximum error to be also about 2 s. Moreover, because this regulation may negatively impact power grid reliability [22], it is either not adopted or considered obsolete and being phased out. Grid time is therefore unsuitable for accurate synchronization of network nodes to UTC.

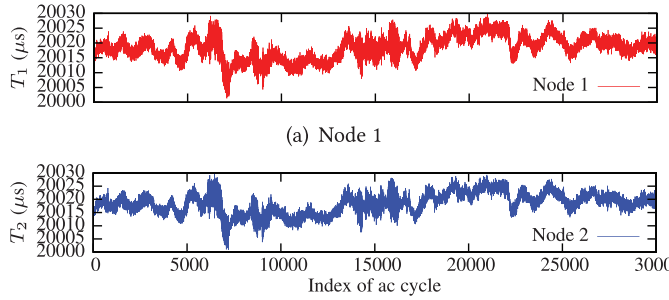
Audio and video recorders can capture the grid's frequency based on electromagnetic interference from power lines or visual interference under fluorescent lighting [6, 7, 24]. The continuously fluctuating grid frequency over time may generate a signature for multimedia forensics. For example, it is possible to authenticate the recording time of an audio/video clip by matching a frequency trace extracted from the clip against a historical grid frequency database recorded directly from the utility grid. Since these forensic approaches sample the grid frequency every few seconds, the identification similarly has a temporal granularity on the order of seconds. In this article, we solve the systems challenge of capturing the fine-grained, ENV-based TiF and validate its ability to "encode" time information with submillisecond accuracy. We further apply the TiF to design a novel clock synchronization system that satisfies both the security and accuracy requirements for critical infrastructures.

3 CAPTURING POWER GRID TIME FINGERPRINT

In this section, we use real data traces to illustrate the fluctuations of ENV cycle lengths at different locations of a utility grid, which motivate the concept of TiF. We then describe our hardware design to capture the fluctuations in high resolution.

3.1 ENV Cycle Length Fluctuations

Figure 1 illustrates two ENV signals with a phase shift. We can observe these phase shifts when the signals are measured at different locations in a power network owing to characteristics of the electrical power lines [13]. An ac cycle length of the red solid ENV signal is the time period T_1 between



(a) Node 1

(b) Node 2

Fig. 2. ENV ac cycle lengths measured by two nodes at different locations in a building for a duration of 10 minutes.

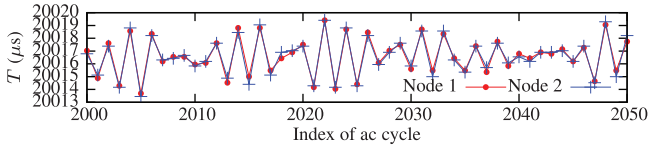


Fig. 3. A zoomed-in view of Figure 2.

two consecutive zero crossings, illustrated in Figure 1. In this article, we design a hardware device to continuously measure the ac cycle lengths. Details of the design will be presented in Section 3.2. We deploy two of the hardware devices, Node 1 and Node 2, in two different rooms on the same floor of an office building. Figure 2 shows 30,000 ac cycle length measurements obtained by the two nodes, respectively, over the same time period of about 10 minutes. The ac cycle lengths shown are around 20,015 μs, because the nominal grid frequency in our region is 50 Hz. The ac cycle length changes over time and the fluctuations at the two nodes are almost identical. To illustrate, Figure 3 shows a zoomed-in view of Figure 2, where the traces measured by both nodes are depicted over a selected window of 1 s. We can see that the ac cycle lengths measured by the two nodes fluctuate synchronously. The fluctuations are within 10 μs, which is just 0.05% of the nominal cycle length.

The good match between the profiles of the fluctuating ac cycle lengths, as shown in Figure 3, suggests that (i) the fluctuations at different locations on a building floor are nearly identical, and (ii) a trace of the fluctuations over a certain time period is unique over a longer time horizon. These two hypotheses, if true, imply that a trace of the fluctuations may naturally "encode" a unique signature for when the trace was captured. We thus call a vector of some number of consecutive ac cycle lengths (recorded at some location) a *time fingerprint* (TiF). In Section 4, we present extensive measurements in a wide range of settings (e.g., length of the TiF and physical distance between two synchronizing nodes distributed in Singapore) to test these two hypotheses.

3.2 Time Fingerprint Capture Hardware

This section presents a hardware design for capturing minute fluctuations of the ENV cycle length, as shown in Figure 3. The design must allow high-resolution measurements to preserve even tiny features. Moreover, it should be designed as a portable periphery that can be easily integrated with commercial off-the-shelf (COTS) IoT platforms.

3.2.1 Hardware and Firmware. A possible method is to directly sample the ENV signal using a high-speed analog-to-digital converter (ADC) and compute the cycle lengths from the captured data. However, processing high-rate data will incur significant compute overhead, which may

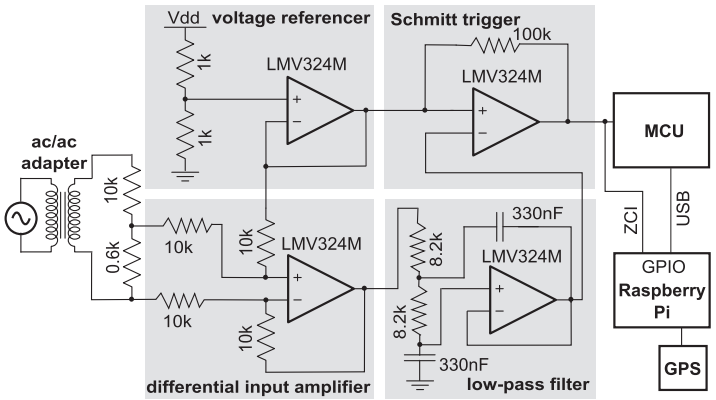


Fig. 4. Schematics of the proposed hardware prototype. GPS and Raspberry Pi are used for evaluation purposes only and can be eliminated or replaced in actual deployments.

threaten the system's real-time performance. High-speed ADC is expensive as well. This sampling method is therefore not advisable. Instead, we design a circuit to generate interrupts upon zero crossings of the ENV. These crossings are then analyzed by a microcontroller (MCU) to give the ac cycle lengths. Figure 4 shows the schematics of our prototype hardware. Given the line-to-neutral utility voltage, the prototype uses a certified off-the-shelf ac/ac adapter and a voltage divider to step down the voltage. Thus, the prototype has the electrical protection features provided by the off-the-shelf ac/ac adapter. The voltage signal is conditioned and converted to a square wave signal that preserves the cycle lengths and generates interrupts to an MCU. A firmware on the MCU then uses an internal high-frequency timer to measure the cycle lengths locally and efficiently.

Details of the signal processing components in Figure 4 are as follows. The combination of the ac/ac adapter and the voltage divider outputs a differential sinusoid signal with a peak-peak voltage of around 2 V. A unit-gain *differential input amplifier* converts the differential signal to a single-ended signal and adds to it a reference voltage of 1.65 V provided by a *voltage referencer*. The resulting output is thus a voltage signal referenced by ground and centered at 1.65 V. The measurement of ac cycle lengths can be affected by localized high-frequency (e.g., tens to hundreds of kHz) voltage noise emitted by electrical appliances and consumer electronics in the environment [8]. To reduce their impact, we apply a Sallen-Key second-order, low-pass filter to the single-ended signal. The cut-off frequency of the filter is 58.8 Hz. The filter can also remove malicious high-frequency noise injected into related power lines by an attacker. The filtered signal is passed to a Schmitt trigger that compares the signal with the reference voltage. When the filtered signal goes above the reference voltage, the output swings at the positive rail; otherwise, it swings at the negative rail. Thus, the time duration between two consecutive rising edges of the square wave output of the Schmitt trigger gives an ac cycle length. In our design, the differential input amplifier and the Schmitt trigger use the same reference voltage from the voltage referencer for adding and triggering, respectively. Thus, noise in the reference voltage will not affect the measurement.

Our prototype uses a development board equipped with an STM32F407VGT6 32-bit MCU to measure the ac cycle lengths. The firmware running on the MCU is written in C. Specifically, we configure a hardware timer running at 8.4 MHz. On receiving an ENV zero-crossing interrupt (ZCI) from the Schmitt trigger (i.e., a rising edge of the Schmitt trigger's output), the MCU outputs an unsigned integer that is the difference between its present timer value and the timer value at the last interrupt. By excluding a time quantity that corresponds to the nominal ac cycle length

(20 ms), the MCU's ac cycle length measurement can be represented in 2 B . The time resolution is $1/8.4\,\text{MHz} \approx 0.12\,\mu\text{s}$, which is sufficient for capturing the fluctuations with a magnitude of $10\,\mu\text{s}$, as shown in Figure 3. The MCU delivers each measurement immediately to its host IoT device with the corresponding ZCI. By handling the ZCI, the IoT device can accurately timestamp the present cycle length measurement using its local clock despite communication delay from the MCU to the IoT device.

3.2.2 Integration with IoT Platforms. In Figure 4, the components in the shaded areas make up the core of the TiF capture hardware. The MCU can be a native unit in the capture hardware or it can be one on the IoT platform to which the capture hardware is attached. For instance, we can leverage the MSP430 MCU, widely found in sensor network platforms and power grid devices [27], to measure the cycle lengths. The TimerA provided by TinyOS on an MSP430-based kMote used in a smart plug platform can achieve a 4.2 MHz clock rate after a reconfiguration of sourcing the SMCLK clock from the DCO clock without a divider. The achieved $0.24\,\mu\text{s}$ time resolution is sufficient for capturing the ac cycle length fluctuations. Since the TiF capture hardware (i.e., the shaded components in Figure 4) needs to access the ENV, we can integrate it into the power supply unit of IoT devices.

In our setup, we use a Raspberry Pi (RPi) single-board computer as an example IoT platform. This is because the RPi supports diverse peripheral and networking interfaces that facilitate the evaluation. The RPi receives the ac cycle length measurements from the MCU through a virtual COM port over a USB cable. The ZCI signal is connected to a general-purpose input/output (GPIO) pin of the RPi. A GPIO interrupt handler records the RPi's system clock at microsecond resolution to obtain a timestamp that it adds to the incoming cycle length measurement from the USB. To obtain accurate ground-truth time for the evaluation, we integrate the RPi with an Adafruit GPS receiver [1] that delivers both NMEA sentences and pulse-per-second (PPS) signals with 10-ns accuracy through the GPIO pins to the RPi. By using Raspbian OS's `gps-gpio` kernel module and a few other software packages, including `gpsd`, the RPi's clock can be synchronized to the UTC with an offset of $1\,\mu\text{s}$ or less. Note that in actual deployments of the proposed TiF-based clock synchronization, the GPS receiver will not be needed and other IoT hardware platforms can replace the RPi.

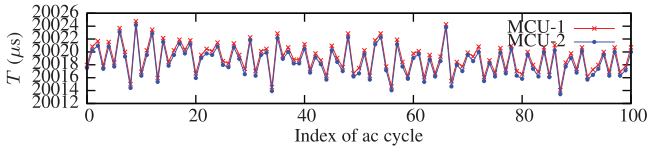
4 MEASUREMENT STUDY

This section presents extensive measurements using the hardware prototype in Section 3.2 to understand key properties of the ac cycle length fluctuations. These properties form an important basis for designing the TiF-based clock synchronization platform in Section 5 with appropriate choice of system parameters.

