WB-GWS: An IoT-Oriented Lightweight Gateway System Based on White-Box Cryptography

Jinfu Hao[1], Yongbao Wang[2], Weijie Deng[1], Nanjiang Xie[1], and Zheng Gong[1(✉)]

[1] School of Computer Science, South China Normal University, Guangzhou, China
cis.gong@gmail.com
[2] AISINO CO.LTD, Beijing, China

Abstract. The Internet of Things (IoTs) has been widely applied for convenient data gathering. Therefore, the security and privacy issues related to IoT applications attract more and more attentions. Secure data transmission plays pivotal role of the security of IoT applications. However, there are still many IoT devices do not support cryptographic functions due to their constrained resources. Moreover, cryptographic key storage also becomes a practical security context problem in resource-constrained devices. In this paper, an IoT-oriented lightweight gateway system, which is named WB-GWS, is proposed by using white-box cryptography. WB-GWS is designed for the secure data transmission in typical IoT applications, and reduces the risk of key leakage in the white-box security context. The performance of WB-GWS is analyzed amongst different platforms. The experiment results support that WB-GWS can meet the security requirements of IoT applications with low-rate data transmission.

Keywords: Iot security · Secure data transmission · Virtual tunneling · White-box cryptography

1 Introduction

With the rapid development of IoT applications, people are aware of the vital security and privacy issues of data gathering and transmission [1,19,27,34]. Atzori *et al.* described the IoT system as a three-layer architectural model which consists of a perception layer, a network layer, and a layer of services [3]. The security architecture of the IoT can also be divided into these three layers. Due to its constrained resources, there are still many IoT systems that cannot support secure data transmission. The transport layer security can use public key cryptography and symmetric cryptography to ensure its security. Researchers use SSL/TLS (combination of public key cryptography and symmetric cryptography) in combination with HTTP or MQTT to ensure the secure transmission of

J. Lin and Q. Tang (Eds.): AC3 2022, LNICST 448, pp. 29–45, 2022.
https://doi.org/10.1007/978-3-031-17081-2_3

data [22]. Mazen *et al.* developed a Hybrid End-to-End VPN security approach by combining the IPSec/IPv6 and OpenSSL security approaches for secure data transfer for the IoT [18]. However, many IoT devices with constrained resources are not suitable for using public key cryptography. On the other hand , it is still challenging that public key cryptography to work with IoT devices [17].

IoT devices are commonly implemented in open access environment, or the environment cannot to be trusted [17]. Therefore, it could be feasible for hacking the cryptographic key from the devices. For mitigating key exposure and code-lifting problems in untrusted environments, many white-box cryptography (WBC) schemes have been proposed [12,24,25,33]. In 2019, Saha *et al.* used WBC to increase the security of IoT systems [31]. To the best of our knowledge, there is little research on WBC in secure data transmission of gateway based IoT system.

Our Contribution. In this paper, the design and implementation of an IoT-oriented lightweight gateway system based on white-box cryptography (WB-GWS) is proposed. WB-GWS is a secure data transmission system designed for gateway based IoT system. Firstly, a secure data transfer capabilities is provided for IoT systems that do not support secure data transformation. Secondly, the problem of insecure key storage on the IoT client is solved by using white-box cryptographic services. Finally, the performance of WB-GWS is analyzed amongst different platforms for convincing the practicality of the WB-GWS in the IoT environment.

Organization. The remainder of this paper is organized as follows. In Sect. 2, notions and notation that related to WB-GWS are described. Section 3 presents the design of WB-GWS and its protocols. Section 4 and 5 describe how to manage nodes in WB-GWS. The security analysis and performance results are given in Sect. 6 and 7, respectively. Section 8 concludes this paper.

2 Preliminaries

In this section, first the IoT protocols that are related to WB-GWS are briefly introduced. And then white-box cryptography is recalled. Table 1 summarizes the symbols and acronyms used in this paper.

