EURASIP Journal on Advances
in Signal Processing

RESEARCH Open Access

Chain-structure time-delay reservoir computing for synchronizing chaotic signal and an application to secure communication

Leisheng Jin[1*], Zhuo Liu[1] and Lijie Li[2]

*Correspondence:
jinls@njupt.edu.cn

[1] College of Integrated Circuit Science and Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, Jiangsu, China
[2] College of Engineering, Swansea University, Swansea SA1 8EN, UK

Abstract
In this work, a chain-structure time-delay reservoir (CSTDR) computing, as a new kind of machine learning-based recurrent neural network, is proposed for synchronizing chaotic signals. Compared with the single time-delay reservoir, our proposed CSTDR computing shows excellent performance in synchronizing chaotic signal achieving an order of magnitude higher accuracy. Noise consideration and optimal parameter setting of the model are discussed. Taking the CSTDR computing as the core, a novel scheme of secure communication is further designed, in which the "smart" receiver is different from the traditional in that it can synchronize to the chaotic signal used for encryption in an adaptive manner. The scheme can solve the issues such as design constrains for identical dynamical systems and couplings between transmitter and receiver in conventional settings. To further manifest the practical significance of the scheme, the digital implementation using field-programmable gate array is conducted and tested experimentally with real-world examples including image and video transmission. The work sheds light on developing machine learning-based signal processing and communication applications.

Keywords: Reservoir computing, Machine learning, Synchronization, Chaos signal, FPGA, Secure communication

1 Introduction

Reservoir computing (RC), also known as echo state network (ESN), has been attracting widely interests [1–5]. Compared with the conventional recurrent neural network (RNN), the RC is characterized by its simplicity that only the output connection weights need to be trained by a linear regression algorithm. Apart from applications such as spoken digit recognition [6], noisy image recognition [7] and fault diagnosis [8], the RC is also performed as a powerful paradigm for multivariate time series prediction [9–12]. For example, Lu et al. [13] used RC to deduce the time-varying state of a dynamical system from a limited number of concurrent system state measurements. Rafayelyan et al. [14] proposed an optical scheme performing RC over very large networks for realizing spatiotemporal chaotic systems prediction. In order to cope with more complicated tasks relating to time series prediction, various improved RC models in terms of structure modification have been proposed. These include double-reservoir in parallel RC (DRESN) [15], broad-ESN [16], hierarchical delay-memory echo state network (HDESN) [17], integer echo state networks (intESN) [5], etc. Particularly, it is worth to mention that L. Appeltant et al. introduced an architecture of RC that only used a single dynamical node with delay feedback–time-delay reservoir (TDR), which can be seen a special class of RC characterized by resource-efficient implementation[18].

Very recently, researchers from the field of nonlinear dynamics used the RC based on conventional structure for realizing chaos synchronization [13, 19, 20]. Different from the tasks of pure time series predication, realizing stable and complete synchronization between the RC and practical dynamical systems is challenging. On the one hand, the sensitivity to initial values and parameters mismatch regarding to chaotic systems impose a negative impact on maintaining synchronization. On the other hand, general RC models have limitations in extracting the valuable evolution patterns from previous time series because of a short-term memory. RC models that can be used for achieving high quality of chaos synchronization need to be explored in depth covering aspects of network structure, training mechanism and hardware realization.

It is well known that the chaos synchronization plays a vital role for realizing secure communication [21–24]. In conventional scheme of chaos-based secure communication, the receiver is somehow demanded to equip a chaotic system (responser) identical to the one (driver) used for encryption in the transmitter, or a very complicated coupling between the driver and responser (in the case that driver and responser use different types of dynamical systems) needs to be well-designed [25, 26]. In addition, for the sake of security [27, 28], the chaotic system used for encryption in the transmitter needs to be changed, and if so the receiver has to do the same, which may undesirably involve the replacement and resetting of the hardware. Solving the above-mentioned issues is significant for developing advanced secure communications based on chaos synchronization.

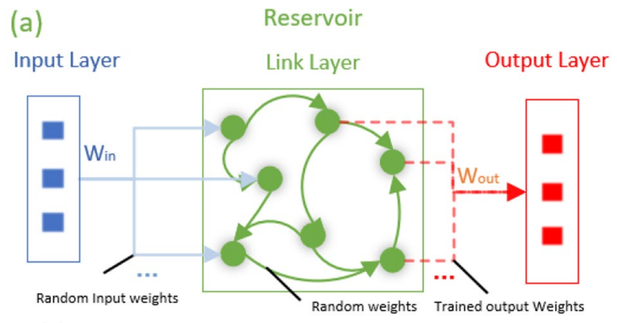
In this work, we focus on realizing chaos synchronization and propose a new kind of TDR-based machine learning paradigm, i.e., chain-structure time-delay reservoir (CSTDR) computing. The model can synchronize to chaotic systems with high accuracy in an adaptive manner. Based on the CSTDR, a novel secure communication scheme is designed, in which the receiver is endowed with the ability for decoding encrypted signal generated by any dynamical systems in the transmitter. There is no complicated coupling needed to be designed, while instead the chaos synchronization between transmitter and receiver can be achieved by training. Numerically, the image transmission encrypted by Lorenz chaotic system is simulated. Moreover, the electronic realization of the proposed CSTDR and secure communication scheme based on FPGA is implemented. The practical application of video transmitting employing this communication scheme is experimentally conducted, proving the effectiveness and feasibility of our scheme. The work sheds light on developing smart secure communications applications-based reservoir computing. Moreover, academically, the CSTDR would attract attention for digging intrinsic properties of state dynamics of layered TDRs, for example, how the dynamical properties of CSTDR are related to its memory capacity (MC) and how that can be used for optimization. The continuous study in academic would further make CSTDR more mature and facilitate to develop better bespoke systems for applications in industry.