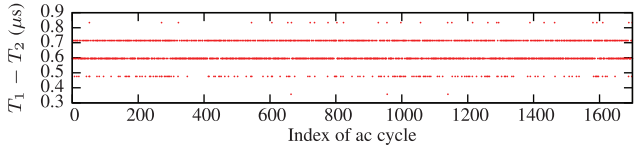
4.1 Hardware Prototype Tests

As the ac cycle length fluctuations of interest are tiny, it is important to ensure that the hardware has sufficient precision of measurement. This section validates the achieved precision of the hardware design.

4.1.1 MCU's Time Precision. We test the precision of the MCU in measuring ac cycle lengths. We connect the output signal of the Schmitt trigger to two MCUs. As the timer runs at 8.4 MHz, we divide each output of the MCUs by 8.4 to yield an ac cycle length measurement in microseconds. Figure 5(a) shows the ac cycle lengths measured by the two MCUs. Their measurements fluctuate over time but are almost identical. Figure 5(b) shows the difference between their corresponding measurements as a function of time. The differences are discrete and in multiples of $0.12\,\mu\text{s}$. This is because the timer output is an integer and a timer tick corresponds to $0.12\,\mu\text{s}$. Ideally, since the two

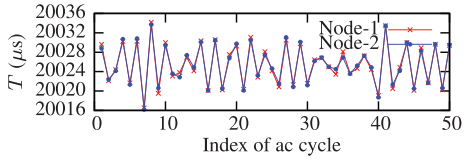


(a) Ac cycle length measured by two MCUs.

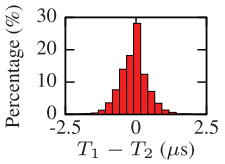


(b) Measurement difference between the two MCUs.

Fig. 5. Ac cycle lengths measured by two MCUs connected to the same Schmitt trigger of a node.



(a) Ac cycle lengths measured by two nodes vs. index of ac cycle

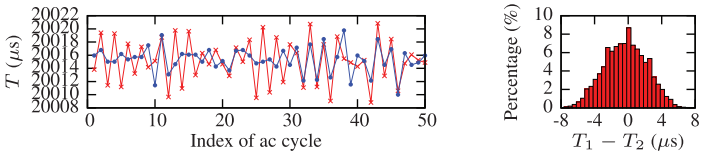


(b) Empirical distribution of the two nodes' ac cycle length differences

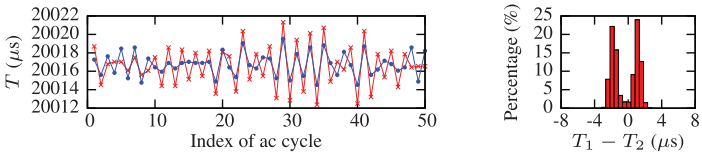
Fig. 6. Ac cycle lengths measured by two nodes connected to the same power extension cord and the empirical distribution of their differences. The empirical distribution has 10,000 data points. The mean and standard deviation (s.d.) of the distribution are $-0.008\,\mu$s and $0.571\,\mu$s, respectively.

MCUs receive the same signal from the Schmitt trigger, their measured ac cycle lengths are also exactly the same. The small differences in practice, as illustrated in Figure 5(b), may be caused by jitters of the hardware timers and the MCUs' respective random delays in handling the interrupts. However, these submicrosecond differences will not affect the ac cycle measurements much since these measurements change by more than $10\,\mu$s, as shown in Figure 5(a). This set of measurements confirms that the MCUs can achieve the required high time resolution.

4.1.2 Prototype Precision. To assess the overall precision of the hardware prototype, we connect two separate hardware nodes to the same power extension cord to measure cycle lengths of the same ENV signal. Figure 6 shows the measurements by the two nodes and the distribution of the differences between their corresponding measurements. The mean and standard deviation (s.d.) of the differences are $-0.008\,\mu$s and $0.571\,\mu$s, respectively. The small mean suggests a tiny difference between the measurement biases of the two nodes and the low s.d. suggests a high precision of measurement. Note that, as discussed in Section 4.2.1, the TiF decoding is not sensitive to small measurement bias differences. For instance, a test in Section 4.2.1 shows that the TiF decoding result is not affected if the two nodes have an additional bias difference no more than $6.5\,\mu$s, which is 800 times the achieved $0.008\,\mu$s. Thus, we choose not to calibrate the nodes to remove the bias difference using a reference node or other accurate instruments. This also suggests the effectiveness of our design to avoid labor-intensive per-node calibration for large-scale network deployments.



(a) No low-pass filter. The mean and s.d. of the empirical distribution are $-0.167\ \mu$s and $2.546\ \mu$s, respectively.



(b) With low-pass filter. The mean and s.d. of the empirical distribution are $-0.167\ \mu$s and $1.473\ \mu$s, respectively.

Fig. 7. Ac cycle lengths measured by two nodes in two different rooms, and empirical distribution of the differences of their corresponding measurements. The distribution has 50,000 data points.

4.1.3 Effectiveness of Low-Pass Filter. To understand the effectiveness of the low-pass filter, we deploy two nodes in two different rooms on the same floor of a building. Figure 7 shows the measurements by the two nodes with and without low-pass filters, respectively. The figure also shows the distribution of the differences between the two nodes' corresponding measurements. Because the nodes experience different localized high-frequency noises, without low-pass filters, their measurements exhibit differences having a larger s.d. ($2.546\ \mu$s) than the measurements in Figure 6. The low-pass filters reduce the s.d. to $1.473\ \mu$s, as shown in Figure 7(b), which suggests that the filter is effective in mitigating the impact of localized high-frequency noise. The two-lobe shape of the error distribution in Figure 7(b) may be caused by different efficiency of the filters in removing the noises seen by the two nodes. A more basic cause might be hardware biases between the low-pass filters of the two tested nodes. We note that it is possible to design a more complicated filter that can configure itself automatically to yield a concentrated error distribution closer to Gaussian during the per-node calibration process. As discussed earlier, such a tedious process is not desirable for large-scale network deployments. The results presented in this article are based on our prototype nodes without per-node calibration.

4.2 Time Fingerprint Decoding

The good match of the ac cycle length fluctuations observed in Section 3.1 implies that a TiF captured by a node — say, A — can be time aligned within a trace of ac cycle lengths captured by another node —say, B. In other words, B can "decode" the time according to B's local clock, during which the TiF was captured by A, provided that the measurements in the trace were timestamped using B's clock. In this section, we evaluate extensively the accuracy of the decoded time under different settings.

4.2.1 Decoding Algorithm and Evaluation Methodology. A TiF, denoted by $\mathbf{x}$, is a vector of n consecutive ac cycle lengths measured by node A. Let a vector $\mathbf{y}$ denote a trace of m consecutive ac cycle lengths measured by node B. The measurements in $\mathbf{y}$ are timestamped with B's clock, whereas only the last element of $\mathbf{x}$ is timestamped with A's clock. We assume that the time duration of measuring $\mathbf{x}$ is within the time duration of measuring $\mathbf{y}$, which is denoted as $\mathbf{x} \lhd \mathbf{y}$. How to ensure this condition is discussed in Section 5.3.

Decoding $\mathbf{x}$ means identifying the time instant, according to B's clock, for the last element of $\mathbf{x}$ (i.e., $\mathbf{x}[n]$). Why we choose the last element of $\mathbf{x}$ will be discussed in Section 5.3. The basic idea of the decoding is to match $\mathbf{x}$ with a TiF within a window of size n in $\mathbf{y}$ using a similarity metric, for example, the reciprocal of sum of square errors (RSSE). By sliding the window within $\mathbf{y}$, the timestamp of the last element of the window that yields the largest similarity is identified as the time instant for $\mathbf{x}[n]$. Formally, the index of the window that yields the largest similarity is given by

$$i^* = \underset{i \in [1, m-n+1]}{\arg\max} \; s(\mathbf{x}, \mathbf{y}[i : i+n-1]), \tag{1}$$

where $s(\cdot, \cdot)$ is the similarity function and $\mathbf{y}[i : i+n-1]$ represents a vector consisting of the ith to $(i+n-1)$th elements of $\mathbf{y}$. Then, the decoding algorithm outputs the timestamp of the last element of the window (i.e., $\mathbf{y}[i^*+n-1]$) for $\mathbf{x}[n]$.

Note that the TiF capture devices may have measurement biases. By using the RSSE as the similarity metric, a small difference between the measurement biases in $\mathbf{x}$ and $\mathbf{y}$ will most likely not affect the result of the optimization in Equation (1). This is because their bias difference will decrease the RSSEs of the sliding windows in $\mathbf{y}$ by similar amounts. We also evaluate the sensitivity of the decoding result to the bias difference using the data traces collected in the prototype precision test in Section 4.1. We set $n = 400$ and $m = 1000$. We add Gaussian noises of s.d. $0.6\,\mu$s to $\mathbf{y}$, where the setting of the s.d. is based on the measured s.d. in Section 4.1. We increase the mean of the noise generator until the decoding result based on the noisy $\mathbf{y}$ is different from that based on the original $\mathbf{y}$ trace. The largest additional bias difference that the decoding can tolerate is $6.5\,\mu$s. If we add noises to $\mathbf{x}$, the result is $8.5\,\mu$s. Such bias difference bounds of several microseconds are much larger than the achieved $0.008\,\mu$s bias difference in Figure 6.

We evaluate the accuracy of the decoding algorithm as follows. We deploy the hardware prototype at two nodes at different locations. By leveraging ground-truth timestamps from the integrated GPS receivers, we select two traces of ac cycle length measurements of length m that are captured by the two nodes during the same time period, respectively. Within the trace captured by A, we slide a window of size n to generate a total of $(m-n+1)$ TiFs. We use the algorithm in Equation (1) to decode each TiF. Let i_k^* denote the output of Equation (1) for the kth TiF from A's trace. Since the two selected traces are captured during the same time period, $i_k^* = k$ signifies a *correct decoding*. We thus measure the *probability of correct decoding* as the ratio of the number of correctly decoded TiFs to the total number $(m-n+1)$ of the TiFs. Moreover, we call $(i_k^* - k)$ the *decoding error*. For the measurement results presented in this section, we set m to be 30,000, which corresponds to ten minutes of data. In actual deployments of the TiF-based clock synchronization, the setting of m for the decoding algorithm can be much shorter. A detailed guideline for setting m will be discussed in Section 5.3. Thus, by setting $m = 30{,}000$ in our empirical study, the measured probability of correct decoding gives a lower bound of the actual probability of correct decoding when the setting of m is smaller.

4.2.2 Measurement Results on a Building Floor. We conduct extensive measurements on an entire floor of an office building that seats around 100 office employees. Figure 8 shows the floor plan. All the power outlets on this floor are branched from a main power panel and wired to the three phases of R, Y, and B of the utility grid. We select 16 power outlets as our test points, which are marked in Figure 8. The label color of a test point represents the grid phase that the corresponding power outlet is on. In each test, we connect two units of the hardware prototype to the two selected test points, respectively.