IoT Transport Layer Protocol. IoT communication protocol can use the traditional TCP/IP protocol. MQTT, CoAP [4,30], and other protocols designed for the IoT can also be used [22]. A security layer is added to these protocols to secure the transmission of data. The Transport Layer Security (TLS) protocol and its datagram-oriented variant Datagram TLS (DTLS) protocol are the actual protocols for communication security in IoT applications [2,26]. However, the current authentication approaches of TLS are confronted with the heavy overhead and security issues in the resource-constrained IoT scenario. Although the standardized Internet security protocols have many advantages, typical deployment scenarios of IoT limit the feasibility of TLS because of highly constrained devices in low-power and lossy networks [26,29]. The standard TLS protocol not

Table 1. List of symbols and acronyms.

$WBEnc\{\cdot\}$	White-box encryption
$WBDec\{\cdot\}$	White-box decryption
$LUT\_Gen\{\cdot\}$	Generate a LUT
$Token\_Gen\{\cdot\}$	Generate download LUT credential
$Virtual\_IP\_Gen\{\cdot\}$	Generate virtual IP address
$Tag\_Gen\{\cdot\}$	Generate GCM message authentication code
$Auth\{Tag\}$	Data integrity check
$Check\{v\}$	Check that the current LUT of v is up to date
$Verify\{v\}$	Verify the validity of the IoTS-GW v
$Free\{r\}$	Release resource r
$Add\_To\_Reg\{v\}$	Add a GN v to the registry
$Remove\_From\_Reg\{v\}$	Remove GN v from the registry
$Send\_To\_AS\{m\}$	Send message m to the application server
$Shutdown\{s\}$	Stop the service s
$A \rightarrow B : [type] \| m$	A sends the following type of message m to B REGISTER: a request for registration REQUEST: a common request RESPONSE: return information DOWNLOAD: download the LUT
$\|$	Concatenation operator
$IoTD_i$	The IoT device i
$IoTS-GW$	The IoT-side gateway
$SS-GW$	The server-side gateway
GN	The gateway node IoTS-GW or SS-GW
V_*	Set of GN
$WBCS$	The white-box cryptography service
AS	The application service
IV	The initialization vector
N_v	Nonce sent by v
$Flag$	The flag of IoTS-GW or SS-GW
LUT	The white-box look-up table
$token$	The credential used to download the LUT
$Virtual\_IP$	The virtual IP address
Ver	Version of the LUT
Tag	GCM certification mark
Ret	The result of validating or establishing a connection
$PMsg$	The plaintext message
$CMsg$	The ciphertext message
ID_v	The identity of v
$rand$	The random number
$release\ the\ connection$	The release connection message
$acknowledgement\ character$	The confirmation message
APP_A	Common application of host A
VPN_A	Application layer VPN implementation of host A
$Stack_A$	Kernel network protocol stack of host A
$eth0_A$	Physical network interface card (NIC) of host A
$tun0_A$	Virtual NIC of host A

will be implemented in WB-GWS. Instead, WBC is used at the transport layer for the confidentiality and integrity of the data.

White-Box Cryptography. WBC was first proposed by Chow *et al.* in SAC 2002 [12], which is also called CEJO framework hereafter. The core idea of CEJO framework is to transform the round functions into a series of look-up tables (LUTs). And secret invertible encodings are used to protect the intermediate values. Let T denote a LUT, f and g be random bijective mappings. Then the encoded LUT T' is defined as $T' = g \circ T \circ f^{-1}$, where f^{-1} and g are called the input and output encodings, respectively. To maintain the functionality of AES [14], the input and output encodings of consecutive rounds should be constructed as pairwise invertible mappings. Let an encoded LUT R' be defined as $R' = h \circ R \circ g^{-1}$ such that a networked encoding can be depicted as $R' \circ T' = (h \circ R \circ g^{-1}) \circ (g \circ T \circ f^{-1}) = h \circ (R \circ T) \circ f^{-1}$.

The intermediate state of AES can be represented by a byte array such that the index is ranked from 0 to 15. For each round of AES, AddRoundKey and SubBytes can be combined into LUT operations called T-box. For a byte x input, it is computed as follows.

$$T_i^r(x) = S(x \bigoplus \hat{k}_{r-1}[i]), \qquad i = 0 \cdots 15, r = 1 \cdots 9,$$
$$T_i^{10}(x) = S(x \bigoplus \hat{k}_9[i]) \bigoplus \hat{k}_{10}[i], \qquad i = 0 \cdots 15,$$

where S denotes the Sbox and $\hat{k}_r$ represents the result of applying ShiftRows to the byte array of round key. For more details on the implementation of WBC, please refer to Muir's tutorial on white-box AES [24].