The contributions of this paper are listed as follows:

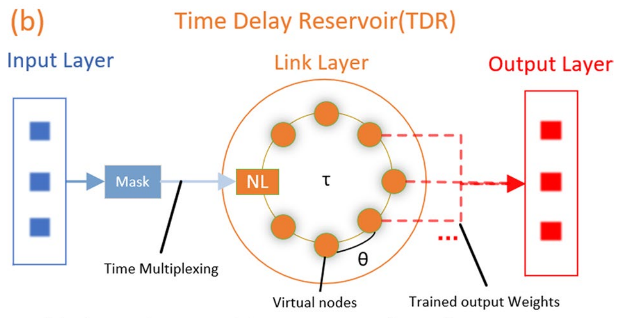
1. A chain-structure time-delay reservoir (CSTDR) computing is for the first time proposed. It can be used for achieving stable and complete synchronization with one-order higher accuracy than traditional TDR computing.
2. A novel scheme of secure communication based on CSTDR is designed, solving practical design constrains and shedding light on developing machine learning-based secure communication.
3. The electronic implementation of the CSTDR-based secure communication using FPGA is present. Taking the video transmission as example, the feasibility of the idea is verified, laying a foundation for practical application of CSTDR in the field of secure communication.

2 The regular time-delay reservoir (TDR) computing and its performance in chaos synchronization

The standard structure of a traditional RC is shown in Fig. 1a. It consists of an input layer, a hidden recurrent layer (reservoir) and an output layer. In the hidden recurrent layer, there are *N* sparsely connected neurons. In order to reduce the usually required large number of elements in traditional reservoir, a novel architecture that utilizes a nonlinear node with delayed feedback (TDR) for replacing the traditional reservoir is introduced [18]. The basic structure of TDR computing thus includes an input layer, an output layer and a link layer with *N* virtual nodes as shown in Fig. 1b. The virtual nodes are obtained by dividing the delay loop into *N* intervals and using time multiplexing. The weights of the output layer can be adjusted to make the TDR computing output desired signals. The training of the readout follows the standard procedure for RC.



(a) The traditional structure of reservoir computing



(b) A single time-delay reservoir (TDR) computing

Fig. 1 Basic structure of traditional RC and TDR computing

The nonlinear Mackey–Glass oscillator because of its easy implementation by electronic circuit is widely chosen for being the dynamical system to generate virtual nodes, which can be written as:

$$\dot{X}(t) = -X(t) + \eta \frac{X(t-\tau) + \gamma J}{1 + (X(t-\tau) + \gamma J)^p} \tag{1}$$

where the η is intensity of feedback and γ is input scale. The J is a temporal input stream derived by multiplying the input state at any input time t_0 with a Mask. The Mask is an N-dimensional vector of random numbers in the range [0,1]. The $X(t-\tau)$ denotes the state of the node at $t_0 - \tau$ time. The exponent p can be used to tune the non-linearity. In each τ period, there are $N\theta$ ($\theta = \tau/N$). During one τ time, the state of each node is updated once.

The Lorenz system was first proposed in 1963 by Lorenz [29], and it becomes one of the most famous nonlinear model for studding chaos. Here, we first study the realization of chaos synchronization between a TDR computing and Lorenz system. The model of Lorenz system is expressed as:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \gamma x - y - xz \\ \dot{z} &= xy - \beta z \end{aligned} \tag{2}$$

where $\sigma = 10$, $\gamma = 28$ and $\beta = 8/3$ and in such a parameter setting the system works in chaotic state. Equation 2 can be solved numerically to get a three-dimensional Lorenz chaotic sequence $L(t)$ ($L_x(t)$, $L_y(t)$ and $L_z(t)$). In the training phase, we take a part of the $L_x(t)$ and $L_z(t)$ as the input signal and target signal (the rest part of $L(t)$ is used for testing), respectively, which is different with the traditional training that the dimension of target signal is chosen as same as the input. Before training, the signals can be scaled to [0, 1] through the following procedure:

$$\begin{aligned} L_x &= \frac{L_x}{\max(L_x) - \min(L_x)} \times 0.8 + 0.5 \\ L_z &= \frac{L_z}{\max(L_z) - \min(L_z)} \times 0.8 + 0.5 \end{aligned} \tag{3}$$

Assuming that the length of $L_x(t)$ is M, the J $(N \times M)$ can be derived by multiplying the Mask $(N \times 1)$ and L_x $(1 \times M)$ expressed by:

$$J = Mask \times L_x \tag{4}$$

The J is then input into the Mackey–Glass oscillator for evolution, and the dynamical state x $(N \times M)$ can be stored. After discarding the initial 100 points, the ridge regression can be used to calculate w $(1 \times N)$, and thereby to make the output of the TDR as close as $L_z(1 \times M)$, in which the ridge regression [30] is given by :

$$w = L_z x^T \left(x x^T + \lambda II \right)^{-1} \tag{5}$$

where λ is a parameter with a size of 1×10^{-6} for avoiding over fitting, and II is an identity matrix. The calculated w is then needed to be deployed to the output layer of the

TDR. The training phase is done. Next, if we let the output value of the TDR feeding back into the input layer, the TDR computing can automatically generate values close to actual $L_z(t)$ but only for a few time steps. However, if we expect a long and stable output values that synchronize to actual $L_z(t)$, the signal L_x should be kept for inputting. During the synchronization, as the sequence L_x flowing into the TDR, the output value U_z is given by:

$$U_z = w \times x \tag{6}$$

To investigate the performance of a TDR computing for achieving chaos synchronization in Lorenz system, we study the cases ($L(t)$ as the dataset): the x time series as input signal, y and z as target (to synchronize with) signals; y as input signal, x and z as target (to synchronize with) signals; and z time series as input signal, x and y as target (to synchronize with) signals. The results are summarized in Table 1.

In Table 1, the minimum normalized mean square error (NMSE) [31] is used to evaluate the discrepancy of the achieved synchronization, which is defined as :

$$NMSE = \frac{\sum_t [G(t) - U_z(t)]^2}{\sum_t U_z(t)^2} \tag{7}$$

where the $G(t)$ represents the actual signal. It can be seen from Table 1 that the NMSE can reach a relatively smaller value when the x- and y-dimension is taken as input signal. For the case of z time series as input, the NMSE gets larger. The reason for this desynchrony is attributed to the inherent dynamics of the Lorenz system [32]. There is one positive conditional Lyapunov exponent for z driving, which leads the desynchrony in subsystem (x, y).

Similarly, the performance of the TDR for achieving synchronization in Rossler system is also studied. The Rossler system is seen as another famous nonlinear dynamical system for having chaos [33], which is given by:

$$\begin{aligned} \dot{x} &= -\omega y - z \\ \dot{y} &= \omega x + \alpha y \\ \dot{z} &= \beta + z(x - \gamma) \end{aligned} \tag{8}$$

where $\omega = 1, \alpha = 0.2, \beta = 0.2, \gamma = 5.7$ are adopted for generating chaotic time series. Again, we calculated the NMSE of this model considering different cases, and the results are summarized in Table 2. It is overall satisfactory with somehow the NMSE can reach as low as 10^{-3}, except for the z-dimension as input signal. Similar situation as the Lorenz system.