In the first set of tests, the two test points in each test are on the same phase. Table 1 shows the probability of correct decoding under different settings of n, that is, the length of the TiF. When

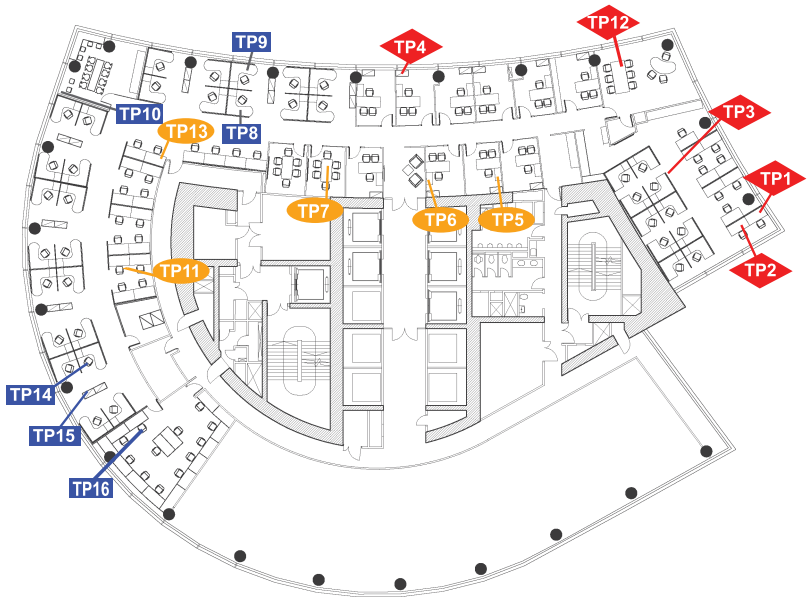


Fig. 8. Floor plan of the floor in an office building with test points marked. The label shapes and colors represent the grid phases of the electrical wiring. Specifically, the test points TP1, TP2, TP3, TP4, and TP12 with red diamond marks are on the R-phase; TP5, TP6, TP7, TP11, and TP13 with yellow oval-shaped marks are on the Y-phase; TP8, TP9, TP10, TP14, TP15, and TP16 with blue rectangular marks are on the B-phase.

Table 1. Probability of Correct Decoding for Different Test Points on the Same Phase on the Building Floor in Figure 8

	Test points		Correct decoding probability			
Phase	Node A	Node B	$n = 100^*$	$n = 200$	$n = 400$	$n = 800$
R	TP1	TP2	1	1	1	1
	TP1	TP3	0.950	1	1	1
	TP1	TP4	0.997	1	1	1
	TP3	TP4	1.0	1	1	1
Y	TP7	TP6	1	1	1	1
	TP7	TP11	1	1	1	1
	TP6	TP7	0.896	0.998	1	1
	TP7	TP11	0.827	0.999	1	1
B	TP8	TP9	1	1	1	1
	TP8	TP10	1	1	1	1

*n is the length of the TiF.

$n \geq 400$, the probability is one. This result suggests that if A samples a TiF for at least 8 seconds, the time instant at which each measurement in the TiF is sampled by A can be exactly identified by B. Since each ac cycle length measurement is represented by 2 B, the raw data volume of a TiF of length of 400 is 0.8 kB only.

In the second set of tests, the two test points in each test are on different phases. Table 2 shows the correct decoding probability with a TiF length of up to 6,400. We can see that, although the

Table 2. Probability of Correct Decoding for Different Test Points on Different Phases on the Building Floor in Figure 8

Node *A*		Node *B*		Correct decoding probability							
Test point	Phase	Test point	Phase	n=50	n=100	n=200	n=400	n=800	n=1,600	n=3,200	n=6,400
TP1	R	TP5	Y	0.000	0.000	0.000	0.005	0.013	0.097	0.385	0.660
TP1	R	TP6	Y	0.001	0.002	0.003	0.011	0.019	0.116	0.205	0.260
TP7	Y	TP8	B	0.002	0.007	0.020	0.067	0.108	0.161	0.205	0.245

Table 3. Correct Decoding Probability for Different Test Points on Different Floors

Node A		Node B		Correct decoding probability							
Test point	Phase	Test point	Phase	n=50	n=100	n=200	n=400	n=800	n=1,600	n=3,200	n=6,400
Apex	R	TP4	R	0.106	0.295	0.532	0.683	0.849	0.979	1	1
		TP6	Y	0	0.003	0.012	0.042	0.097	0.295	0.319	0.528
		TP8	B	0.011	0.031	0.093	0.179	0.302	0.428	0.505	0.791

correct decoding probability increases with n, it remains low. This observation suggests that the ac cycle length fluctuations on different grid phases have much lower correlation compared with those on the same grid phase as shown in Table 1. This is mainly because the changes of load connected to the three phases of the grid are not fully correlated, leading to a certain degree of independence among the grid frequencies on the different phases. This important observation poses a challenge in designing the TiF-based clock synchronization system, because prior knowledge of the grid phases of *A* and *B* will be needed. In Section 5, we will present an approach for the system to identify autonomously each network node's grid phase.

4.2.3 Measurement Results Across Different Floors and Geographic Locations. The floor shown in Figure 8 is the 8th floor of an office building. We also deploy a node in a room called Apex on the 13th floor of the same building. Apex is on the R-phase. Table 3 shows the correct decoding probability when we decode the TiFs collected in Apex using the traces collected at TP4, TP6, and TP8 on the floor shown in Figure 8. These three test points are on different phases. We can see that when the TiF length is 3,200, which is about one minute of data, the correct decoding probability is 1 when *A* and *B* are on the same phase. Compared with the results in Table 1, the TiF needs to be longer to achieve correct decoding between the two different floors. This result is consistent with the intuition that the ENV correlation decreases as the distance of the power network path between the two test points increases, because the transients of load can have localized effects on the grid frequency [13]. Moreover, the correct decoding probabilities across different phases are lower than the corresponding probabilities for the same phase, which is consistent with the results in Section 4.2.2.

In 2016, we deployed nodes at widely separated geographic locations in Singapore, that is, TP-A to TP-D in Figure 9. The test point TP-A in Figure 9 is the floor shown in Figure 8. TP-B, TP-C, and TP-D are within four buildings; they are on the R-phase, B-phase, and Y-phase of Singapore's utility grid, respectively. The line-of-sight distances from TP-A to the other three test points are from 9 km to 12 km. In particular, at TP-D, we deploy a Wi-Fi extender that is based on power-line communication (PLC), at the same power extension cord that our node is connected to. This helps us understand whether PLC affects our hardware. Table 4 shows the correct decoding probability when *A* is at three remote test points and *B* is at a TP-A's test point that is on the same phase

Fig. 9. The locations of six distributed test points in Singapore. The line-of-sight distances from TP-A to other test points are from 9 to 12 km. (Image credit: Google Map).

Table 4. Correct Decoding Probability Across Different Geographic Locations on the Same Phase

Node *A*		Node *B* at TP-A		Correct decoding probability						
Test point	Phase	Test point	Phase	n=400	n=800	n=1,600	n=3,200	n=6,400	n=12,800	n=20,000
TP-B	R	TP4	R	0.48	0.652	0.78	0.82	0.83	1	1
TP-C	B	TP8	B	0.34	0.475	0.63	0.8	0.99	1	1
TP-D	Y	TP6	Y	0.143	0.23	0.39	0.69	0.95	0.98	1

Table 5. Correct Decoding Probability Across Different Geographic Locations on Different Phases

Node *A*		Node *B* at TP-A		Correct decoding probability						
Test point	Phase	Test point	Phase	n=400	n=800	n=1,600	n=3,200	n=6,400	n=12,800	n=20,000
TP-E	R	TP4	R	0.26	0.45	0.62	0.80	0.97	1	1
TP-E	R	TP8	B	0.28	0.10	0.25	0.40	0.55	0.69	0.55
TP-E	R	TP6	Y	0.06	0.16	0.27	0.29	0.24	0.80	0.80

as *A*. We can see that for geographic distances within Singapore, a TiF length of 20,000, which corresponds to about 6.7 minutes of data, is needed to achieve correct decoding. At TP-D, the Wi-Fi extender's PLC does not affect the decoding owing to the low-pass filter in our design. Many critical infrastructures are deployed within limited geographic areas, for example, within a building or in a factory area. Nevertheless, the measurement results in Table 4 show that the TiF remains effective for Singapore-scale geographic distances. The raw data volume of a TiF of the length of 20,000 is 40 kB. The transmission of this amount of data collected over 6.7 minutes imposes little overhead on today's cyber networks.

In 2017, we deployed two additional nodes at TP-E and TP-F, shown in Figure 9. Table 5 shows the correct decoding probability when *A* is at TP-E, which is on the R-phase, and *B* is at three TP-A test points on different phases. Consistent with the results in Table 2 and Table 3, when the two nodes are on different phases, the correct decoding probabilities remain low. More evaluation results at these two test points will be presented in Section 7.

Table 6. Synchronism of ZCIs at Different Locations

Year	Measurement duration	Node A		Node B		$t_1 - t_2$	
		Test point	Phase	Test point	Phase	mean (μs)	s.d. (μs)
2016	1 hour	TP10	B	TP8	B	6	2
2016	1 hour	Apex	R	TP4	R	70.6	1.6
2016	1 hour	TP-C	B	TP8	B	85	18
						156	17
						118	14
2016	100 hours	TP-C	B	TP8	B	132	63
2017	100 hours	TP-C	B	TP8	B	120	53
2017	2 weeks	TP-E	R	TP4	R	188	62

4.3 Synchronism of ZCIs

In this section, we evaluate the synchronism of the ZCIs for the same grid phase at different locations. To improve measurement accuracy, we connect the GPS receiver's PPS output to a digital pin of the MCU, such that the MCU can accurately calibrate its clock and timestamp the ZCI interrupts from the Schmitt trigger. After calibrating two nodes by connecting them to the same power extension cord and measuring the biases between them, we deploy these two nodes at different test points to evaluate the synchronism of their ZCIs. As illustrated in Figure 1, t_1 and t_2 denote the timestamps of the ZCIs generated by the two nodes, respectively. Because of the phase characteristics of the impedance of power lines, the phase shift $(t_1 - t_2)$ is often nonzero and we measure this phase shift to characterize the synchronism of the ZCIs.

Table 6 shows the mean and s.d. of $(t_1 - t_2)$ when the two nodes are deployed at different test points on the same grid phase. When they are on the same building floor shown in Figure 8 (i.e., TP10 and TP8), the phase shift has a mean value of 6 μs and s.d. of 2 μs. The small s.d. suggests the stability of the phase shift. For the test point Apex on the 13th floor and TP4 on the 8th floor of the same building, the mean value increases to 70.6 μs. We also measure the phase shifts between TP-C and TP-A, which are about 10 km apart, during three time slots on one day. The mean value ranges from 85 μs to 156 μs. The change of the phase shift may be caused by the change of load distribution in the power grid [13]. Additional measurement results obtained over longer periods of time are presented in the last three rows of Table 6. They will be discussed in Section 7 in detail.

4.4 Summary and Implications

We can draw the following three important conclusions from the above extensive measurement results.

First, the measurements validate the TiF within Singapore. They also provide guidance on setting the TiF length. When the nodes reside within a local power distribution tree network rooted at a power panel, a TiF length of 400 appears to be enough. For nodes separated by up to 10 km, a TiF length of up to 20,000 may be needed.

Second, by decoding a TiF captured by node A, node B can identify the ac cycle within its trace that corresponds to a given ac cycle in A's TiF. Moreover, as shown in Table 6, the time offsets between an ac cycle's ZCIs detected at different locations are generally below 200 μs. Thus, if the two nodes can handle the ZCIs without delay in timestamping the ac cycle length measurements, using the correspondence of ac cycles given by the TiF decoding, B will be able to determine the offset between A's and B's clocks with a 200 μs accuracy. Thus, submillisecond accuracy clock synchronization is possible.

Third, time delays in transmitting A's TiF to B does not affect the result of the TiF decoding. The synchronization is thus resilient against packet-delay attacks. Conventional cryptographic techniques (e.g., signed messages or message digests) can be applied to ensure the integrity of the TiF itself during network transmissions.