Although there are many attacks against WBC, such as DCA [11,21], DFA [15,20], and adaptive side-channel analysis [32], researchers are also designing new WBC proposals [7,9,10] to resist these attacks. Without losing of generality, we note that it is assumed that WB-GWS uses a secure WBC implementation.

Virtual Tunnel Technology. Tunneling is a virtual link between networks that uses the Internet infrastructure to transmit data [35]. The tunneling technology can be used to transmit protocol data unit (PDU) of different protocols. And it can also facilitate the transformation of data in the channel, such as encryption and integrity check. Tunnel protocols include L2TP, IPsec, and SSL VPN [6,35] *etc.* The reason why mentioned above tunneling protocols are not used is that it is difficult to apply WBC to the modification of those protocols. In this paper, Linux virtual network technology TUN/TAP is used to implement the tunnel protocol. Whilst a white-box block cipher is used to encrypt and decrypt data in tunnels.

Without tunneling, the process of sending data from host A to host B is illustrated by the dotted arrow in Fig. 1.

1. APP_A constructs the packet and sends it to kernel $Stack_A$.
2. The packet is sent to $eth0_A$ after $Stack_A$ adds the TCP header, IP header and other processing.

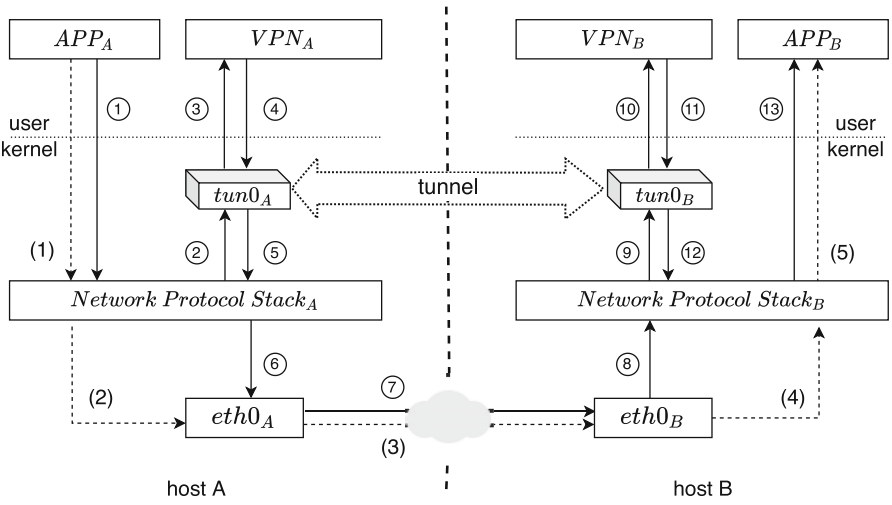

Fig. 1. The principle of virtual tunnel technology.

3. $eth0_A$ sends packet to $eth0_B$ over the Internet.
4. $eth0_B$ sends the received packet to $Stack_B$.
5. $Stack_B$ processes the packet and sends it to APP_B.

When data is transmitted over a virtual tunnel, the data flow of host A and host B is shown by the solid arrows in Fig. 1.

1. The APP_A constructs the packet and sends it to kernel $Stack_A$.
2. The data processed by $Stack_A$ is not directly sent to $eth0_A$, but forwarded to $tun0_A$.
3. The VPN_A listens to $tun0_A$ and reads the packets received by $tun0_A$.
4. The VPN_A can reprocess packet, such as data encryption and decryption, and integrity check. VPN_A then writes the processed data to $tun0_A$.
5. The $tun0_A$ receives the data and forwards it to $Stack_A$.
6. The packet is processed by $Stack_A$ and sent to $eth0_A$.
7. The $eth0_A$ sends packet to $eth0_B$ over the Internet.
8. The $eth0_B$ sends the received packet to $Stack_B$.
9. The $Stack_B$ writes data to $tun0_B$ after removing IP and TCP headers.
10. The VPN_B listens to $tun0_B$ and reads the packets received by $tun0_B$.
11. The VPN_B can reprocess packet, such as data encryption and decryption, and integrity check. VPN_B then writes the processed data to $tun0_B$.
12. The $tun0_B$ receives the packet and forwards it to $Stack_B$.
13. $Stack_B$ processes the packet and sends it to APP_B.

