Exploiting Power Grid for Accurate and Secure Clock Synchronization in Industrial IoT

Sreejaya Viswanathan
Advanced Digital Sciences Center (ADSC), Illinois at Singapore

Rui Tan*
Nanyang Technological University, Singapore

David K. Y. Yau
Singapore University of Technology and Design; ADSC

***Abstract*—Desynchronized clocks among nodes in industrial Internet of Things (IoT) can degrade system performance and even lead to safety incidents. Clock synchronization protocols based on network message exchanges, though widely used in current industrial 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 paper proposes to use the sine voltage waveform of a utility power grid to synchronize "things" connected to the same grid. Our experiments demonstrate that minute fluctuations of the voltage's cycle length encode fine-grained global time information in a city-scale utility grid. Based on this key result, we develop a clock synchronization approach that achieves sub-ms 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 IoT nodes that are 10 km apart. When the proposed system is deployed within the same floor of a building, the error reduces to 10 μs.**

I. Introduction

Modern industries are embracing the vision of Internet of Things (IoT) [1], which provides a fabric that connects advanced sensing, computing, communications, and actuation [2]. For the "things" (i.e., network nodes) in an industrial IoT system, trustworthy time information can be critical. Accurate timestamps of data allow us to make sense of the data relative to extrinsic events, and synchronized time 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 sub-ms accuracy. Stale measurements will result in erroneous control that endangers the grid's safety [3]. Desynchronization of consumer smart meters may also lead to instability of a real-time energy market [4]. In car manufacturing, desynchronized robots in a Roboteam [5] working on a same car can cause clashes of their arms and disrupt the production pipeline.

*Corresponding author.

In existing industrial systems, 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 [6], 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 [7], [8]. 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 [7], [8]. To the best of our knowledge, there is no solution to completely solve this attack. The effectiveness of existing mitigation approaches [6], 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 an industrial system may harden the 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 [9]) using low-cost (e.g., $300 [10]) hardware.

Industrial systems have also resorted to cyber isolation to protect their networks. Such isolation is shaky, however [11]. Zero-day vulnerability exploits, insider attacks, and stepping stone attacks can render the isolation futile, as evidenced in recent high-profile intrusions including Dragonfly [12] and Stuxnet [13] against power and nuclear plants. For the problem context of this paper, 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 IoT will only further lower the barriers of penetrating industrial systems. Hence, providing secure clock synchronization in industrial IoT, where connectivity is a defining characteristic, is an imperative research problem.

Recent research efforts have investigated clock synchro-

nization in wireless sensor networks and mobile/pervasive computing. These approaches leverage the nodes' built-in radios [14], [15], [16] or external wireless time broadcasts or periodic signals found in AM/FM radios [17], [18], Wi-Fi beacons [19], power line electromagnetic radiation [20], and fluorescent light flickering [21]. However, they are designed without security considerations. Moreover, reliance on wireless electromagnetic signals in a critical industrial context often raises reliability and security concerns, due to the possibility of wireless jamming and spoofing.

In this paper, we advance a desirable notion of *inherent security* in providing secure and accurate time in time-critical industrial IoT 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 an industrial IoT 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 sub-ms average error in the clock synchronization. Second, in industrial settings, many things 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 unforgeable physical signal that is practically infeasible 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 huge amount of energy, which raises insurmountable barriers economically and logistically for would-be attackers.

Subject to the grid-wide ENV, different IoT nodes can count the ac cycles of the ENV signal to achieve *clock calibration*[1] [18], [21], and remain 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 paper is the identification, validation, and exploitation of a novel physical fingerprint embedded in the ENV signal, which we call *time fingerprint* (TiF), that provides resilience against the 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 [22], 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 node, say A, can be correctly *time-aligned* within a trace of ac cycle lengths captured by another node, say B, at the granularity of an ac cycle.

The above key observations enable a novel approach to achieving the objectives of accurate and secure clock synchronization simultaneously. 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 transmission delays. This principle that applies for a pair of nodes underlines a complete clock synchronization system among all the grid-connected nodes.