In summary, high-resolution TiF provides a highly promising basis for accurate and secure clock synchronization for a network system connected to the same utility grid. In contrast, NTP's synchronization accuracy depends a lot on network conditions and it is often on the order of milliseconds or even tens of milliseconds. Although PTP can achieve submillisecond accuracy, it requires special hardware support, including PTP-enabled network interface cards at the hosts and all the switches and routers along the network path. Thus, in practice, PTP is often used in Ethernet LANs only. It is also important to note that both NTP and PTP are susceptible to packet-delay attacks, whereas the proposed system is not.

5 ACCURATE AND SECURE CLOCK SYNCHRONIZATION

Based on the key observations in Section 4, this section presents the design of an accurate and secure clock synchronization approach for critical infrastructures. Specifically, Section 5.1 describes our threat model of the packet-delay attack; Section 5.2 overviews our approach; and Section 5.3 provides an analysis of our approach regarding its security against the packet-delay attack.

5.1 Threat Model

Our threat model is the *packet-delay attack*. Specifically, we assume that the endpoints (master and slave) of a clock synchronization protocol are trustworthy. However, one or more attackers on a network path of the protocol's packets may delay the transmission of these packets. We assume that the total malicious delay for a packet is finite. Moreover, we assume that the protocol's packets cannot be tampered with because of cryptographic protection.

Our own experiments easily achieved the attack by executing a Linux traffic control command (i.e., `tc`) at either a legitimate router or a separate malicious host that impersonated the slave via a prior ARP spoofing attack in the victim slave's Ethernet LAN. The compromise of a key legitimate router may affect a large number of slaves in a network system. The ARP-based attack can affect selected victims in an Ethernet, for example, power meters and IEDs in a power grid substation.

As analyzed in [19, 28], the delay attack will introduce an additional synchronization error of $\frac{\tau_1-\tau_2}{2}$ for an NTP slave, where τ_1 and τ_2 are the malicious delays introduced to the NTP request and reply packets, respectively. For completeness of this article, Appendix A reviews the analysis of the attack impact. An attack injecting large delays may be detected by monitoring the total transmission time of the request and reply packets. However, an attacker who knows the attack detection methods can control the injected delays to bypass the detection. In Appendix A, we give conditions for bypassing two attack detection approaches: a timeout approach adopted by the NTP implementation and an approach based on exponential moving average. To the best of our knowledge, there is no solution to completely solve the packet delay attack for traditional clock synchronization protocols that exchange network messages containing solely local clock values of the slave/master.

5.2 Overview of Our Approach

This section presents the master–slave system architecture for our approach. Then, we present a method for the system to identify network nodes' grid phases autonomously.

5.2.1 Master–slave Architecture. As observed in Section 4.2.2, if two nodes are on different phases, the TiF decoding will have errors. To address this issue, we propose to adopt a

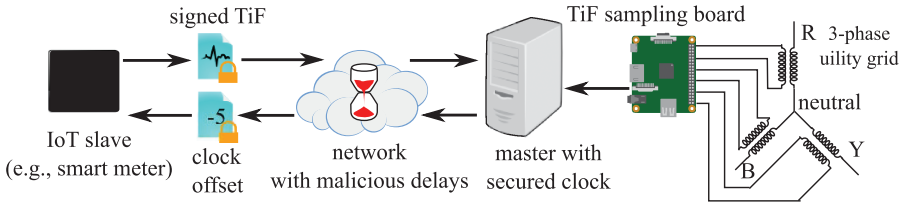


Fig. 10. System architecture and clock synchronization.

master–slave architecture as shown in Figure 10, in which a smart meter (either customer or industry class) is used as an example slave node. In the architecture, a centralized master is equipped with a TiF sampling board with three channels connected to the three grid phases, respectively, in which each channel is a set of the components shown in Figure 4. For instance, this board can be installed at the main power panel of a building. We assume that the master's clock is secured. (Special efforts to secure the master are practical, since a network system has one or at most only a few of these masters. Security of the master(s) will allow us to bootstrap the security of a much larger system.) If the system requires global synchronization, the master's clock can be synchronized securely with UTC, for example, using a GPS receiver that is geographically isolated from the outside with an air gap sufficient to prevent wireless spoofing attacks. The master timestamps its real-time ac cycle length measurements and stores them in a memory buffer. This data will be retrieved for processing synchronization requests from the slave nodes. A memory buffer of 100 MB is sufficient for storing data generated by the three sampling channels in the last 24 hours. One master may serve many slaves.

In a synchronization session, a slave (i) captures a TiF $\mathbf{x}$, (ii) timestamps it using the slave's clock value upon the ZCI of its last ac cycle, (iii) signs it for integrity, and (iv) transmits it to the master for clock synchronization. The slave performs the sampling only when it needs to resynchronize. Upon receiving the TiF, the master (i) checks its integrity based on the digital signature, (ii) decodes it using a trace of the latest ac cycle length measurements retrieved from the memory buffer, and (iii) sends back a signed packet containing the difference between the decoding output and the $\mathbf{x}$'s timestamp, which is the slave's clock offset. Finally, the slave sets and/or calibrates its clock using the received offset. A detailed analysis of this approach, including how to ensure security in the face of packet-delay attack, will be presented in Section 5.3.

For a network system spanning a large geographic area (e.g., a city), multiple masters can be deployed in a distributed anycast manner. A slave node may select a closest master for the best ZCI synchronism, for example. The deployment of masters will be further discussed in Section 8.2.

5.2.2 Autonomous Grid Phase Identification. The master has access to all three grid phases. In synchronizing with a slave, correct TiF decoding requires knowledge of which grid phase the slave is on. It is infeasible to manually label every network node in a large system. This section presents an autonomous grid phase identification method. It is based on a key observation from our measurements that, if nodes A and B are on the same grid phase, the decoding errors will be nearly all zero; otherwise, they are dispersed. Thus, we reuse the method for evaluating decoding errors in Section 4.2.1 to identify a slave's grid phase. Specifically, the slave captures and transmits m consecutive ac cycle lengths to the master. Upon receiving the data, the master retrieves the latest m consecutive ac cycle lengths on all three grid phases. Unlike the offline evaluation in Section 4.2.1 in which the two nodes' data traces are collected during exactly the same time period, this autonomous identification allows a small displacement of their time periods. For each pair of the slave's trace and the master's trace on a grid phase, we follow the method in Section 4.2.1

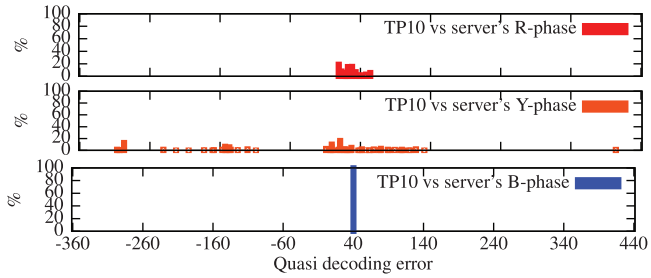


Fig. 11. PDFs of quasi-decoding errors for identifying TP10's grid phase.

to slide a window within the slave's trace to generate many *quasi-decoding errors* (they are quasi because they are not true decoding errors owing to the aforementioned displacement) and form a discrete probability density function (PDF) of the quasi errors. The slave's grid phase is identified as the master's grid phase that yields a PDF with the highest bar among all three PDFs. We note that this autonomous identification is a one-time procedure that should be executed when a network node is added to the system or it changes power supply. For instance, a device can initiate this procedure when it is powered up.

Figure 11 shows the PDFs generated by a master on the floor shown in Figure 8 when identifying the grid phase of a slave connected to the test point TP10 on B-phase. The slave transmits 1,000 cycle length measurements. The PDF generated with the master's B-phase data trace gives the highest bar. Hence, the identification is correct. When the slave is at the other test points in Figure 8, the identification results are all correct.

5.3 Security Analysis of Our Approach

5.3.1 Latency of a Synchronization Session. The latency of a synchronization session of our approach, denoted by t, is defined as the time from when the slave completes sampling $\mathbf{x}$ to when it receives the clock offset from the master. It characterizes how fast the slave can get resynchronized. The slave can measure t accurately using its own clock. We also analyze its breakdown as follows. Denote by t_1 the time delay (in ms) for the slave to sign the TiF $\mathbf{x}$ and by v the network speed (in kb/s) for transmitting $\mathbf{x}$. As each data point of $\mathbf{x}$ uses 16 bits, the time for transmitting $\mathbf{x}$ is $\frac{16n}{v}$ ms. Denote by t_2, t_3, and t_4 the time delays (in ms) for the master to verify the integrity of $\mathbf{x}$, decode it, and transmit the clock offset back to the slave, respectively. Then, $t \approx t_1 + \frac{16n}{v} + t_2 + t_3 + t_4$. In Section 7, we will measure t_1, t_2, t_3, and t_4.

We note that the offset returned by the master is for a past state of the slave (i.e., when the slave completed sampling $\mathbf{x}$). The slave's clock may have drifted further during the synchronization session. This observation is a key reason why we timestamp the TiF $\mathbf{x}$ upon the ZCI of the last cycle in $\mathbf{x}$, in order to maximize the freshness of the offset returned by the master. From our evaluation in Section 7, a synchronization session requires less than 1 s. Typical crystal oscillators found in MCUs and personal computers have drift rates of 30 to 50 ppm [9]. Thus, during the synchronization session, the slave's clock may have drifted for tens of microseconds. Even if we simply set the client's clock according to the returned offset, our approach can still achieve the submillisecond accuracy.

5.3.2 Security Against Packet Delay Attack. Let T denote the nominal ac cycle length. For instance, $T = 20$ ms for a 50 Hz grid. Upon receiving a TiF $\mathbf{x}$ from the slave, the master retrieves a trace of the latest ac cycle length measurements (denoted by $\mathbf{y}$) from its memory buffer to decode

$\mathbf{x}$. We set the length of $\mathbf{y}$ to be $m = n + L$, where L is a large enough number such that $L \gg \frac{t}{T}$. For instance, in our performance evaluation in Section 7, the measured t is around 1 s and we set $L = 1{,}000$ to make $L \gg \frac{t}{20\,\text{ms}}$. This setting ensures that the time duration of measuring $\mathbf{x}$ is within the time duration of measuring $\mathbf{y}$, that is, $\mathbf{x} \lhd \mathbf{y}$, which is a prerequisite for decoding $\mathbf{x}$. The attacker may delay the transmission of $\mathbf{x}$ to violate the requirement $\mathbf{x} \lhd \mathbf{y}$. We propose a countermeasure stated in the following proposition.

Proposition 5.1. *The slave discards the clock offset returned by the master if $\frac{t}{T} > L$, where t is the latency of the synchronization session measured by the slave. Under this strategy,*

(1) any packet-delay attack on the transmission of $\mathbf{x}$ that invalidates $\mathbf{x} \lhd \mathbf{y}$ will not affect the slave's clock;
(2) if $\frac{t}{T} \leq L$, the packet-delay attack has no effects.