Although one can directly encrypt and decrypt data and verify the integrity in the application layer, we use a virtual tunnel that can encrypt and decrypt the data sent between APP_A and APP_B and verify the integrity without changing APP_A and APP_B. Therefore, data can be transparently and securely transmitted between A and B.

3 The Construction of WB-GWS

In this section, the framework of WB-GWS is illustrated and then the protocols of WB-GWS are demonstrated.

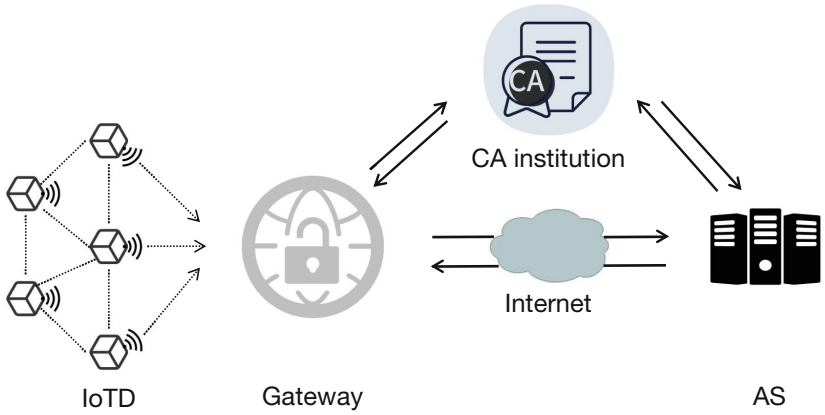

Fig. 2. The architecture of a gateway based IoT system.

3.1 The Framework of WB-GWS

According to the architecture of a gateway based IoT system, a gateway and a dedicated AS are illustrated in Fig. 2. IoT-side gateway (IoTS-GW) and server-side gateway (SS-GW) are added to the gateway and AS, respectively. Before the IoTS-GW and the SS-GW communicates, a virtual tunnel is established through Internet. The tunnel is located at the transport layer of the TCP/IP protocol family. Furthermore, when data is transmitted through the tunnel, WBC is used for confidentiality and integrity. This process is transparent to sender and receiver. Consequently, for managing the LUTs and virtual IP addresses, the white-box cryptography service (WBCS) was introduced into WB-GWS. WB-GWS adds IoTS-GW, SS-GW, and WBCS to the original gateway based IoT system, which is depicted in Fig. 3. Typically, WBCS and SS-GW are deployed in the same Intranet environment, so the channel between them is secure. Although WBCS and IoTS-GW use an insecure channel, the LUTs obtained by IoTS-GW can only be used for encryption and will be replaced periodically. Even if an attacker obtains a LUT from IoTS-GW, it cannot decrypt the transmitted message. The functionalities of each part of WB-GWS are summarized as follows.

- **IoTD:** IoT devices are the sender of data transmissions, such as temperature and humidity sensing.

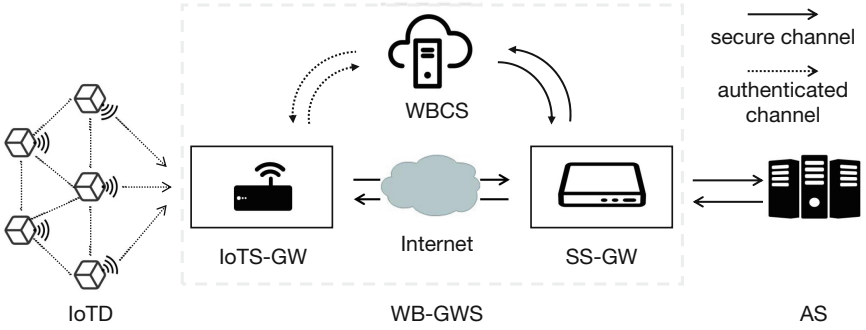

Fig. 3. The architecture of WB-GWS.