3 Method: the proposed chain-structure time-delay reservoir (CSTDR) computing and its performance in chaos synchronization

As shown in Fig. 2, a chain-structure time-delay reservoir (CSTDR) computing which has a few TDRs in series connection is proposed. There is an additional output layer designed for connecting the virtual nodes in all TDRs, which is different from the general DeepESN schemes. It should be noted that the TDR adopted in the proposed CSTDR could be more or less depending on practical need. For the task of synchronizing chaotic signals, we have

Table 1 Performance comparison of TDR and CSTDR computing for achieving chaos synchronization with Lorenz system

Driving	Sync	NMSE(TDR)	NMSE(CSTDR)
x	*y*	0.014435000	0.004093500
x	*z*	0.011834000	0.000560200
y	*x*	0.000199440	0.000019790
y	*z*	0.033652000	0.004502300
z	*x*	0.089724000	0.089737000
z	*y*	0.061575000	0.062309000

Table 2 Performance comparison of TDR and CSTDR computing for achieving chaos synchronization with Rossler system

Driving	Sync	NMSE(TDR)	NMSE(CSTDR)
x	*y*	0.01842000	0.00409600
x	*z*	0.00195150	0.00060731
y	*x*	0.00179330	0.00179210
y	*z*	0.00928880	0.00194980
z	*x*	0.05558000	0.09885000
z	*y*	0.07602900	0.07166000

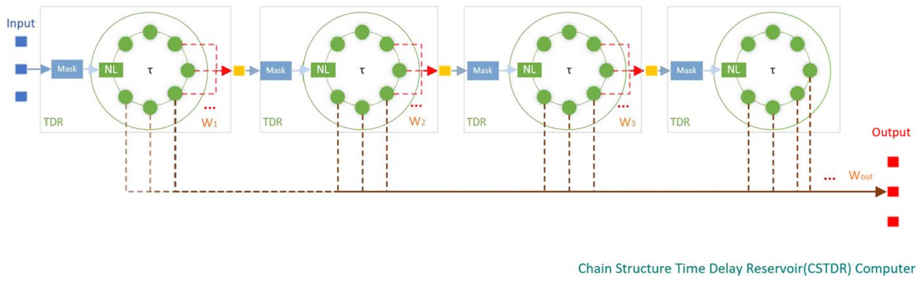


Fig. 2 The proposed chain-structure time-delay reservoir (CSTDR) computing

calculated the performance of CSTDR in the following discussion section. In this work, there are four TDRs with each having 12 virtual nodes generated by Mackey–Glass oscillator used in the CSTDR model.

Taking the CSTDR computing with four reservoirs (both reservoirs are generated by Mackey–Glass oscillator) as example, the training procedure includes the following steps:

① Train the first TDR as same as the standard method for traditional RC. The specific training process for the first TDR can be expressed as:

$$\begin{cases} J = Mask \times T_x, & (a) \\ \dot{X}_i(t) = -X_i(t) + \eta \frac{X_i(t-\tau)+\gamma J}{1+(X_i(t-\tau)+\gamma J)^p}, & (b) \\ W_i = T_z X_i^T (XX_i^T + \lambda II)^{-1}, & (c) \\ U_{x-i} = W_i \times X_i & (d) \end{cases} \tag{9}$$

where $i = 1$ represents the first TDR. The x-dimension of training set $T_x(t)$ is input to the first TDR and multiplied by the Mask, i.e., Eq. 9a. Dynamical states X_i are generated by Eq. 9b, in which the parameters are taken the same as described in last section. Equation 9c. is the standard linear regression algorithm for calculating the output layer weight: W_i (i=1). Last, the output of the first TDR: U_{x-1} can be derived by using Eq. 9d.

② Take the output of first TDR as input for the second TDR, and train it as same as the first TDR, outputting U_{x-2}.

③ Input U_{x-2} into the third TDR and train it as same as the first TDR, outputting U_{x-3}.

④ Input U_{x-3} into the fourth TDR. The training procedure for this one can be skipped.

⑤ Use the ridge regression to calculate the weights of the additional output layer W_{out} $(1 \times 4N)$ by combining dynamical states in all TDRs, i.e., X $(4N \times M)$ into consideration, that is:

$$\begin{aligned} W_{out} &= T_z X^T (XX^T + \lambda II)^{-1} \\ X &= [X_1; X_2; X_3; X_4] \end{aligned} \tag{10}$$

where λ is a parameter with a size of 1×10^{-6} used for avoiding over fitting, and II is an identity matrix.

After training, the output of the additional output layer is $G(t)$, which can be given by:

$$\begin{aligned} G(t) &= W_{out} \times X \\ X &= [X_1; X_2; X_3; X_4] \end{aligned} \tag{11}$$

4 Results and discussion

To verify the proposed model, we first test it with the Lorenz model. Similar to the case using a single TDR in last section, we take a part of $L_x(t)$ and $L_z(t)$ as the input and target signal, respectively. The rest of $L(t)$ is used for testing the performance of CDSTDR in realization of chaos synchronization. The preliminary simulation result is given in Fig. 3. It is seen that the output of our proposed CSTDR after training can completely synchronize to the actual signal of z-dimension.

To further illustrate the performance of our proposed model CSTDR in realization of chaos synchronization, we also investigate all the cases considered in last section where the traditional (single) TDR computing is used. The CSTDR computing here employs four TDRs with each having 12 virtual nodes generated by Mackey–Glass oscillator. The training and testing data set is generated by Lorenz system and Rossler system with unchanged parameters as same as in last section. The results are given in Tables 1 and 2. It can be seen that the NMSE can reach one order lower compared with the results derived by TDR in executable cases, while in some cases the NMSE is not improved. That is due to the inherent character of the specific dynamical systems as discussed above, which is nothing to do with methods adopted. All the results take the average of ten independent tests to avoid random errors.

In order to confirm the universality of our proposed CSTDR computing in realization of chaos synchronization, the Lorenz system and Rossler system with different parameters setting are also investigated.