Proof. Denote by $t_{s \to m}$ the time from when the slave completes sampling $\mathbf{x}$ to when the master receives $\mathbf{x}$, inclusive of the delay added by the attacker to the transmission of $\mathbf{x}$. A necessary condition for the packet-delay attack to invalidate $\mathbf{x} \lhd \mathbf{y}$ is $\frac{t_{s \to m}}{T} > L$. As $t > t_{s \to m}$, the attack must result in $\frac{t}{T} > L$. Vice versa, if $\frac{t}{T} \leq L$, the delay attack cannot invalidate $\mathbf{x} \lhd \mathbf{y}$ and thus has no effects on the completed synchronization session. □

The setting of L used by the master to decode $\mathbf{x}$ can be communicated to the slave together with the clock offset. Once the slave detects an attack that invalidates $\mathbf{x} \lhd \mathbf{y}$, it can notify the master in the next synchronization session. Depending on the (customizable) security policy, the master can increase the setting of L to contain the attack. We should also alert the system operator to investigate the attack. Appendix B provides pseudocodes for both the slave and the master that implement the above security safeguard and the autonomous grid phase identification in Section 5.2.2. The autonomous identification can similarly employ the above safeguard against the packet-delay attack.

We omit discussions of other attacks, such as impersonation and packet replay. These attacks generally can be solved using conventional security measures.

6 TIME FINGERPRINT DECODING ERROR CORRECTION

From our measurement study in Section 4.2, when the TiF length is sufficiently large, the empirical correct decoding probability is one. However, this does not preclude the possibility of decoding errors. To improve the robustness of clock synchronization against decoding errors, this section proposes an approach to cross-checking the decoding results among multiple nodes and correcting the decoding errors. We also analyze the error correction capability under several settings.

6.1 Decoding Error Correction Algorithm

Consider a system of N nodes: $\{n_0, n_1, \ldots, n_{N-1}\}$. Let δ_{ij} denote the offset between the clocks of n_i and n_j, which is unknown and to be estimated. Specifically, $\delta_{ij} = c_i(t) - c_j(t)$, where $c_i(t)$ and $c_j(t)$ are the clock values of n_i and n_j at any given time instant t. To simplify the discussion, we assume that δ_{ij} is time invariant. By designating n_0 as the reference node, we have that $\delta_{ij} = \delta_{0j} - \delta_{0i}$. In our approach, any pair of two nodes, n_i and n_j, will perform a synchronization session as described in Section 5.2 to measure δ_{ij}. If a TiF decoding error occurs, the measured clock offset, denoted by $\widetilde{\delta}_{ij}$, is given by $\widetilde{\delta}_{ij} = \delta_{ij} + e_{ij} + \epsilon_{ij}$, where e_{ij} is the decoding error, which is a multiple of the ac cycle length T, and ϵ_{ij} is the unknown phase shift. If the TiF decoding is correct, $\widetilde{\delta}_{ij} = \delta_{ij} + \epsilon_{ij}$. The $\widetilde{\delta}_{ij}$ is transmitted to a central node (e.g., n_0). We note that this $\widetilde{\delta}_{ij}$ can also be obtained through fusing the decoding results of multiple synchronization sessions between n_i and n_j, for example,

ALGORITHM 1: Decoding Error Correction Algorithm

Given: $\{\widetilde{\delta}_{ij} | \forall i, j \in [0, N-1], i \neq j\}$
Output: $\{\hat{\delta}_{ij}, \hat{e}_{ij} | \forall i, j \in [0, N-1], i \neq j\}$
1: $k = 0$
2: **while** $k \leq \frac{N(N-1)}{2}$ **do**
3: **for** each distribution of the k decoding errors among $\frac{N(N-1)}{2}$ synchronization sessions **do**
4: **if** the corresponding Equation (2) has a solution **then**
5: return $\{\hat{\delta}_{ij}, \hat{e}_{ij} | \forall i, j \in [0, N-1], i \neq j\}$
6: **end if**
7: **end for**
8: $k = k + 1$
9: **end while**

by a majority voting. However, any fusion rule cannot rule out the possibility that the fusion result $\widetilde{\delta}_{ij}$ still contains a decoding error. To correct such potential errors, the central node will fuse the clock offsets measured by a total of $\frac{N(N-1)}{2}$ node pairs. The error correction algorithm is described as follows.

Denote by $\hat{\delta}_{ij}$ and $\hat{e}_{ij}$ the estimates for δ_{ij} and e_{ij}, respectively. The $\hat{e}_{ij}$ is a multiple of the nominal ac cycle length. A general equations system that assumes that all measured clock offsets contain decoding errors is given by

$$\begin{cases} \hat{\delta}_{0j} + \hat{e}_{0j} = \widetilde{\delta}_{0j}, & \forall j \in [1, N-1]; \\ \hat{\delta}_{0j} - \hat{\delta}_{0i} + \hat{e}_{ij} = \widetilde{\delta}_{ij}, & \forall i, j \in [1, N-1], i \neq j. \end{cases} \tag{2}$$

The variables to be solved are $\{\hat{\delta}_{0j} | \forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij} | \forall i, j \in [1, N-1], i \neq j\}$, where the $\hat{\delta}_{0j}$ is the estimated clock offset between n_j and the reference node n_0 after decoding error correction. For the case that there are k decoding errors, we may keep k estimated decoding errors (i.e., $\hat{e}_{ij}$) in Equation (2) and remove the other estimated decoding errors. Thus, there will be $\binom{\frac{N(N-1)}{2}}{k}$ possible distributions of the k decoding errors among the $\frac{N(N-1)}{2}$ synchronization sessions. Algorithm 1 shows the pseudocode of the decoding error correction algorithm. It starts by assuming that there are no decoding errors (i.e., $k = 0$). In each iteration, that increases k by 1, it solves Equation (2) for all possible distributions of the k decoding errors. We note that, owing to the phase shifts, Equation (2) generally has no exact solutions. It can be solved by minimizing the overall residual of Equation (2), however, since the phase shifts are small (less than 1 ms, as measured in Section 4.3). Once a solution is found, Algorithm 1 returns.

6.2 Decoding Error Correction Capability

This section analyzes the maximum number of decoding errors that Algorithm 1 can correct. Let K denote the number of decoding errors. We say that Algorithm 1 can correct K decoding errors, if (i) $\forall k \in [0, K)$, Algorithm 1 does not return; and (ii) for $k = K$, Equation (2) has no solutions except when the tested distribution of the k decoding errors (Line 3) is identical to the actual distribution. To simplify discussion, the analysis below assumes zero phase shifts.

We observe that Algorithm 1 cannot correct a decoding error in a three-node system. This can be verified as follows. When $k = 0$, Equation (2) has no solutions no matter which $\widetilde{\delta}_{ij}$ contains a decoding error. When $k = 1$, Equation (2) may give a wrong solution. For instance, when the decoding error occurs between n_0 and n_1, Equation (2) gives a wrong solution when the decoding error is

assumed to occur between n_0 and n_2. Specifically, Equation (2) is $\{\hat{\delta}_{01} = \delta_{01} + e_{01}, \hat{\delta}_{02} + \hat{e}_{02} = \delta_{02}, \hat{\delta}_{02} - \hat{\delta}_{01} = \delta_{02} - \delta_{01}\}$, which gives a wrong solution of $\{\hat{\delta}_{01} = \delta_{01} + e_{01}, \hat{\delta}_{02} = \delta_{02} + e_{01}, \hat{e}_{02} = -e_{01}\}$.

Proposition 6.1. *Algorithm 1 can correct one decoding error, but not more than one, in a four-node system.*

Proof. *Case 1 ($K = 1$):* When $k = 0$, an exhaustive check shows that Equation (2) has no solutions no matter which two nodes the decoding error occurs between. When $k = 1$, an exhaustive check shows that Equation (2) has a correct solution only when the assumed decoding error distribution is identical to the actual distribution.

Case 2 ($K = 2$): We use a counterexample to prove that Algorithm 1 cannot correct two decoding errors. Consider two identical decoding errors $e_{01} = e_{02} = e$. When $k = 1$ and the decoding error is assumed to occur between n_0 and n_3, Equation (2) is $\{\hat{\delta}_{01} = \delta_{01} + e, \hat{\delta}_{02} = \delta_{02} + e, \hat{\delta}_{03} + \hat{e}_{03} = \delta_{03}, \hat{\delta}_{02} - \hat{\delta}_{01} = \delta_{02} - \delta_{01}, \hat{\delta}_{03} - \hat{\delta}_{01} = \delta_{03} - \delta_{01}, \hat{\delta}_{03} - \hat{\delta}_{02} = \delta_{03} - \delta_{02}\}$, which gives a wrong solution of $\{\hat{\delta}_{01} = \delta_{01} + e, \hat{\delta}_{02} = \delta_{02} + e, \hat{\delta}_{03} = \delta_{03} + e, \hat{e}_{03} = -e\}$.

Case 3 ($K = 3$): We also prove by a counterexample. Consider three decoding errors: $e_{01} = e_1$ and $e_{02} = e_{03} = e_2$. When $k = 2$ and the decoding errors are assumed to occur between n_1 and n_2, as well as n_1 and n_3, Equation (2) is $\{\hat{\delta}_{01} = \delta_{01} + e_1, \hat{\delta}_{02} = \delta_{02} + e_2, \hat{\delta}_{03} = \delta_{03} + e_2, \hat{\delta}_{02} - \hat{\delta}_{01} + \hat{e}_{12} = \delta_{02} - \delta_{01}, \hat{\delta}_{03} - \hat{\delta}_{01} + \hat{e}_{13} = \delta_{03} - \delta_{01}, \hat{\delta}_{03} - \hat{\delta}_{02} = \delta_{03} - \delta_{02}\}$, which gives a wrong solution of $\{\hat{\delta}_{01} = \delta_{01} + e_1, \hat{\delta}_{02} = \delta_{02} + e_2, \hat{\delta}_{03} = \delta_{03} + e_2, \hat{e}_{12} = \hat{e}_{13} = e_1 - e_2\}$.

Case 4 ($K \geq 4$): When $k = K$, Equation (2) is an underdetermined problem and Algorithm 1 cannot correct errors. □

From Proposition 6.1, by cross-checking six pairwise decoding results among four nodes, a decoding error can be corrected. Since the correct decoding probability is high with sufficiently large TiF length, the decoding error correction in a four-node system significantly improves the robustness of the clock synchronization. We will demonstrate this in Section 7. We leave the study of the decoding error correction capability of a general N-node system to future work.

7 PERFORMANCE EVALUATION

7.1 Experiment Settings

We have implemented the synchronization approach presented in Section 5. We use the node shown in Figure 4 as a slave. The RPi uses `OpenSSL` to sign the data to be transmitted using SHA256. For evaluation purposes, the slave node is integrated with a GPS receiver. The setup of the master is as follows. We use three RPi-based nodes and connect them to TP4, TP6, and TP8, respectively, shown in Figure 8, which are on different grid phases. Each of these nodes is equipped with a GPS receiver for global synchronization. Moreover, to improve the accuracy of timestamping at the master, we connect the GPS receiver's PPS output to a digital pin of the MCU and use the MCU to timestamp ac cycle length measurements. All three nodes stream their measurements to a tower server with an Intel Xeon 3 GHz quad-core processor. The decoding algorithm in Equation (1) can be parallelized since the computation for each iterator i in Equation (1) is independent. Our implementation divides the computation equally among four threads to fully utilize the quad cores. We note that, to serve many slave nodes, more computation resources can be allocated to maintain the timeliness of the decoding.