- **IoTS-GW:** Establishes a security tunnel with the SS-GW and encrypts the data. IoTS-GW is used to collect data generated by IoT devices and send them to SS-GW.
- **SS-GW:** Establishes a security tunnel with the IoTS-GW and decrypts the data from the IoTS-GW. SS-GW also transfers its received data to AS. A SS-GW can communicate with multiple IoTS-GWs simultaneously.
- **WBCS:** Managing LUTs and virtual IP addresses. WBCS is also responsible for message and client authentications.
- **AS:** The application server for the underlying IoT system.

3.2 The Protocols of WB-GWS

The main process of WB-GWS can be divided into four phases: bootstrap, WBCS establish, data transmission and release connection.

Bootstrap. This protocol occurs when WB-GWS is initialized or a new GN requests for registration. This process includes creating virtual IP addresses, generating LUTs, and downloading LUTs, which are described in Fig. 4. A formal definition can be represented in the following steps.

1. First a request of REGISTER is generated by a gateway node (denoted by v, and V_* represents the gateway nodes set).
2. WBCS generates a series of data including virtual IP address, LUTs, *token*, and version number, and responds to gateway node with IP address, *token*, and version number.
3. While the response is confirmed, the gateway node will make another request to WBCS based on the returned *token* to download the LUTs. We note that both IoTS-GW and SS-GW need to run the bootstrap protocol with WBCS.

Protocol 1. Bootstrap protocol

$\forall v \in \mathrm{V}_*$ (set of GN),

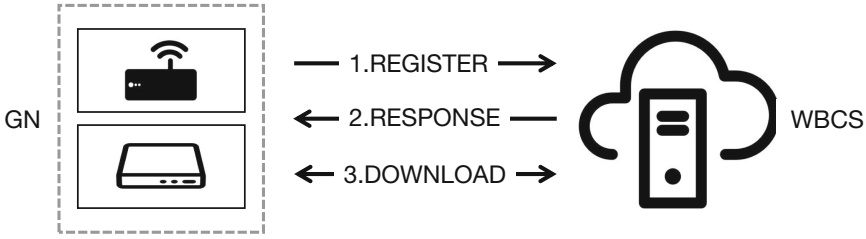

Fig. 4. The bootstrap phase.

$v \longrightarrow WBCS$ (White-box cryptography server):

$$\textbf{REGISTER} \| \ ID_v \ \| \ N_v$$

$$WBCS \quad : LUT = LUT\_Gen\{ \ rand \ \}$$

$$token = Token\_Gen\{ID_v \ \| \ rand\}$$

$$Virtual\_IP = Virtual\_IP\_Gen\{ID_v\}$$

$$Add\_To\_Reg\{v\}$$

$$WBCS \longrightarrow v : token \ \| \ Virtual\_IP \ \| \ Ver$$

$$v \quad : Check\{v\}$$

$$v \longrightarrow WBCS \ : \textbf{DOWNLOAD} \ \| \ token$$

WBCS Establish. Before data transmission, a virtual channel will be created between the IoTS-GW and SS-GW with a virtual IP assigned by WBCS. And then a connection is established over the virtual channel, which is described in Fig. 5. Device authentication will also be performed during connection establishment. The procedure is described in Protocol 2 and the steps can be described as follows.

1. IoTS-GW makes a request for WBCS to obtain the *Virtual_IP* of the server and the tag to establish the connection.
2. WBCS generates a tag and returns the tag and *Virtual_IP* of SS-GW to IoTS-GW.
3. IoTS-GW encrypts the tag by using the WBC and random IV, and then sends the ciphertext, IV, ID to SS-GW.
4. SS-GW uses WBC and IV to decrypt the ciphertext and sends the plaintext and ID to WBCS for verification.
5. WBCS checks whether the plaintext is the same as the authentication code sent to the IoTS-GW, and returns the verification result to SS-GW.
6. SS-GW determines the verification result returned by WBCS. And SS-GW returns the final result to IoTS-GW.