This paper presents a prototype implementation of our system and discusses its performance based on extensive empirical evaluations. The results show that our approach achieves an average synchronization error of 0.1 ms between two network nodes 10 km apart. When the proposed system is deployed within the same floor of a building, the error reduces to $10\,\mu$s.

The balance of the paper is organized as follows. §II reviews related work. §III presents the design and implementation of the TiF capture hardware. §IV presents extensive measurements that characterize key properties of the TiF under different deployment environments. §V presents the design and implementation of the proposed secure clock synchronization approach for industrial IoT. §VI presents evaluation results of the system prototype. §VII concludes.

II. 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 [14], TPSN [15], and FTSP [16]) achieves clock synchronization by exchanging radio messages among the nodes in question. The second category [17], [18], [19], [20], [21] exploits external wireless time broadcasts and periodic signals. Chen et al. [17] design 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. [18] exploit the Radio Data System

[1] *Clock calibration* ensures that different clocks will advance at the same speed; *clock synchronization* regulates the clocks to have the same value.

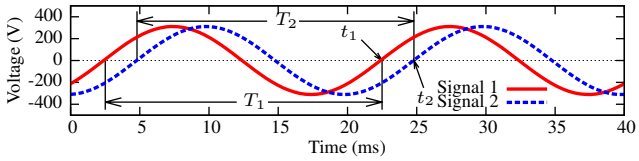


Figure 1. Illustration of ENV ac cycle length. T_1 and T_2 are two ac cycle length measurements.

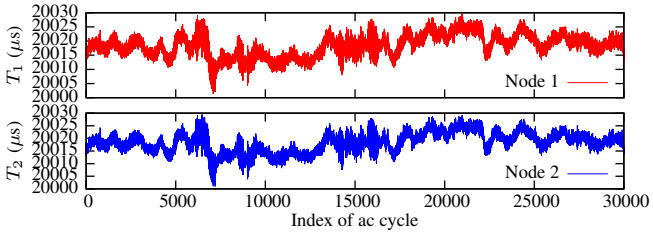


Figure 2. ENV ac cycle lengths measured by two nodes at different locations in a building for a duration of 10 minutes.

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 [19]. Rowe et al. [20] design a mote peripheral to receive periodic electromagnetic radiation from utility power lines and calibrate the clocks of motes based on the detected ac cycles. Li et al. [21] 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 grid-wide *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 communications and signals often raises reliability and security concerns for mission- and safety-critical systems. We do not use wireless signals.

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*, i.e., 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 [23], thus keeping the maximum error to be also about 2 s. Moreover, because this regulation may negatively impact power grid reliability [24], it is either not adopted or considered obsolete and being phased out. Grid time is therefore unsuitable for accurate synchronization of things 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 [25], [26], [27]. 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 forensics approaches sample the grid frequency every few seconds, the identification similarly has a temporal granularity on the order of seconds. In this paper, we solve the systems challenge of capturing the fine-grained ENV-based TiF and validate its ability to "encode" time information with sub-ms accuracy. We further apply the TiF to design a novel clock synchronization system that satisfies both the security and accuracy requirements for industrial IoT.

III. 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.

A. ENV Cycle Length Fluctuations

Fig. 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, due to characteristics of the electrical power lines [22]. An ac cycle length of the red solid ENV signal is the time period T_1 between two consecutive zero crossings illustrated in Fig. 1. In this paper, we design a hardware device to continuously measure the ac cycle lengths. Details of the design will be presented in §III-B. We deploy two of the hardware devices, Node 1 and Node 2, in two different rooms on the same floor of an office building. Fig. 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, Fig. 3 shows a zoomed-in view of Fig. 2, where the traces measured by both nodes are depicted over a selected window of one second. 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 Fig. 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

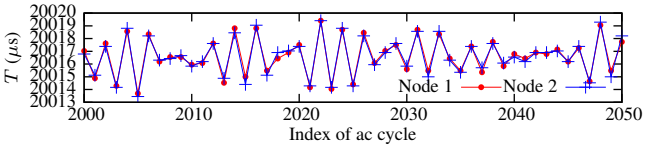


Figure 3. A zoomed-in view of Fig. 2.

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 §IV, we present extensive measurements under a wide range of settings (e.g., length of the TiF and physical distance between two synchronizing nodes up to city scale) to test these two hypotheses.