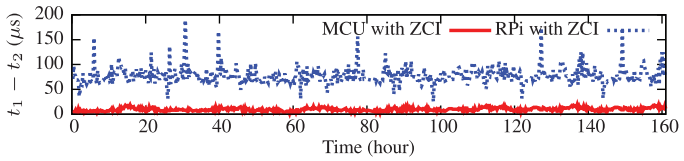
7.2 Experiment Results

7.2.1 Time Profiling. We measure the latency of the key steps of a clock synchronization session under two settings of the TiF length n, that is, $n = 400$ and $n = 20{,}000$, when the slave and master

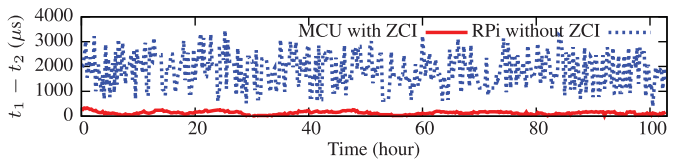
Table 7. Time Delays in a Clock Synchronization Session

n	m	t_1 (ms)	t_2 (ms)	t_3 (ms)	t_4 (ms)	Tx delay* (ms)	Total (ms)
400	1,400	126	6	10	0.2	*12.8*	*155*
20,000	21,000	129	6	168	4	*640*	*947*

*v = 500kb/s is used for estimating the TiF transmission delay. Italic numbers are estimates, as they depend on the setting of v.



(a) Within the building floor shown in Fig. 8



(b) Between TP-C and TP-A

Fig. 12. Clock synchronization errors over days.

nodes are deployed on the same floor of a building and in two buildings that are 10 km apart, respectively. Table 7 shows (i) the RPi's delay in signing a TiF (t_1); and (ii) the server's delay in verifying the TiF's integrity (t_2), decoding the TiF (t_3), and sending the offset (t_4). We note that the t_4 value when the two nodes are 10 km apart is four times that when they are on the same floor of a building. This is because they communicate through the Internet and a local low-latency Ethernet in the two experiments, respectively. The total latency, estimated based on v = 500 kb/s, is less than 1 s.

7.2.2 One-Week Evaluation within a Building Floor. We deploy a slave node at TP10, as shown in Figure 8. It can identify its grid phase correctly using the method in Section 5.2.2. It adopts a TiF length of 400 and resynchronizes every 15 minutes. Throughout the whole experiment, the master is always able to decode the slave's TiF correctly. Thus, we use the phase shift between the ZCIs detected by the slave and the master, respectively, as the metric of synchronization error. On the slave, the TiFs are timestamped by both the MCU and the RPi. Figure 12(a) shows the phase shifts between the master's ZCI and the slave's ZCI timestamped by the MCU and RPi, respectively. For the phase shifts measured by the MCU, the mean and s.d. are 9.8 μs and 3.3 μs, respectively. The difference between the two curves, which is around 80 μs, is caused by the Raspbian OS's delay in handling the ZCI at the slave. The results show that, if a network node can handle the ZCI without delay, it can achieve an average synchronization error of around 10 μs if the master resides within the same power distribution tree network.

7.2.3 Multiday Evaluation Among Nodes Distributed in Singapore. In 2016, we deployed a slave node at TP-C and synchronized it with the master at TP-A. For this slave node, we disconnect the ZCI from the RPi so that the RPi will timestamp a TiF upon receiving it from the USB. This setup evaluates the synchronization performance of a network node without an interface for handling

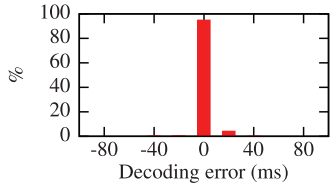


Fig. 13. PDFs of decoding errors between TP-E and TP-A in 2 weeks.

low-level interrupts. Figure 12(b) shows the results. The phase shifts measured by the MCU are generally below 200 μs, with mean and s.d. of 132 μs and 63 μs, respectively. Thus, between TP-A and TP-C, our approach can achieve an average synchronization error of about 0.1 ms. The RPi without ZCI experiences up to 3 ms error due to the USB communication delay. About one year after the above experiment, we conducted another experiment for 100 hours using the same setup. The mean and s.d. of the phase shifts are 120 μs and 53 μs, respectively. This shows that the phase shift profile is stable over time. We deploy an additional slave node at TP-E shown in Figure 9, which is on the R-phase. We evaluate the decoding errors between TP-E and TP-A over two weeks. During the evaluation period, the slave node samples 100 TiFs every hour and transmits to the master node at TP-A. Figure 13 shows the PDF of the decoding errors. The probability of correct decoding is 94.9%. Most of the decoding errors are 20 ms. Note that such decoding errors can be corrected by the approach in Section 6. The mean and s.d. of the phase shifts between TP-E and TP-A during the two weeks are 188 μs and 62 μs, respectively. The phase shift results are also summarized in the last three rows of Table 6.

7.2.4 TiF Decoding Error Correction. This section demonstrates the TiF decoding error correction presented in Section 6. Compared with nodes within the same building, nodes at widely separated geographic locations more likely have decoding errors. Thus, we conduct this set of experiments in Singapore. While the error correction algorithm in Section 6 assumes that all nodes are on the same grid phase, due to logistic constraints, we cannot find geographically distributed test points that are on the same grid phase. Instead, we use four nodes $\{n_0, n_1, n_2, n_3\}$ at TP-A (R-phase), TP-E (R-phase), TP-D (Y-phase), and TP-C (B-phase) in Figure 9, respectively. For a synchronization session between two nodes on different grid phases, we subtract a nominal phase shift from the measured clock offset to simulate the case that they are on the same grid phase. The details are as follows.

In the context of TiF decoding, the phase shift between two grid phases — say, A-phase and B-phase — is empirically determined as the time offset between A-phase's and B-phase's TiF traces of the same length that gives the largest similarity metric RSSE. Figure 14 illustrates the three phases' ENVs measured by the four nodes. Based on the data collected by the four nodes, if we select an R-phase's TiF and a Y-phase's TiF that begin with the ac cycle marked by the red rectangular and the yellow rectangular, respectively, shown in Figure 14, the RSSE is maximized. As illustrated in the figure, the time offset between these two marked ac cycles is two-thirds of the ac cycle length. Thus, the nominal phase shift between R-phase and Y-phase is $13,333$ μs for decoding the TiF in our power grid. Figure 14 also illustrates the nominal phase shifts for other phase pairs. Table 8 summarizes the pairwise nominal phase shifts. Now, we explain an example of estimating the clock offset between n_0 and n_2 by decoding n_0's TiF using n_2's ac cycle length trace. Assume that n_0 and n_2 are actually synchronized perfectly. The TiF decoding will give a clock offset estimate of about two-thirds of the ac cycle length. Then, we rectify this estimate by subtracting the nominal phase shift of $13,333$ μs to generate a near zero clock offset estimate.

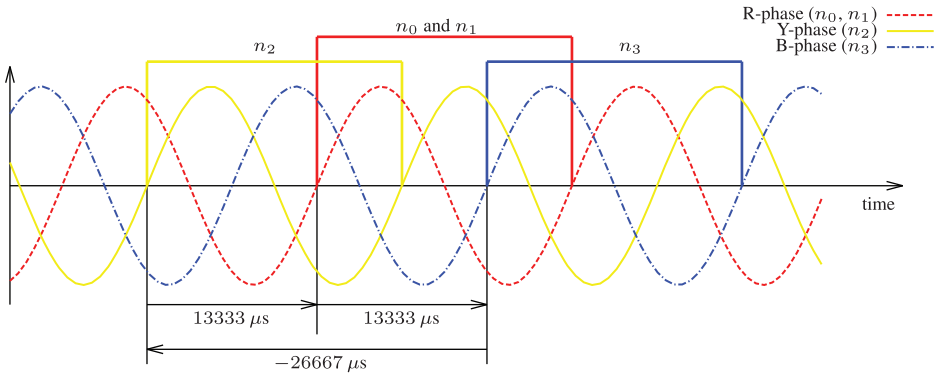


Fig. 14. Norminal phase shifts among different power grid phases.

Table 8. Nominal Phase Shifts and Correct Decoding Probabilities

Node pair	n_1, n_0	n_2, n_0	n_3, n_0	n_2, n_1	n_3, n_1	n_3, n_2
Nominal phase shift	0	13,333	−13,333	13,333	−13333	−26667
Correct decoding probability	0.954	0.952	1	0.958	1	0.320

The nominal phase shifts are in microseconds.

Table 9. Measured Clock Offsets and Error Detection Results

Round	$\widetilde{\delta}_{01}$	$\widetilde{\delta}_{02}$	$\widetilde{\delta}_{03}$	$\widetilde{\delta}_{12}$	$\widetilde{\delta}_{13}$	$\widetilde{\delta}_{23}$	Error detection
1	−97	45	−83	142	14	−127	no errors
2	−97	45	−83	142	14	19,880	error in $\widetilde{\delta}_{23}$
3	−97	45	−83	142	14	39,869	error in $\widetilde{\delta}_{23}$

The clock offsets are in microseconds.

The last row of Table 8 shows the empirical correct decoding probabilities measured through decoding 1,000 distinct TiFs with a length of 20,000. We can see that the empirical correct decoding probabilities are close to one, except between n_2 and n_3. Most decoding errors between n_2 and n_3 are ±20 ms. This facilitates demonstrating the decoding error correction. But we highlight that, in practice, decoding across different grid phases is not advisable owing to its lower correct decoding probability.

We apply Algorithm 1 to detect and correct errors for the four-node system. Table 9 shows the measured pairwise clock offsets in three rounds of experiments. We note that all four nodes are synchronized to UTC using GPS for ground-truth information. The ground truth allows us to observe decoding errors and evaluate the performance of Algorithm 1. In the first round, there are no decoding errors. Algorithm 1 returns when $k = 0$ (i.e., no errors detected). In the second and third rounds, a decoding error occurs between n_2 and n_3. Algorithm 1 can detect the decoding error and correct it.

Based on the results in Table 8, the empirical probability that no decoding errors occur during the six pairwise synchronization sessions is the product of the empirical correct decoding probabilities in the last row of Table 8, that is,

$$0.954 \times 0.952 \times 1 \times 0.958 \times 1 \times 0.320 = 0.278.$$

Moreover, as the empirical correct decoding probabilities for the pairs (n_3, n_0) and (n_3, n_1) are 1, the empirical probability that there is only one decoding error in the remaining four pairs, that is,

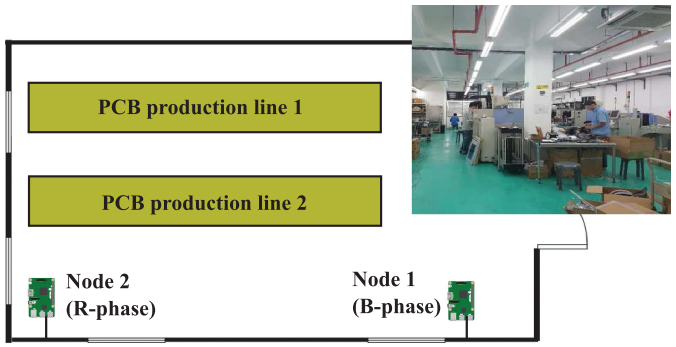


Fig. 15. Factory floor plan overlaid with a picture in the factory.