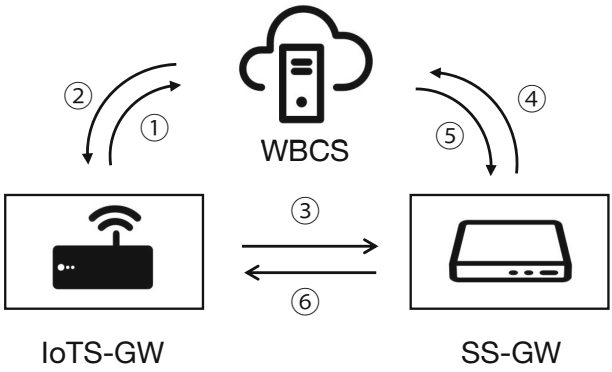

Fig. 5. The WBCS establish phase.

Protocol 2. WBCS establish protocol

$\forall v \in \mathrm{V}_*$ (IoTS-GW set),

$v \longrightarrow WBCS$: **REQUEST** $\| ID_v \| N_v$

$WBCS$: $Flag = Flag\_Gen\{\cdot\}$

$WBCS \longrightarrow v$: **RESPONSE** $\| Flag \| Virtual\_IP$

v : $CMsg = WBEnc\{PMsg \| IV\}$

$v \longrightarrow SS\text{–}GW$: **REQUEST** $\| ID_v \| CMsg \| IV$

$SS\text{–}GW$: $PMsg = WBDec\{CMsg \| IV\}$

$SS\text{–}GW \longrightarrow WBCS$: **REQUEST** $\| ID_v \| PMsg$

$WBCS$: $Ret = Verify\{v\}$

$WBCS \longrightarrow SS\text{–}GW$: **RESPONSE** $\| Ret$

$SS\text{–}GW \longrightarrow v$: **RESPONSE** $\| Ret$

Data Transmission. Data transmission between IoTS-GW and SS-GW takes place through a secure virtual tunnel that is created in WBCS establish. During communication between IoTS-GW and SS-GW, data will be encrypted and decrypted using WBC. Whilst message authentication codes are also necessary for verifying data integrity (Fig. 6).

Protocol 3. Data transmission protocol

$IoTS\text{–}GW$: $CMsg = WBEnc\{PMsg\}$,
$Tag = Tag\_Gen\{CMsg \| IV\}$

$IoTS\text{–}GW \longrightarrow SS\text{–}GW$: **REQUEST** $\| CMsg \| Tag \| Ver$

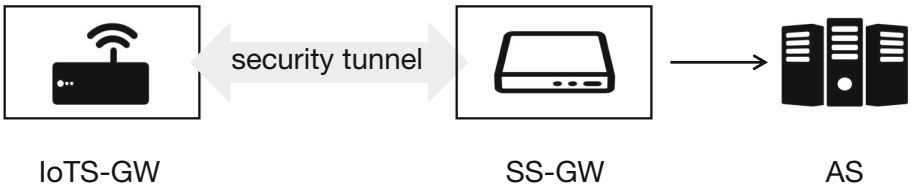

Fig. 6. The data transmission phase.

$$
\begin{aligned}
&\textit{SS–GW} && : Ret = Auth\{Tag\} \\
& && \quad PMsg = WBDec\{CMsg\} \\
&\textit{SS–GW} \longrightarrow \textit{AS} && : Send\_To\_AS\{PMsg\}
\end{aligned}
$$

Release Connection. This protocol will take effect under one of the following circumstances: 1) GN is out of service. 2) IoTS-GW sends a request to release the connection.

Protocol 4. Release connection protocol

$\forall v \in \mathrm{V}_*$ (set of GN),

Case 1: v is out of service.