B. Time Fingerprint Capture Hardware

This section presents a hardware design for capturing minute fluctuations of the ENV cycle length as shown in Fig. 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.

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 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. Fig. 4 shows the schematics of our prototype hardware. Given the line-to-neutral utility voltage, the prototype uses an ac/ac adapter and a voltage divider to step down the voltage. 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 Fig. 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 [28]. To reduce their impact, we apply a Sallen-Key second-order low-pass filter to the

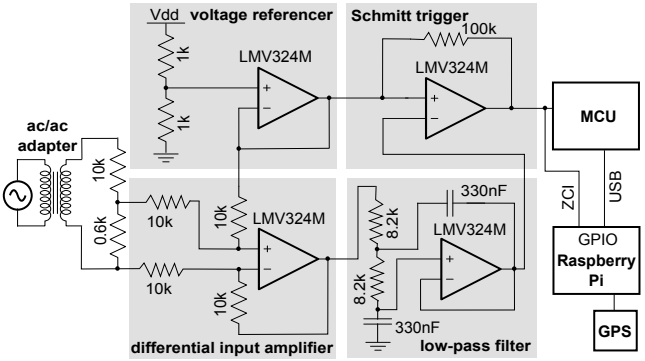


Figure 4. Schematics of the proposed hardware prototype. GPS and Raspberry Pi are used for evaluation purposes only, and can be eliminated or replaced in actual deployments.

single-ended signal. The cutoff 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 two bytes. 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 μs, as shown in Fig. 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.

2) Integration with IoT platforms: In Fig. 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 [29], to measure the cycle lengths. The TimerA provided by TinyOS on an MSP430-based kMote used in a smart plug platform can achieve a $4.2\,\text{MHz}$ clock rate after a reconfiguration of sourcing the SMCLK clock from the DCO clock without a divider. The achieved $0.24\,\mu$s time resolution is sufficient for capturing the ac cycle length fluctuations. Since the TiF capture hardware (i.e., the shaded components in Fig. 4) needs to access the ENV, we can integrate it into the power supply unit of industrial 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 μs resolution, to obtain a timestamp that it adds to the incoming cycle length measurement from the USB. To obtain accurate groundtruth time for the evaluation, we integrate the RPi with an Adafruit GPS receiver [30] 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 $1\,\mu$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.

IV. Measurement Study

This section presents extensive measurements using the hardware prototype in §III-B 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 §V with appropriate choice of the system parameters.

A. Time Fingerprint Decoding

The good match of the ac cycle length fluctuations observed in §III-A 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.

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 $\mathbf{x} \triangleleft \mathbf{y}$. How to ensure this condition is discussed in §V-C.

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 §V-C. 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, e.g., 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. Our extensive controlled experiments show that the small measurement biases of our prototype nodes do not affect the decoding result. The details of the experiments are omitted here due to space constraints and can be found in the long version of this paper [31].

We evaluate the accuracy of the decoding algorithm as follows. We deploy the hardware prototype at two nodes at different locations. By leveraging groundtruth timestamps from the integrated GPS receivers, we select two traces of ac cycle length measurements of length m that are respectively captured by the two nodes during the same time period. 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 Eq. (1) to decode each TiF. Let i_k^* denote the output of Eq. (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 §V-C. Thus, by setting $m = 30000$ 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.

2) Measurement results on a building floor: We conduct extensive measurements on an entire floor of an office

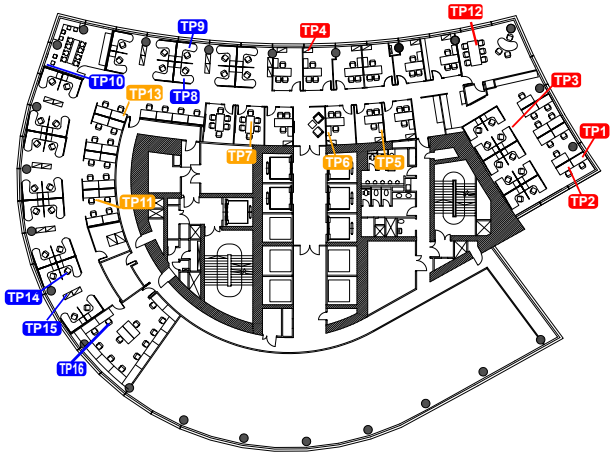


Figure 5. Floor plan of the floor in an office building with test points marked. The label colors represent the grid phases of the electrical wiring. Specifically, the test points TP1, TP2, TP3, TP4, and TP12 are on the R-phase; TP5, TP6, TP7, TP11, and TP13 are on the Y-phase; TP8, TP9, TP10, TP14, TP15, and TP16 are on the B-phase.