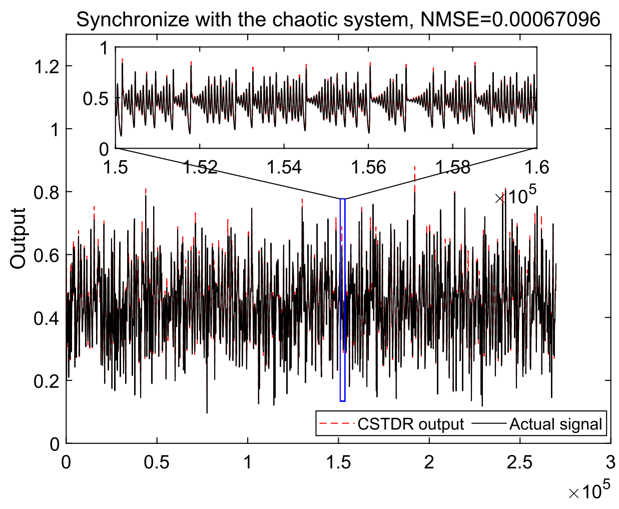


Fig. 3 Synchronization result of CSTDR computing

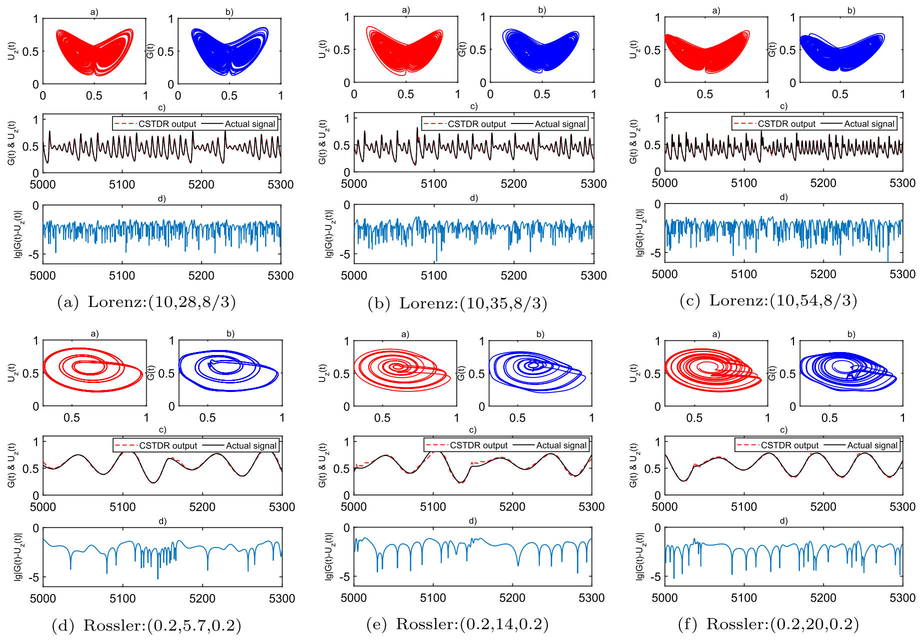


Fig. 4 Synchronization results between actual chaotic signal and output of CSTDR computing with Lorenz and Rossler system using different parameters. **a** and **b** in **a**, **b** and **c**: Chaotic attractors of the original Lorenz system and the trained CSTDR computing, respectively; **c** in **a**, **b** and **c**: Plot of time-varying output signal and actual chaotic signal of Lorenz system; **d** in **a**, **b** and **c**: Plot of time-varying differences between output signal of CSTDR and actual signal in Lorenz system. Captions of **a**, **b** and **c** with replacing the Lorenz with Rossler apply to **d**, **e** and **f**

Part of simulation results is shown in Fig. 4. The chaotic attractors ($x - z$) of Lorenz system and the generated by CSTDR computing (x-dimension signal as input) are, respectively, plotted, as shown in upper sub-figures of Fig. 4a–c. Meanwhile, the output signal of the CSTDR computing ($G(t)$) and actual signal (U_z) and their differences with varying time are also presented in the middle and lower sub-figures of Fig. 4a–c. All these results confirm that the proposed CSTDR computing can be used for achieving chaos synchronization. Likewise, the Rossler system with different settings is also under investigation, with results shown in Fig. 4d–f. Overall, the proposed CSTDR performs much better than the traditional TDR computing.

To investigate how the key parameters: the number of employed TDR in CSTDR and the number of virtual nodes in each TDR, affect the realization of chaos synchronization, we calculate minimum normalized mean square error (NMSE) between $G(t)$ and U_z by varying them in [1, 8] and [2, 50], respectively.

The result is shown in Fig. 5. It can be seen that as the number of reservoirs and the number of virtual node increase, the NMSE decreases gradually. Specifically, when the number of reservoirs switches between 1 and 2, the change of NMSE is particularly obvious. It seems to explain that single TDR computing cannot synchronize the chaotic system, while the CSTDR computing can. Meanwhile, along the dimension of number of virtual node, it is shown that when the TDR is in the range of [2, 8], there is a step change of NMSE when the number of virtual nodes changes from 2 to 4. It is also verified that when the number of reservoirs is larger than 8, the change of NMSE is becoming not obvious. Here in this work, we select the number of reservoirs to be 4 and the number of virtual nodes to be 12 as the optimal parameters.

In reality, the input signal is usually disturbed by noise. Here, we test the robustness of our proposed scheme under the impact of white noise. The white noise is added to the chaotic signals $L(t)$ (generated by Lorenz system), and the CSTDR is trained for fitting the L_z. The calculation result is shown in Fig. 6. It is seen that with the increase in signal-to-noise ratio (SNR), the NMSE between the actual signal of the Lorenz system and the output of CSTDR computing decreases continuously, and when the SNR exceeds 42dB, the NMSE stabilizes below $10^{-3.3}$. It can be concluded that the system has a certain level of anti-noise ability.