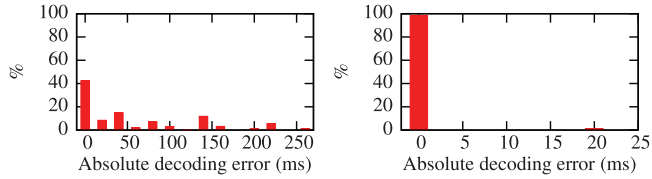
(n_1, n_0), (n_2, n_0), (n_2, n_1), (n_3, n_2), is

$$(1-0.954) \times 0.952 \times 0.958 \times 0.32 + 0.954 \times (1 - 0.952) \times 0.958 \times 0.320 + 0.954 \times 0.952 \times (1 - 0.958) \times 0.320 + 0.954 \times 0.952 \times 0.958 \times (1 - 0.320) = 0.631.$$

Thus, the probability that there are no decoding errors or only one decoding error is $0.278 + 0.631 = 0.909$. Since the four-node system can tolerate one decoding error, its systemwide clock synchronization will succeed with a probability of 0.909 despite the synchronization sessions between n_2 and n_3 having an error rate of 0.68. This shows that Algorithm 1 significantly improves the robustness. We reemphasize that these results are based on the inadvisable cross grid phase TiF decoding. If all decodings are performed within the same grid phase, the systemwide clock synchronization success probability will be extremely high, given that each node pair can already achieve high probabilities of correct decoding (see Table 4 and Figure 13).

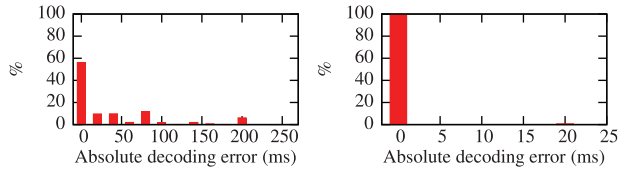
7.2.5 Experiments in a Factory Workshop. To evaluate the performance of our system in an industrial setting, we deploy two prototype nodes on a printed circuit board (PCB) manufacturer's factory workshop with two production lines. This factory workshop is in a 9-floor building that hosts a total of 109 factory workshops with various industrial loads. The venue of the factory building is marked as TP-F in Figure 9. Figure 15 shows the workshop's floor plan and the installation positions of the two nodes. The heavy loads of the two production lines are powered by the 415 V line-to-line voltages, while our prototype nodes sense the line-to-neutral voltages. Specifically, Node 1 and Node 2 in Figure 15 are connected to the B- and R-phases, respectively. Both nodes have sufficient sky views to receive GPS signals for ground-truth time. TiFs collected by the two nodes, when the production lines are on and off, respectively, are decoded using the B- and R-phase TiFs collected at TP-A. The distance between TP-A and TP-F is about 10 km. This allows us to evaluate the impact of nearby heavy industrial loads on the accuracy of our system. In total, 2,000 TiFs are collected by each node for four hours.

Figure 16 shows the PDFs of the absolute decoding errors between Node 1 and TP8 (B-phase) at TP-A, where the production lines are on and off, respectively. When the production lines are on, the probability of correct decoding is 0.42 and the maximum decoding error is about 220 ms. When the production lines are off, the probability of correct decoding is 0.98. Figure 17 shows the PDFs of the absolute decoding errors between Node 2 and TP4 (R-phase) at TP-A. The probabilities of correct decoding are 0.56 and 0.99, when the production lines are on and off, respectively. From these results, we can see that the heavy industrial loads may affect the TiFs captured by our prototype nodes. Note that our TiF capture device has a low-pass filter to remove high-frequency noises. However, heavy industrial loads may generate tiny and random perturbations to grid



(a) Production lines are on. (b) Production lines are off.

Fig. 16. PDF of decoding errors between Node 1 and TP8 at TP-A (B-phase).



(a) Production lines are on. (b) Production lines are off.

Fig. 17. PDF of decoding errors between Node 2 and TP4 at TP-A (R-phase).

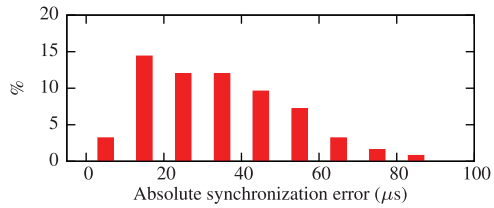
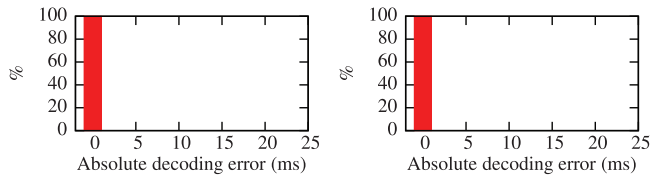


Fig. 18. PDF of absolute synchronization errors between Node 1 and Node 2.



(a) Production lines are on. (b) Production lines are off.

Fig. 19. PDF of decoding errors between Node 1 and Node 2.

voltage phases, which will negatively affect the synchronization accuracy between geographically distant nodes. These results suggest that our approach can achieve subsecond accuracy for two nodes 10 km apart in the presence of nearby heavy industrial loads.

The effect of load on the grid voltage generally does not propagate over long distances in power networks. To verify this, we evaluate the decoding errors between Node 1 and Node 2. Since they are on different phases, as in Section 7.2.4, we subtract a nominal phase shift of −13,333 μs from the decoding results. Figure 19 shows the PDF of the absolute decoding errors between the two nodes. The empirical probability of correct decoding is 1 irrespective of the operation state of the production lines. This confirms that the industrial loads generate local effects only. Figure 18 shows the PDF of the absolute synchronization errors between Node 1 and Node 2. The average synchronization error is 34 μs. Thus, if the slave nodes are in the vicinity of the master node, our system can still achieve high synchronization accuracy in industrial settings.

8 DISCUSSIONS

This section discusses several issues that need further research.

8.1 Battery-Based Wireless Sensors

The approach presented in this article is for sensor nodes directly connected to ac power lines. Thus, power consumption is not a concern for these nodes.

There are several research questions in applying our approach for battery-based wireless sensors. First, as these wireless sensors should sense the power line EMR, we need to study (i) the existence of TiF in power line EMR; (ii) the trade-off between the time accuracy and communication overhead (because intuitively higher time accuracy requires longer TiF); and (iii) other factors, including hardware cost, size constraints, and energy consumption for sensing. Second, to develop a secure clock synchronization approach, we need to study the security of the EMR sensing, for example, whether and how the EMR-based TiF is susceptible to practical wireless jamming. The first set of questions has been addressed in our recent work [15]. Specifically, we have shown that the EMR-based TiF with a data volume of 1.2 kB can achieve an average decoding error of about 0.5 s. By increasing the data volume of the TiF to 60 kB, the average decoding error can be reduced to about 0.2 s. Thus, compared with the approach presented in this article that achieves submillisecond accuracy with tens of kilobytes TiF, the EMR-based TiF is less accurate. We refer to [15] for details about hardware cost, size constraints, and energy consumption for sensing. However, the second set of questions—that is, the security of EMR sensing—are still open issues.

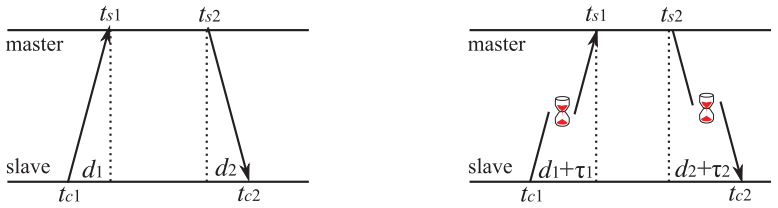
8.2 Deployment of Masters

From our experimental results obtained in nonindustrial settings (e.g., campuses, office buildings, and homes), the average synchronization errors are at submillisecond level for two nodes that are up to 10 km apart. This result gives guidance for the density of the masters deployed to provide clock synchronization services in nonindustrial settings. In our future work, we plan to conduct more extensive measurements in a larger-scale grid—for example, China's Southern Power Grid—to investigate the synchronization errors over longer geographic distances. The results will be useful for determining the density of the masters.

From the experiment results in Section 7.2.5, the heavy industrial loads generate local impact on the TiF. However, as also shown in Section 7.2.5, when the slave and the master are on the same factory floor, they can still achieve 100% empirical correct decoding probability. As most industrial systems (e.g., assembly lines) are located within a local area (e.g., a factory floor), deploying a master to serve the IoT nodes in the local area is practical. On the contrary, it is undesirable to synchronize these local IoT nodes with remote masters over the Internet, given the high criticality of industrial systems. This is because the synchronization could be affected by distributed denial-of-service (DDoS) attacks. Although the DDoS attacks can be easily detected, the attack isolation/mitigation—for example, by switching to new masters—may take time and still affect the industrial systems negatively. In our future work, we plan to conduct experiments in diverse industrial systems to understand comprehensively the impact of industrial loads on our approach. The results will provide better guidance for the deployment of masters under general industrial settings.

9 CONCLUSION

We identified and validated an important property of the periodic voltage signal in a utility power grid: namely, that the signal's cycle length fluctuations encode fine-grained global time information. Based on this key finding, we developed accurate clock synchronization with provable security against packet-delay attacks for a network system connected to the same grid. We also developed a synchronization error correction algorithm by cross-checking the synchronization



(a) No attack (b) Packet delay attack

Fig. 20. Illustration of NTP and packet-delay attack.

results among multiple nodes and proved that a four-node system can correct one erroneous synchronization session out of six synchronization sessions during a cross-check round. Extensive empirical evaluations show that our approach achieves an average synchronization error of 0.1 ms between two network nodes in an office building and a residential building that are 10 km apart, and 10 μs within the same floor of an office building. We also conducted experiments in buildings with heavy industrial loads. The results show that our approach can achieve subsecond synchronization accuracy when industrial loads are present in nearby power networks.

APPENDIX

A PACKET DELAY ATTACK AGAINST NTP

This appendix reviews the impact of packet-delay attack against NTP [19, 28] and discusses the challenges of dealing with it through analysis of a few foremost solution approaches.

Figure 20(a) illustrates the principle of NTP: t_{c1} and t_{c2} are the slave's clock values when it transmits an NTP request and receives a reply; t_{s1} and t_{s2} are the master's clock values when it receives the request and sends the reply. Let d_1 and d_2 denote the slave-to-master and master-to-slave transmission delays. Upon receiving the reply, the slave resets its clock to $t_{\text{set}} = t_{s2} + \frac{(t_{c2}-t_{c1})-(t_{s2}-t_{s1})}{2} = t_{s2} + \frac{d_1+d_2}{2}$. However, the actual time according to the master's clock when the slave receives the reply is $t_{\text{actual}} = t_{s2} + d_2$. Thus, the synchronization error is $t_{\text{set}} - t_{\text{actual}} = \frac{d_1-d_2}{2}$. Figure 20(b) illustrates the packet-delay attack, in which the transmissions of the request and reply are maliciously delayed by τ_1 and τ_2 seconds, respectively. Following the above analysis procedure, the synchronization error in the presence of the attack is $\frac{d_1-d_2}{2} + \frac{\tau_1-\tau_2}{2}$. Thus, the additional synchronization error introduced by the attack is $\frac{\tau_1-\tau_2}{2}$. If $d_1 > d_2$, the most effective attack is to delay the request packet only (i.e., $\tau_2 = 0$); otherwise, delaying the reply packet is the most effective (i.e., $\tau_1 = 0$).

We now discuss the challenges in detecting the attack and mitigating its impact. Our discussion is based on the Kerckhoffs's setting (i.e., the attacker knows the system, including its attack detection methods), which provides insights into the worst scenario and is necessary for critical industrial IoT systems. A *timeout* approach sets an upper bound for the total transmission time, that is, $(t_{c2} - t_{c1}) - (t_{s2} - t_{s1})$, to detect the attack. Thus, the malicious delay cannot exceed the timeout, limiting the offset caused by the attack. However, to address transient and natural network latency, large timeout settings on the order of seconds (e.g., 1 s in `ntp-4.2.6`) are often needed, which enables the attack to breach the submillisecond accuracy requirement of time-critical industrial applications.