WBCS : Free{ LUT ‖ *Virtual_IP* }

: Remove_From_Reg{ v }

Case 2: IoTS-GW sends a request to release the connection

$v \longrightarrow$ WBCS : **REQUEST** ‖ *release the connection*

WBCS $\longrightarrow v$: **RESPONSE** ‖ *acknowledgement character*

v : Shutdown{v}

WBCS : Free{ LUT ‖ *Virtual_IP* }

: Remove_From_Reg{ v }

4 Gateway Addition

WB-GWS supports multiple IoTS-GW nodes. For adding IoTS-GW nodes during system initialization, new nodes can also be added while the system is running. WB-GWS will dynamically assign *Virtual_IPs* to new nodes, create LUTs, and add new nodes to the registration list. After new nodes are added successfully, they can communicate with SS-GW in the system.

Among the three core parts of WB-GWS, only IoTS-GW supports multi-node deployment. The SS-GW node and WBCS can run only on a single node.

To add a new IoTS-GW node to WB-GWS, one should execute the following steps.

1. Bootstraps a new IoTS-GW node and sends a registration request to WBCS.
2. WBCS assigns virtual IP addresses to IoTS-GW and creates LUTs.
3. IoTS-GW, SS-GW, and WBCS establish connections.
4. The WBCS adds new IoTS-GW node to the registration list.

5 Gateway Removal

To reduce system overhead, one can remove an IoTS-GW node while it is out of service (Protocol 4). There are two conditions that can trigger a node removal operation: 1) The WB-GWS will remove IoTS-GW after it is out of service. 2) When an IoTS-GW node is unnecessary, IoTS-GW can actively send a request to remove it. When an IoTS-GW node is removed, the virtual IP address and LUTs of IoTS-GW will be released as well.

6 Security Analysis

IoTS-GW Access Authentication. Device access security is one of the important factors of IoT system security. Current IoT security solutions include standards-based proposals. Among them, the TLS and VPN are the de-facto protocols for communication security in the IoT. These two protocols support authentication using the symmetric pre-shared key (PSK) or public-key certificates. Although PSK based authentication consumes a small amount of computing resources. There exist key scalability and management issues when using the PSK. Additionally, the PSK established out of band is assailable to attacks due to restricted security features and uncontrolled deployment environment. Certificate-based authentication can resolve many problems where specifically PSK-based solutions fall short, so plenty of IoT systems choose certificates as the authentication method. However, it also has some problems such as cumbersome use and difficult certificate revocation. In IoTS-GW, we assign LUTs for device access authentication. Only IoTS-GW with a LUT can be successfully authenticated. The WB-GWS has the same authentication security as PSK in IoT systems. In addition, the problem of an attacker sending data through a forged IoTS-GW after acquiring a LUT can be solved by device binding or APP binding [5,8,28] technology, which is not implemented in this paper.

Security of Data Transmission. In IoT systems using TLS and VPN protocols, the security of data transmission is guaranteed by cryptographic Primitive. Data is encrypted and integrity checked during data transmission. After data is encrypted using an encryption algorithm, it is transmitted in ciphertext over the network. An attacker cannot obtain useful information even after listening to ciphertext messages, which ensures the confidentiality of data transmission. Because data is checked for integrity, attackers cannot tamper or manipulate it.

In other words, even if an attacker modifies to the data, they can be detected. The WB-GWS also encrypts and verifies the integrity of data. Because of the secure WBC algorithm used in WB-GWS, it is difficult for an attacker to recover plaintext from ciphertext. Therefore WB-GWS has the same security of data transmission as TLS and VPN. Meanwhile, WB-GWS is more flexible and secure than the traditional black-box mode when the LUT is lost.

Forward Security After GN Remove or Release. There are forward security issues with the pre-sharing key (PSK) mode. After an attacker obtains pre-sharing key, all the secret messages transmitted using the key are broken. For IoT systems that use public-key cryptography, a new session key is created for each session. The attacker obtains the current session key at a certain moment. Because the key is changed, the attacker cannot decrypt the previous session information with this key. Therefore, there is no forward security problem in IoT systems using public-key. WB-GWS takes the following three measures to ensure forward security. 1) Regularly WB-GWS will update the distributed LUTs. 2) For GNs that are removed or released, their LUTs become invalid. 3) The LUTs of each GN can only be used for encryption or decryption, *i.e.*, just like public-key algorithm.