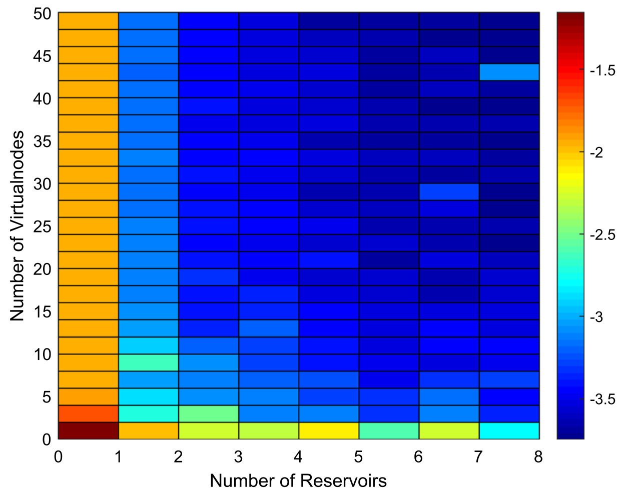


Fig. 5 Influence on synchronizing performance based on different number of time-delay reservoirs and virtual nodes in each TDR

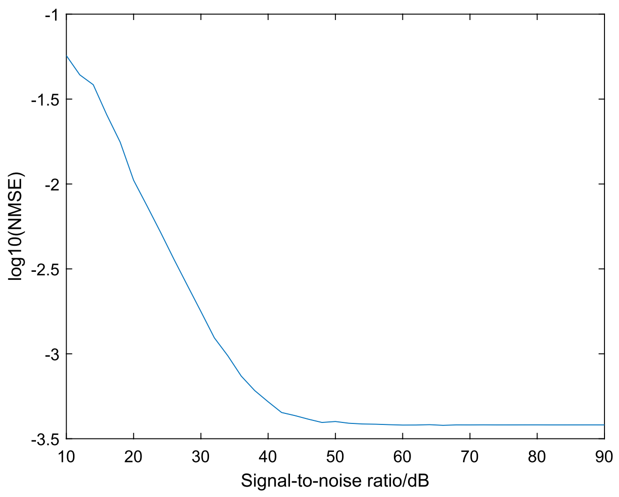


Fig. 6 Calculation of NMSE with varying SNR ratio

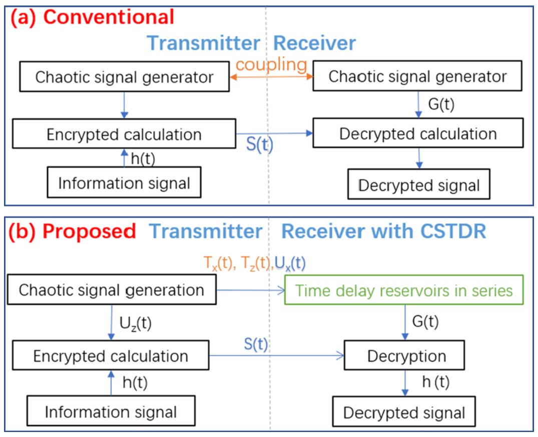


Fig. 7 **a** The conventional secure communication scheme in general. **b** The proposed scheme based on CSTDR

5 Secure communication scheme based on CSTDR computing

The proposed secure communication scheme is shown in Fig. 7, which mainly consists of two parts: transmitter and receiver. The overall idea is that the receiver in the proposed scheme can be trained, and then to synchronize the chaotic signal used for encryption in the transmitter based on RC [32]. With synchronization that we express in previous section, the hidden message can be decoded by subtraction.

In the transmitter, the signal $h(t)$ to be sent is encrypted by one vector of chaotic signal $L(t)$ (z-vector: $L_z(t)$ in our case) which is generated by a three-dimensional chaotic system. The $S(t)$ represents the signal after encryption. During the working process with the scheme, the chaotic signal $L(t)$ generated by a chaotic system in the transmitter needs to be divided into two time continuous parts, i.e., $T(t)$, $U(t)$. The z-vector of $U(t)$ ($U_z(t)$) is used to encrypt the signal $h(t)$. The $T(t)$ is deliberately left for training the receiver.

In the training phase (indicated by the yellow letter), the CSTDR computing of receiver learns the received training set $T(t)$. The $T_x(t)$ is input into the CSTDR computing, and training follows the method we proposed in previous section with target at $T_z(t)$. After training, it is believed that the receiver has learned the chaotic signal used for encryption in the transmitter.

In the synchronizing phase (indicated by the blue letter), the transmitter only needs to send the scalar signal $U_x(t)$, the x-vector of signal $U(t)$, to the receiver to drive the CSTDR, and the receiver can synchronize and generate a desired signal: $G(t)$, which is supposed to be synchronized with z-vector of $U(t)$, i.e., $U_z(t)$. Finally, we can decrypt the encrypted information $S(t)$ with signal $G(t)$ to receive the signal $h(t)$.

To compare with the conventional scheme of secure communication shown in Fig. 7, our proposed method is designed based on a totally different idea, which has the following merits. First, the receiver in our scheme does not need equip a fixed dynamical system which is normally supposed to be as same type as the one in the transmitter. It can synchronize to the chaotic signal generated by any dynamical systems for encrypting in an adaptive manner. A complicated coupling between transmitter and receiver such that in conventional scheme is saved. Second, our proposed method can better adapt to the changes occurring in the chaotic system employed in transmitter. Normally, in traditional technique once the encrypting chaotic system is changed, the design of coupling and dynamical system in receiver has to be changed accordingly, which could involve hardware replacement and security reduced, while the proposed here can update itself in-time just by learning a limited length of chaotic signal. Therefore, the security is enhanced and the cost is reduced. In terms of cost, though the CSTDR looks more complicated than a single TDR, its practical realization actually can use only one TDR. In the following hardware implementation of CSTDR-based secure communication, only one TDR is made by FPGA, the CSTDR computing can be conducted by multiplexing the same one. The cost increased is more about the algorithm part.

6 Numerical simulation and hardware implementation of the CSTDR-based communication scheme

6.1 Numerical results: image transmission using CSTDR-based communication scheme

In order to verify the effectiveness of the proposed scheme, the secure communication of image is numerically simulated. Taking the image shown in Fig. 8a whose standard size is $300 \times 300 \times 3$ as an example, the image is firstly arranged into an one-dimensional vector signal $h(t)$. The signal $h(t)$, as an useful signal waiting to be transmitted, needs to be encrypted. It is modulated to encrypted signal. The encryption equation can be adopted as follows:

$$S(t) = K_1 U_z + K_2 h \tag{12}$$

where the $U_z(t)$ is one vector the chaotic signal $U(t)$ and $S(t)$ is one-dimensional vector after encryption. In order to increase the security performance, the absolute value of K_1/K_2 is taken as large as possible. $K_1 = 1.3$ and $K_2 = -0.3$ are chosen in the study.