We now discuss another approach that monitors the transient changes of the total transmission time to detect anomalies/attacks. Specifically, it compares the latest total transmission time with its exponential moving average (EMA) to accomplish detection. Widely used for such purposes, EMA helps the detector adapt to changing network conditions. Let $D[n]$ and $\bar{D}[n]$ denote the total transmission time and its EMA at the end of the nth synchronization session, where

$\bar{D}[n] = (1-\alpha)\bar{D}[n-1] + \alpha D[n]$ and $\alpha \in (0,1)$ is a constant. If $|D[n] - \bar{D}[n-1]| > \epsilon$, where ϵ is a positive threshold, the detector claims the presence of attack. We now show that an attack strategy that linearly increases the malicious delay over time can bypass the EMA-based detector. To simplify the analysis but still keep the essence of the problem, we assume that, in the absence of the attack, the total transmission time is a constant d. Let $\tau[n]$ denote the total malicious delay in the nth synchronization session. The attack strategy is $\tau[n] = n\tau$, where τ is a constant. As the EMA of the constant d is always d, we can safely ignore the constant d in the following bypass condition analysis. The closed-form expression of the EMA of the arithmetic progression $\tau[n]$, denoted by $\bar{\tau}[n]$, can be derived as $\bar{\tau}[n] = \tau(n - \frac{(1-\alpha)(1-(1-\alpha)^n)}{\alpha})$. Thus, $|\tau[n+1] - \bar{\tau}[n]| = \frac{1-(1-\alpha)^{n+1}}{\alpha} \cdot \tau < \frac{\tau}{\alpha}$. Therefore, by setting $\tau < \alpha\epsilon$, the attack strategy $\tau[n] = n\tau$ can keep bypassing the EMA-based attack detector until the total transmission time reaches the timeout.

The above discussions provide insights into understanding the challenges of completely eliminating the impact of packet-delay attacks solely based on the clock information collected by the master and the slave. Moreover, the setting of the timeout and the threshold ϵ still face a basic dilemma between accommodating network condition dynamics (to reduce false-alarm rate) and sensitivity to the attack (to reduce miss rate).

B PSEUDOCODE OF THE PROPOSED CLOCK SYNCHRONIZATION APPROACH

ALGORITHM 2: Master's Pseudocode

Given: Master's clock `CLK_M`, setting of L

```
1:
2:  // grid phase identification service
3:  event grid_phase_data received from slave do
4:      if verify_integrity(grid_phase_data) is OK then
5:          y⃗ = latest (m + L) cycle length measurements
6:          grid_phase_reply.phase =
                phase_identify(grid_phase_data, y) // cf. Section 5.2.2 in the paper
7:          grid_phase_reply.L = L
8:          sign and transmit grid_phase_reply to slave
9:      else
10:         alert packet integrity attack
11:     end if
12: end event
13:
14: // synchronization service
15: event sync_request received from slave do
16:     if verify_integrity(sync_request) is OK then
17:         if sync_request.under_delay_attack is true then
18:             apply a policy to decide whether to increase L
19:         end if
20:         y = latest (n + L) timestamped cycle length measurements
21:         sync_reply.offset = decode(sync_request.x⃗, y⃗) - sync_request.timestamp // see Section 4.2.1
            in the article
22:         sync_reply.L = L
23:         sign and transmit sync_reply to slave
24:     else
25:         alert packet integrity attack
26:     end if
27: end event
```

ALGORITHM 3: Slave's Pseudocode

Given: Slave's clock CLK_S

```
1:
2:  // whenever the slave is powered up, identify its grid phase
3:  event booted do
4:      under_delay_attack = false
5:      grid_phase_identified = false
6:      grid_phase_data = m consecutive ac cycle lengths
7:      start_time = CLK_S
8:      sign and transmit grid_phase_data
9:  end event
10:
11: // get grid phase identification result
12: event grid_phase_reply received from master do
13:     if verify_integrity(grid_phase_reply) is OK then
14:         t = CLK_S - start_time
15:         if t/T > grid_phase_reply.L then
16:             alert packet delay attack and initialization failure
17:         else
18:             grid_phase = grid_phase_reply.phase
19:             grid_phase_identified = true
20:         end if
21:     else
22:         alert packet integrity attack and initialization failure
23:     end if
24: end event
25:
26: // send synchronization request
27: command resync( ) do
28:     if grid_phase_identified is true then
29:         sample a time-fingerprint x, timestamp x using CLK_S upon ZCI of the last ac cycle in x
30:         start_time = CLK_S
31:         sync_request = {x, x's timestamp, grid_phase, under_delay_attack}
32:         sign and transmit sync_request to master
33:     end if
34: end command
35:
36: // receive synchronization reply and set clock
37: event sync_reply received from master do
38:     if verify_integrity(sync_reply) is OK then
39:         t = CLK_S - start_time
40:         if t/T > sync_reply.L then
41:             under_delay_attack = true
42:             alert packet delay attack
43:         else
44:             under_delay_attack = false
45:             CLK_S = CLK_S + sync_reply.offset
46:         end if
47:     else
48:         alert packet integrity attack
49:     end if
50: end event
```

REFERENCES

[1] Adafruit. 2017. Adafruit Ultimate GPS HAT. Retrieved April 24, 2018 from https://www.adafruit.com/products/2324.

[2] Argonne National Laboratory. 2008. GPS is Easy to Spoof. Retrieved April 24, 2018 from http://www.ne.anl.gov/capabilities/vat/spoof.html.

[3] Yin Chen, Qiang Wang, Marcus Chang, and Andreas Terzis. 2011. Ultra-low power time synchronization using passive radio receivers. In *10th International Conference on Information Processing in Sensor Networks (IPSN'11)*. IEEE, Chicago, IL, USA, 235–245.

[4] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review* 36, SI, 147–163.

[5] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. 2003. Timing-sync protocol for sensor networks. In *1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. ACM, Los Angeles, CA, USA, 138–149.

[6] Ravi Garg, Avinash L. Varna, and Min Wu. 2011. Seeing ENF: Natural time stamp for digital video via optical sensing and signal processing. In *ACM Multimedia (MM)*. ACM, Scottsdale, AZ, USA, 23–32.

[7] Catalin Grigoras. 2007. Applications of ENF criterion in forensic audio, video, computer and telecommunication analysis. *Forensic Science International* 167, 2, 136–145.

[8] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. 2010. ElectriSense: Single-point sensing using EMI for electrical event detection and classification in the home. In *12th ACM International Conference on Ubiquitous Computing (UbiComp'10)*. ACM, Copenhagen, Denmark, 139–148.

[9] Tian Hao, Ruogu Zhou, Guoliang Xing, and Matt Mutka. 2011. WizSync: Exploiting Wi-Fi infrastructure for clock synchronization in wireless sensor networks. In *32nd IEEE Real-Time Systems Symposium (RTSS'11)*. IEEE, Vienna, Austria, 1379–1392.

[10] Arik Hesseldahl. 2014. Hackers Infiltrated Power Grids in U.S., Spain. Retrieved April 24, 2018 from http://on.recode.net/1m6E3Le.

[11] Stamatis Karnouskos. 2011. Stuxnet worm impact on industrial cyber-physical system security. In *37th Annual Conference of IEEE Industrial Electronics (IECON'11)*. IEEE, Crown Conference Centre, Melbourne, Vic, Australia, 4490–4494.

[12] Kaspersky Lab. 2017. Datesheet: Five myths of industrial control systems security. Retrieved April 24, 2018 from http://media.kaspersky.com/pdf/DataSheet_KESB_5Myths-ICSS_Eng_WEB.pdf.

[13] Prabha Kundur, Neal J. Balu, and Mark G. Lauby. 1994. *Power System Stability and Control*. Vol. 7. McGraw-Hill, New York.

[14] Liqun Li, Guoliang Xing, Limin Sun, Wei Huangfu, Ruogu Zhou, and Hongsong Zhu. 2011. Exploiting FM radio data system for adaptive clock calibration in sensor networks. In *9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11)*. ACM, Washington, DC, USA, 169–182.

[15] Yang Li, Rui Tan, and David Yau. 2017. Natural timestamping using powerline electromagnetic radiation. In *IPSN*.

[16] Zhenjiang Li, Wenwei Chen, Cheng Li, Mo Li, Xiang-Yang Li, and Yunhao Liu. 2012. Flight: Clock calibration using fluorescent lighting. In *18th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, Istanbul, Turkey, 329–340.

[17] Qing Yang Lin Huang. 2015. GPS spoofing – Low-cost GPS simulator. In *DEF CON®23 Hacking Conference (DEFCON23)*. DEF CON Communications, Inc., Las Vegas, NV, USA. https://bit.ly/22H2XTA.

[18] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. 2004. The flooding time synchronization protocol. In *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, Baltimore, MD, USA, 39–49.

[19] Tal Mizrahi. 2012. A game theoretic analysis of delay attacks against time synchronization protocols. In *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS'12)*. IEEE, San Francisco, CA, USA, 1–6.

[20] T. Mizrahi. 2014. Security Requirements of Time Protocols in Packet Switched Networks. Retrieved April 24, 2018 from https://tools.ietf.org/html/rfc7384.

[21] North American Energy Standards Board. 2005. Manual Time Error Correction. Retrieved April 24, 2018 from https://www.naesb.org//pdf2/weq_bklet_011505_tec_mc.pdf.

[22] Tekla Perry. 2011. Planned U.S. Power System Experiment Means Some Clocks Will Speed Up. IEEE Spectrum. Retrieved April 24, 2018 from http://bit.ly/1OXu2zZ.

[23] Anthony Rowe, Vikram Gupta, and Ragunathan Raj Rajkumar. 2009. Low-power clock synchronization using electromagnetic energy radiating from AC power lines. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, Berkeley, CA, USA, 211–224.

[24] Richard W. Sanders. 2008. Digital audio authenticity using the electric network frequency. In *Audio Engineering Society Conference: 33rd International Conference: Audio Forensics-Theory and Practice*. Audio Engineering Society, Denver, CO, USA.

[25] Daniel P. Shepard, Todd E. Humphreys, and Aaron A. Fansler. 2012. Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks. *International Journal of Critical Infrastructure Protection* 5, 3, 146–153.

[26] John A. Stankovic. 2014. Research directions for the Internet of Things. *IEEE Internet of Things Journal* 1, 1, 3–9.
[27] TI. 2015. Smart grid and energy solution guide. Retrieved April 24, 2018 from http://www.ti.com/lit/sl/slym071o/slym071o.pdf.
[28] Markus Ullmann and M. Vogeler. 2009. Delay attacks – Implication on NTP and PTP time synchronization. In *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS'09)*. IEEE, Brescia, Italy, 1–6.
[29] Hans Weibel. 2012. Tutorial: IEEE 1588 standard for a precision clock synchronization protocol and synchronous Ethernet. French National Institute of Nuclear and Particle Physics (IN2P3). http://www.in2p3.fr/actions/formation/Numerique12/IEEE_1588_Tutorial_IN2P3_Handout.pdf.
[30] World Economic Forum. 2015. Industrial Internet of Things: Unleashing the Potential of Connected Products and Services. Retrieved April 24, 2018 from http://www3.weforum.org/docs/WEFUSA_IndustrialInternet_Report2015.pdf.

Received August 2017; revised February 2018; accepted March 2018