Table I
PROBABILITY OF CORRECT DECODING FOR DIFFERENT TEST POINTS ON THE SAME PHASE ON THE BUILDING FLOOR IN FIG. 5.

Phase	Test points		Correct decoding probability			
	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.

building that seats around 100 office employees. Fig. 5 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 Fig. 5. 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 respectively to the two selected test points.

In the first set of tests, the two test points in each test are on the same phase. Table I shows the probability of correct decoding under different settings of n, i.e., the length of the TiF. When $n \geq 400$, the probability is one. This result suggests that if A samples a TiF for at least eight 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 two bytes, 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

Figure 6. The locations of four distributed test points in Singapore. The line-of-sight distances from TP-A to TP-B, TP-C, and TP-D are from 9 to 12 km. (Image credit: Google Map.)

are on different phases. Table II shows the correct decoding probability with a TiF length of up to 6,400. We can see that, although the 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 I. 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 §V, we will present an approach for the system to identify autonomously each IoT node's grid phase.

3) Measurement results across different floors and geographic locations: The floor shown in Fig. 5 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 III 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 Fig. 5. 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 one when A and B are on the same phase. Compared with the results in Table I, 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 [22]. 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 §IV-A2.

We also deploy nodes at widely separated geographic locations in our city, as shown in Fig. 6. The test point TP-A in Fig. 6 is the floor shown in Fig. 5. TP-B, TP-C, and TP-D are within three buildings and they are on the R-phase, B-phase, and Y-phase of the city's utility grid, respectively.

Table II
PROBABILITY OF CORRECT DECODING FOR DIFFERENT TEST POINTS ON DIFFERENT PHASES ON THE BUILDING FLOOR IN FIG. 5.

Node A		Node B		Correct decoding probability							
Test point	Phase	Test point	Phase	n=50	n=100	n=200	n=400	n=800	n=1600	n=3200	n=6400
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 III
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=1600	n=3200	n=6400
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

Table IV
CORRECT DECODING PROBABILITY ACROSS DIFFERENT GEOGRAPHIC LOCATIONS SHOWN IN FIG. 6.

Node A		Node B at TP-A		Correct decoding probability									
Test point	Phase	Test point	Phase	n=50	n=100	n=200	n=400	n=800	n=1600	n=3200	n=6400	n=12800	n=20000
TP-B	R	TP4	R	0.089	0.201	0.343	0.48	0.652	0.78	0.82	0.83	1	1
TP-C	B	TP8	B	0.023	0.067	0.17	0.34	0.475	0.63	0.8	0.99	1	1
TP-D	Y	TP6	Y	0.013	0.035	0.074	0.143	0.23	0.39	0.69	0.95	0.98	1

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 IV 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 as A. We can see that for city-scale geographic distances, 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, due to the low-pass filter in our design. Most industrial IoT systems are deployed within limited geographic areas, e.g., within a building or in a factory area. Nevertheless, the measurement results in Table IV show that the TiF is still effective for city-scale distances. The raw data volume of a TiF of 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.

B. Synchronism of ZCIs

In this section, we evaluate the synchronism of the ZCIs for the same grid phase at different locations. To improve the 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 a 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 Fig. 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 non-zero and we measure this phase shift to characterize the synchronism of the ZCIs.

Table V
SYNCHRONISM OF ZCIS AT DIFFERENT LOCATIONS.

Node A		Node B		$t_1 - t_2$	
Test point	Phase	Test point	Phase	mean (μs)	s.d. (μs)
TP10	B	TP8	B	6	2
Apex	R	TP4	R	70.6	1.6
TP-C	B	TP8	B	85	18
				156	17
				118	14

Table V 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 Fig. 5, 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 [22].