According to our scheme proposed above, the $G(t)$ can be derived based on the transmitted $S(t)$. The $h(t)$ can be demodulated by using the following decryption equation:

$$h(t) = (SK_1 - G(t))/K_2 \tag{13}$$

Finally, the $h(t)$ can be transformed into original image through the inverse process of sorting the data positions, as shown in Fig. 8c.

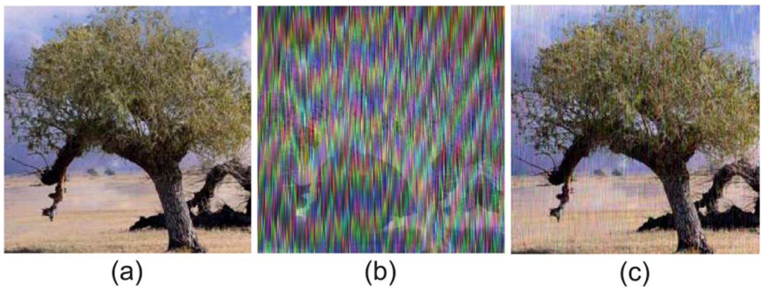


Fig. 8 **a** Picture waiting to be sent, **b** picture encrypted, **c** picture after decryption

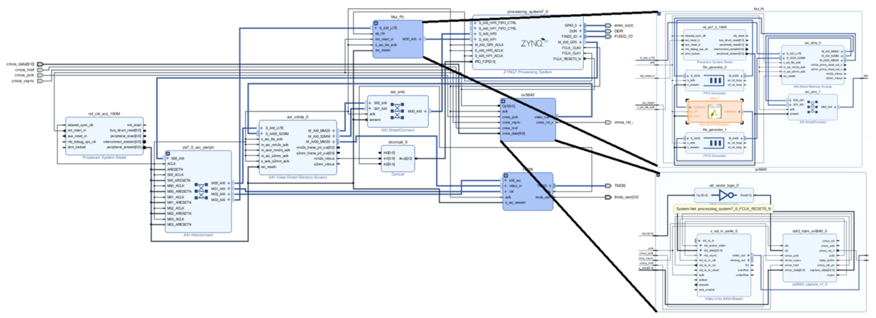
In this example, the Lorenz chaotic system is used, and we select x-vector of $U(t)$ as the driving signal and z-vector of $U(t)$ as the encryption signal. Specifically, the first 6100 data points are taken for training set $L(t)$. The rest of the data are used as encrypted data set $U(t)$. In receiver, four TDRs with additional output layer are used to synchronize the chaotic system. And in each TDR, 12 virtual nodes are generated.

6.2 Experimental result: practical video transmission using CSTDR-based communication scheme

According to the proposed scheme shown in Fig. 7b, the Xilinx xc7z020 Soc, as a popular chip which combines the advantage of the ARM core and FPGA, is used for hardware implementation.

The design diagram of transmitter and receiver is presented in Fig. 9a, b, respectively. In Fig. 9a, it consists of HDMI module, OV5640 module and Mul module. They are designed for video output, camera information processing and accelerating. In Fig. 9b, there are modules of CSTDR, HDMI and Mul, and the core of the receiver part is the CSTDR module. Limited by the on-board resources of Xilinx xc7z020 Soc, we use the High-Level Synthesis [34] to build a single TDR model and use it for time-division multiplexing to form the CSTDR computing. The accuracy of calculations is double-precision floating-point.

The working process diagram of the CSTDR-based communication scheme is shown in Fig. 10a. It is seen that both the transmitter and receiver start from flash, and first all the modules need to be initialized. Then, the transmitter generates a chaotic sequences as training three-dimensional data set $T(t)$, in which the T_x and T_z need to be sent to the receiver through Ethernet. After receiving, the receiver starts to train the CSTDR computing based on the training algorithm given in Sect. 3. Once the training is completed, all the weights affiliating to output layer in each TDR and additional output layer of CSTDR can be deployed. The rest working procedure including synchronization and decryption is as same as described above.


(a) The design of transmitter part

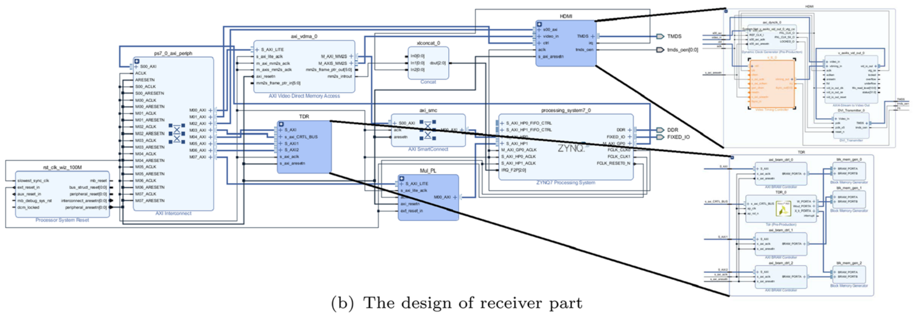

(b) The design of receiver part

Fig. 9 The electronic design of the CSTDR-based secure communication scheme using ZYNQ 7020 Soc

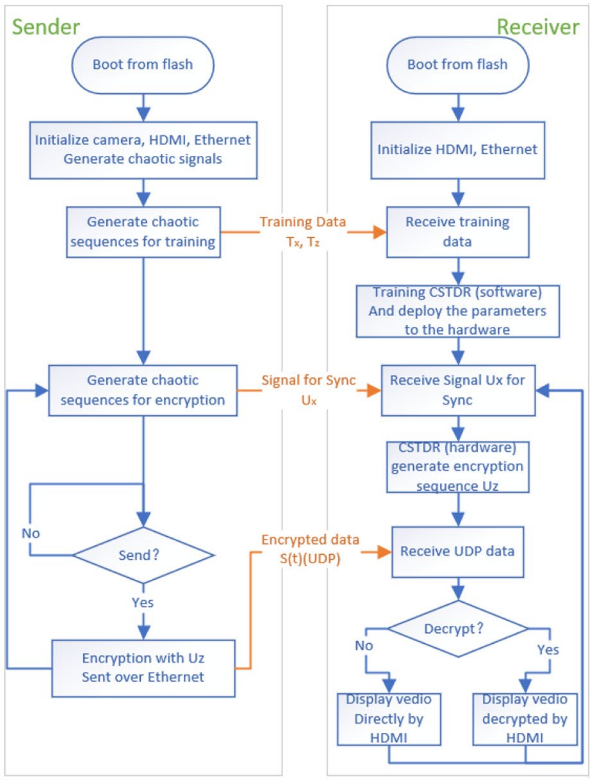
The flowchart of data stream in the whole CSTDR-based communication process is shown in Fig. 10b.