C. Summary and Implications

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

First, the measurements validate the TiF within an up-to-city scale geographic area. 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 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 V, 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, sub-ms 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 an IoT 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 ms or even tens of ms in a city-scale network. Although PTP can achieve sub-ms 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. And importantly, both NTP and PTP are susceptible to packet delay attacks, whereas the proposed system is not.

V. Accurate and Secure Clock Synchronization

Based on the key observations in §IV, this section presents the design of an accurate and secure clock synchronization approach for industrial IoT. Specifically, §V-A describes our threat model of the packet delay attack; §V-B overviews our approach; §V-C provides an analysis of our approach regarding its security against the packet delay attack.

A. 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.

As analyzed in [7], [8], 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

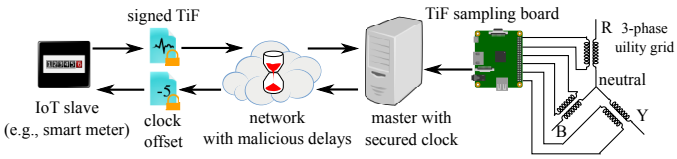


Figure 7. System architecture and clock synchronization.

to the NTP request and reply packets, respectively. 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 [31], 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.

B. 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 IoT nodes' grid phases autonomously.

1) Master-slave architecture: As observed in §IV-A2, if two nodes are on different phases, the TiF decoding will have errors. To address this issue, we propose to adopt a master-slave architecture as shown in Fig. 7, where 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 respectively to the three grid phases, where each channel is a set of the components shown in Fig. 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 an IoT 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, e.g., 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 IoT 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, an IoT 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, e.g., how to ensure security in the face of packet delay attack, will be presented in §V-C.

For an IoT spanning a large geographic area (e.g., a city), multiple masters can be deployed in a distributed anycast manner. An IoT slave may select a closest master for the best ZCI synchronism, for example.

2) Autonomous grid phase identification: The master has access to all the 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 IoT device 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 §IV-A1 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 the three grid phases. Unlike the offline evaluation in §IV-A1 where 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 §IV-A1 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 due 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 the three PDFs. We note that this autonomous identification is a one-time procedure that should be executed when an IoT device is added to the system or it changes power supply. For instance, a device can initiate this procedure when it is powered up.

Fig. 8 shows the PDFs generated by a master on the floor shown in Fig. 5, 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 Fig. 5, the identification results are all correct.

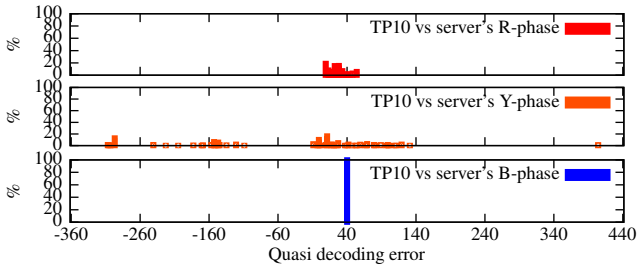


Figure 8. PDFs of quasi decoding errors for identifying the grid phase of TP10.

C. Security Analysis of Our Approach

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 §VI, 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 §VI, a synchronization session requires less than one second. Typical crystal oscillators found in MCUs and personal computers have drift rates of 30 to 50 ppm [19]. Thus, during the synchronization session, the slave's clock may have drifted for tens of μs. Even if we simply set the client's clock according to the returned offset, our approach can still achieve the sub-ms accuracy.

2) Security against packet delay attack: 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 $L \gg \frac{t}{20\,\text{ms}}$. For instance, in our performance evaluation in §VI, the measured t is around one second and we set $L = 1000$ 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 $\mathbf{y}$, i.e., $\mathbf{x} \triangleleft \mathbf{y}$, which is a prerequisite for decoding $\mathbf{x}$. The attacker may delay the transmission of $\mathbf{x}$ to violate the requirement $\mathbf{x} \triangleleft \mathbf{y}$. We propose a countermeasure stated

in the following proposition.