Based on the FPGA design in Fig. 9a, b, the hardware realization is accomplished and given in Fig. 11a. An example of video transmission in the real world is experimentally conducted. The results are presented in Fig. 11b, c. In Fig. 11b, it can be seen that the video is encrypted. The receiver can decrypt the video with a high quality as shown in Fig. 11c.

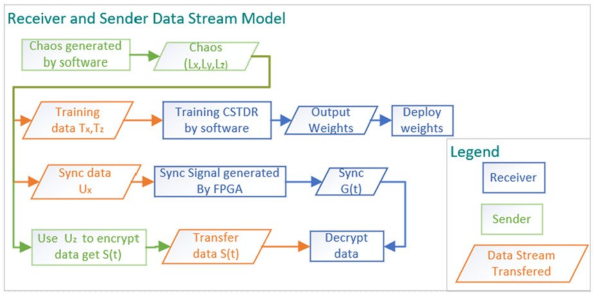
It is worth to mention that for uncompressed video transmission based on our proposed scheme the reusing encryption data set should be taken in consideration. The resolution of image in video is normally about 640×480, i.e., the size of a frame is 921,600 bytes. Assuming 30 frames per second are transmitting, it needs at least an encryption data sequence of length 27,648,000 per second. Therefore, it is unrealistic to transfer uncompressed video without processing. One solution to this issue is to adopt reusing encryption data. Utilizing the method, the CSTDR-based communication can reach the maximum speed of 13.9 Mb/s.

7 Conclusion

In this work, a chain-structure time-delay reservoir (CSTDR) computing is proposed to realize the chaos synchronization in nonlinear dynamical systems. An order higher accuracy of complete synchronization than traditional TDR can be achieved. Based on the CSTDR computing, a novel scheme of secure communication is further put forward. Numerical simulation and digital implementation are both conducted for proving


Sender
Receiver
Boot from flash
Initialize camera, HDMI, Ethernet Generate chaotic signals
Generate chaotic sequences for training
Training Data Tx, Tz
Initialize HDMI, Ethernet
Receive training data
Training CSTDR (software) And deploy the parameters to the hardware
Generate chaotic sequences for encryption
Signal for Sync Ux
Receive Signal Ux for Sync
CSTDR (hardware) generate encryption sequence Uz
No
Send?
Yes
Encrypted data S(t)(UDP)
Receive UDP data
Decrypt?
No
Yes
Encryption with Uz Sent over Ethernet
Display vedio Directly by HDMI
Display vedio decrypted by HDMI

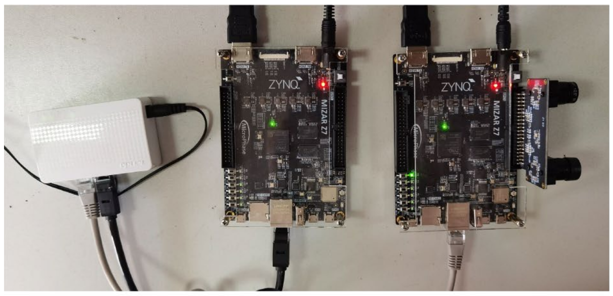
(a) Flow chart of the CSTDR based secure communication realized by ZYNQ 7020 Soc.



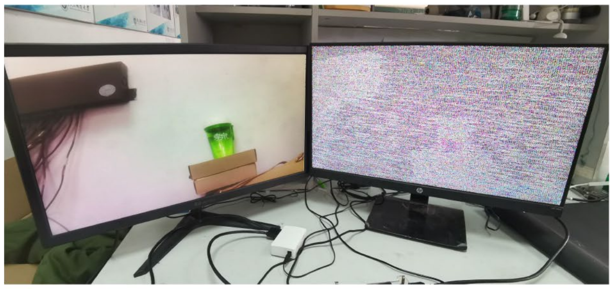
(b) Data Stream Model of the CSTDR based secure communication.

Fig. 10 Detailed working process of the CSTDR-based secure communication scheme

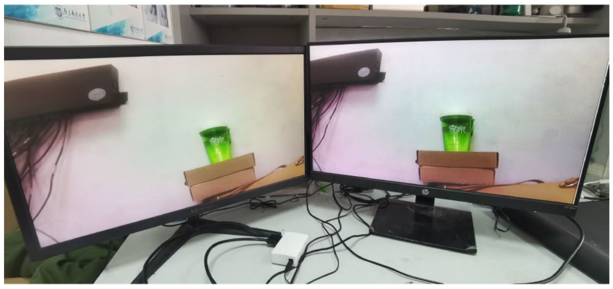
the feasibility of the scheme. By using ZYNQ 7020 Soc, the communication scheme is experimentally realized by hardware, which provides a more practical model on how to utilize our paradigm in reality. The work not only provides a deep RC models based on TDR but also extends the machine learning approach into practical applications, shedding light on the exploration of the intrinsic dynamics of deep TDRs and bespoke systems.



(a) The electronic hardware realization



(b) The video transmitting with encryption



(c) The video transmitting after decryption

Fig. 11 Experimental results based on the CSTDR-based secure communication scheme

Abbreviations

CSTDR	Chain-structure time-delay reservoir
TDR	Time-delay reservoir
FPGA	Field-programmable gate array
RC	Reservoir computing
RNN	Recurrent neural network
DRESN	Double-reservoir in parallel RC
HDESN	Hierarchical delay-memory echo state network
intESN	Integer echo state networks
NMSE	Normalized mean square error
Broad-ESN	Broad echo state network

Acknowledgements
Not applicable.

Author contributions
L. J and Z. L conceived the idea, prepared the manuscript, conducted numerical and experimental validations together. L. J and Z. L contributed equally to this manuscript. All authors took part in discussing the results. All authors read and approved the final manuscript.

Funding
The authors acknowledge the support from China Postdoctoral Science Foundation (No. 2019T120447).

Declarations

Ethics approval and consent to participate
We declare that the studies in this work do not involve any human participants, human data, human tissue and animal.

Consent for publication
We declare that the manuscript does not contain any individual person's data in any form (including individual details, images or videos).

Competing interests
The authors declare no competing interests.

Received: 24 February 2022 Accepted: 28 June 2022
Published online: 28 July 2022

References
1. M. Freiberger, S. Sackesyn, C. Ma, A. Katumba, P. Bienstman, J. Dambre, Improving time series recognition and prediction with networks and ensembles of passive photonic reservoirs. IEEE J. Sel. Top. Quantum Electron. **26**(1), 1–11 (2020)
2. L. Gao, X. Deng, W. Yang, Smart city infrastructure protection: real-time threat detection employing online reservoir computing architecture. Neural Comput. Appl. **34**, 833–842 (2021)
3. X. Zhu, X. Qiang, M. Tang, H. Li, F. Liu, A hybrid machine learning and computing model for forecasting displacement of multifactor-induced landslides. Neural Comput. Appl. **30**, 3825–3835 (2018)
4. D. Patel, D. Canaday, M. Girvan, A. Pomerance, E. Ott, Using machine learning to predict statistical properties of non-stationary dynamical processes: system climate,regime transitions, and the effect of stochasticity. Chaos Interdiscip. J. Nonlinear Sci. **31**(3), 033149 (2021)
5. D. Kleyko, E.P. Frady, M. Kheffache, E. Osipov, Integer echo state networks: efficient reservoir computing for digital hardware. IEEE Trans. Neural Netw. Learn. Syst. 1–14 (2020)
6. Q. An, K. Bai, L. Liu, F. Shen, Y. Yi, A unified information perceptron using deep reservoir computing. Comput. Electr. Eng. **85**, 106705 (2020)
7. A. Jalalvand, K. Demuynck, W. De Neve, J.-P. Martens, On the application of reservoir computing networks for noisy image recognition. Neurocomputing **277**, 237–248 (2018)
8. J. Long, S. Zhang, C. Li, Evolving deep echo state networks for intelligent fault diagnosis. IEEE Trans. Ind. Inf. **16**(7), 4928–4937 (2020)
9. N.A.K. Doan, W. Polifke, L. Magri, Physics-informed echo state networks. J. Comput. Sci. **47**, 101237 (2020)
10. Y.-C. Bo, P. Wang, X. Zhang, An asynchronously deep reservoir computing for predicting chaotic time series. Appl. Soft Comput. **95**, 106530 (2020)
11. H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
12. M.C. Soriano, S. Ortín, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, G. van der Sande, Delay-based reservoir computing: noise effects in a combined analog and digital implementation. IEEE Trans. Neural Netw. Learn. Syst. **26**(2), 388–393 (2015)
13. Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, E. Ott, Reservoir observers: model-free inference of unmeasured variables in chaotic systems. Chaos Interdiscip. J. Nonlinear Sci. **27**(4), 041102 (2017)
14. M. Rafayelyan, J. Dong, Y. Tan, F. Krzakala, S. Gigan, Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction. Phys. Rev. X **10**, 041037 (2020)
15. S. Zhong, X. Xie, L. Lin, F. Wang, Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction. Neurocomputing **238**, 191–204 (2017)
16. X. Yao, Z. Wang, Broad echo state network for multivariate time series prediction. J. Frankl. Inst. **356**(9), 4888–4906 (2019)
17. X. Na, W. Ren, X. Xinghan, Hierarchical delay-memory echo state network: a model designed for multi-step chaotic time series prediction. Eng. Appl. Artif. Intell. **102**, 104229 (2021)
18. L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)
19. T. Weng, J. Song, H. Yang, G. Changgui, J. Zhang, M. Small, Synchronization of reservoir computers with applications to communications. Physica A **544**, 123453 (2020)
20. H. Fan, L.-W. Kong, Y.-C. Lai, X. Wang, Anticipating synchronization with machine learning. Phys. Rev. Res. **3**(2), 023237 (2021)
21. A.J. Lawrance, T. Papamarkou, A. Uchida, Synchronized laser chaos communication: statistical investigation of an experimental system. IEEE J. Quantum Electron. **53**(2), 1–10 (2017)
22. Z. He, K. Li, L. Yang, Y. Shi, A robust digital secure communication scheme based on sporadic coupling chaos synchronization. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **47**(3), 397–403 (2000)
23. A. Tian, F. Chengbiao, H.-T. Yau, S. Xiao-Yi, H. Xiong, A new methodology of soil salinization degree classification by probability neural network model based on centroid of fractional lorenz chaos self-synchronization error dynamics. IEEE Trans. Geosci. Remote Sens. **58**(2), 799–810 (2020)
24. L. Jin, Y. Zhang, L. Li, One-to-many chaotic synchronization with application in wireless sensor network. IEEE Commun. Lett. **17**(9), 1782–1785 (2013)
25. S. Chen, Yu. Simin, J. Lü, G. Chen, J. He, Design and FPGA-based realization of a chaotic secure video communication system. IEEE Trans. Circuits Syst. Video Technol. **28**(9), 2359–2371 (2018)

26. I. Grosu, E. Padmanaban, P.K. Roy, S.K. Dana, Designing coupling for synchronization and amplification of chaos. Phys. Rev. Lett. **100**, 234102 (2008)
27. N. Jiang, A. Zhao, C. Xue, J. Tang, K. Qiu, Physical secure optical communication based on private chaotic spectral phase encryption/decryption. Opt. Lett. **44**(7), 1536–1539 (2019)
28. L. Zhou, F. Tan, X. Li, L. Zhou, A fixed-time synchronization-based secure communication scheme for two-layer hybrid coupled networks. Neurocomputing **433**, 131–141 (2021)
29. E. Lorenz, Deterministic nonperiodic flow. J. Atmos. Sci. **20**, 130–141 (1963)
30. R. Zhang, X. Li, W. Tong, Y. Zhao, Data clustering via uncorrelated ridge regression. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 450–456 (2021)
31. P. Händel, Understanding normalized mean squared error in power amplifier linearization. IEEE Microw. Wirel. Compon. Lett. **28**(11), 1047–1049 (2018)
32. T. Weng, H. Yang, G. Changgui, J. Zhang, M. Small, Synchronization of chaotic systems and their machine-learning models. Phys. Rev. E **99**, 042203 (2019)
33. O.E. Rössler, An equation for continuous chaos. Phys. Lett. A **57**(5), 397–398 (1976)
34. Xilinx. Vivado high-level synthesis. https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html. Accessed 20 July 2020

Publisher's Note