Proposition 1. *The slave discards the clock offset returned by the master if* $\frac{t}{20\,ms} > L$*, 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* $\mathbf{x} \triangleleft \mathbf{y}$ *will not affect the slave's clock;*
2) *if* $\frac{t}{20\,ms} \leq L$*, 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 $\mathbf{x} \triangleleft \mathbf{y}$ is $\frac{t_{s\to m}}{20\,\text{ms}} > L$. As $t > t_{s\to m}$, the attack must result in $\frac{t}{20\,\text{ms}} > L$. Vice versa, if $\frac{t}{20\,\text{ms}} \leq L$, the delay attack cannot invalidate $\mathbf{x} \triangleleft \mathbf{y}$ 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 $\mathbf{x} \triangleleft \mathbf{y}$, 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. 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 can be solved generally using conventional security measures.

VI. Performance Evaluation

We have implemented the synchronization approach presented in §V. We use the node shown in Fig. 4 as a slave. The RPi uses `OpenSSL` to sign the data to be transmitted using SHA256. For evaluation purpose, 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 respectively to TP4, TP6, and TP8 shown in Fig. 5, 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 the three nodes stream their measurements to a tower server with an Intel Xeon 3 GHz quad-core processor. The decoding algorithm in Eq. (1) can be parallelized since the computation for each iterator i in Eq. (1) is independent. Our implementation divides the computation equally among four threads to fully utilize the quad cores. We note that, to serve many IoT slaves, more computation resources can be allocated to maintain the timeliness of the decoding.

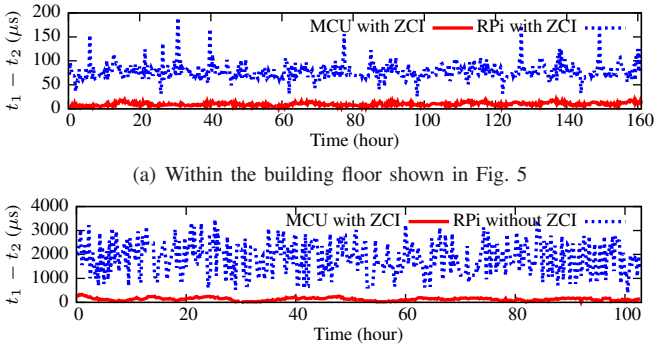
We conduct several sets of experiments as follows.

Time profiling. We measure the latency of the key steps of a clock synchronization session under two settings of the TiF length n to ensure correct decoding within a building floor and over a 10 km distance, respectively. Table VI 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). The total latency, estimated based on $v = 500$ kb/s, is less than one second.

Table VI
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	1400	126	6	10	0.2	*12.8*	*155*
20000	21000	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. 5

(b) Between TP-C and TP-A

Figure 9. Clock synchronization errors over days.

One-week evaluation within a building floor. We deploy a slave node at TP10, as shown in Fig. 5. It can identify its grid phase correctly using the method in §V-B2. 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 respectively by the slave and the master as the metric of synchronization error. On the slave, the TiFs are timestamped by both the MCU and the RPi. Fig. 9(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 an IoT device 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.

Four-day evaluation between TP-C and TP-A. We deploy a slave node at TP-C and synchronize 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 an IoT device without an interface for

handling low-level interrupts. Fig. 9(b) shows the results. The phase shifts measured by the MCU are generally below $200\ \mu$s, with mean and s.d. of $132\ \mu$s and $63\ \mu$s, 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.

VII. Conclusions and Future Work

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 an industrial IoT system connected to the same grid. Extensive empirical evaluations show that our approach achieves an average synchronization error of 0.1 ms between two network nodes 10 km apart, and $10\ \mu$s within the same floor of a building.

The experiments in this paper were conducted in a same city-scale power grid. For future work, it is interesting to conduct experiments in other power grids of different scales. For nodes not directly connected to the grid, we will explore the existence of time fingerprint in powerline electromagnetic radiation.

Acknowledgments

We thank the anonymous reviewers and shepherd for providing valuable feedback on this paper. This research was funded in part by the Energy Innovation Research Programme (EIRP, Award No. NRF2014EWTEIRP002-026), administered by the Energy Market Authority (EMA). The EIRP is a competitive grant call initiative driven by the Energy Innovation Programme Office, and funded by the National Research Foundation (NRF).

References

[1] World Economic Forum, "Industrial internet of things: Unleashing the potential of connected products and services," 2015, http://bit.ly/185DE8E.

[2] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, 2014.

[3] D. P. Shepard, T. E. Humphreys, and A. A. Fansler, "Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks," *Intl. J. Critical Infrastructure Protection*, vol. 5, no. 3, 2012.

[4] R. Tan, V. Badrinath Krishna, D. K. Yau, and Z. Kalbarczyk, "Impact of integrity attacks on real-time pricing in smart grids," in *CCS*, 2013.

[5] H. Weibel, "Tutorial: Precision clock synchronization protocol and synchronous Ethernet," in *IN2P3*, 2012, http://www.in2p3.fr/actions/formation/Numerique12/IEEE_1588_Tutorial_IN2P3_Handout.pdf.

[6] T. Mizrahi, "Security requirements of time protocols in packet switched networks," https://tools.ietf.org/html/rfc7384.

[7] M. Ullmann and M. Vogeler, "Delay attacks – implication on NTP and PTP time synchronization," in *Intl. Symp. Precision Clock Sync. for Meas., Control and Commun.*, 2009.

[8] T. Mizrahi, "A game theoretic analysis of delay attacks against time synchronization protocols," in *Intl. Symp. Precision Clock Sync. for Meas., Control and Commun.*, 2012.

[9] Argonne National Laboratory, "GPS is easy to spoof," http://www.ne.anl.gov/capabilities/vat/spoof.html.

[10] Q. Y. Lin Huang, "GPS spoofing – low-cost GPS simulator," in *DEFCON23*, 2015, https://bit.ly/22H2XTA.

[11] Kaspersky Lab, "Datesheet: Five myths of industrial control systems security," http://media.kaspersky.com/pdf/DataSheet_KESB_5Myths-ICSS_Eng_WEB.pdf.

[12] "Hackers infiltrated power grids in U.S., spain," http://on.recode.net/1m6E3Le.

[13] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *37th Conf. IEEE Ind. Electron. Society*, 2011.

[14] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[15] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys*, 2003.

[16] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *SenSys*, 2004.

[17] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *IPSN*, 2011.

[18] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "Exploiting FM radio data system for adaptive clock calibration in sensor networks," in *MobiSys*, 2011.

[19] T. Hao, R. Zhou, G. Xing, and M. Mutka, "WizSync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks," in *RTSS*, 2011.

[20] A. Rowe, V. Gupta, and R. R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *SenSys*, 2009.

[21] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li, and Y. Liu, "Flight: Clock calibration using fluorescent lighting," in *MobiCom*, 2012.

[22] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.

[23] North American Energy Standards Board, "Manual time error correction," https://www.naesb.org//pdf2/weq_bklet_011505_tec_mc.pdf.

[24] T. Perry, "Planned U.S. power system experiment means some clocks will speed up," IEEE Spectrum, 2011, http://bit.ly/1OXu2zZ.

[25] C. Grigoras, "Applications of enf criterion in forensic audio, video, computer and telecommunication analysis," *Forensic Science International*, vol. 167, no. 2, pp. 136–145, 2007.

[26] R. W. Sanders, "Digital audio authenticity using the electric network frequency," in *Intl. Conf. Audio Forensics-Theory and Practice*, 2008.

[27] R. Garg, A. L. Varna, and M. Wu, "Seeing ENF: natural time stamp for digital video via optical sensing and signal processing," in *ACM Multimedia*, 2011.

[28] S. Gupta, M. S. Reynolds, and S. N. Patel, "Electrisense: single-point sensing using EMI for electrical event detection and classification in the home," in *UbiComp*, 2010.

[29] TI, "Smart grid and energy solution guide," http://www.ti.com/lit/sl/slym071o/slym071o.pdf.

[30] "Adafruit ultimate GPS hat," https://www.adafruit.com/products/2324.

[31] http://publish.illinois.edu/cps-security/files/2016/08/sync.pdf.