7 Experiment

In order to verify the performance of WB-GWS, two experiments are implemented separately. The first is the experiment based on temperature and humidity (T/H) sensor system. The other is a performance comparison experiment. Nevertheless, the data transmission and encryption speeds of WB-GWS are tested on different platforms.

Table 2. The testbenches of WB-GWS modules.

Module	CPU	software
IoTD	T/H sensor DHT11 [16]	–
IoTS-GW	ARMv7 Processor rev @1.20 GHz	Linux Raspberry Pi 5.10.17-v7+
SS-GW	Intel (R) Xeon (R) Platinum 8255C CPU @2.50 GHz	Linux VM-0-13-centos 3.10.0-1127.19.1.e17.x86_64
WBCS	Intel (R) Xeon (R) Platinum 8255C CPU @2.50 GHz	Linux VM-0-13-centos 3.10.0-1127.19.1.e17.x86_64
AS	Intel (R) Xeon (R) Platinum 8255C CPU @2.50 GHz	Linux VM-0-13-centos 3.10.0-1127.19.1.e17.x86_64

7.1 The Experiment Base on Temperature and Humidity Sensor System

In this experiment, the T/H sensor is used as an IoT device. IoTS-GW and SS-GW are deployed on Raspberry Pi and VM-0–13-centos respectively. The Table 2 describes the CPU and software of WB-GWS modules. We let the T/H sensor send T/H data to the AS every 10 s. Wireshark [13] was used to capture the data transmitted between the IoTS-GW and the SS-GW, but the plaintext of the data could not be seen. Moreover, we kept the WB-GWS running for 30 days and counted the receipts received in the database, and none of the data was lost.

JMeter [23] is a performance testing tool originally designed primarily for testing web applications. But it can also be extended to test in other areas. In this experiment, JMeter is used to test the pressure of the system. System loads of 50, 100, 200, and 300 requests per second were tested. The test results are shown in Table 3 (Fig. 7).

Table 3. System pressure test results.

Requests per second	Expected receive	Actual receive	Lost
50	50	50	0
100	100	100	0
150	150	120	30
200	200	120	80
300	300	120	180

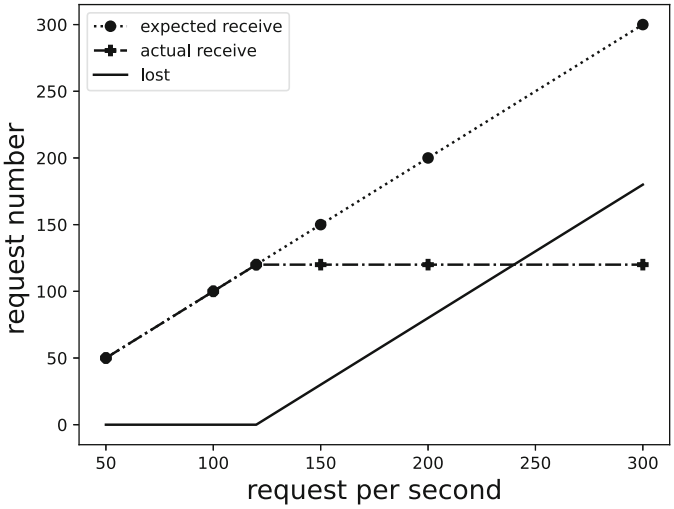

Fig. 7. System pressure test result.

According to the performance test results, data loss occurs when the number of concurrent requests exceeds 100, and the maximum number of concurrent requests is 120. Due to the limitations of the experimental environment, we did not conduct experiments in more IoT scenarios. However, through this experiment, we found that WB-GWS can be successfully run in the IoT environment with low data transfer.

7.2 The Performance Comparison

Transmission rate is one of the most critical evaluation metrics used to measure the performance of the data transferring in the IoT networks. Firstly, the benchmark test of WBC in each platform is done. Secondly, we tested the transfer rate on different platforms by transferring files from the IoTS-GW to the SS-GW. The file transfer monitors the average transfer rate for 1000 experiments by using tools such as Netperf, Iperf. In our experiments, we use Raspberry Pi 3 B (ARMv7 Processor rev @1.20 GHz) as IoTS-GW, use different platforms as SS-GW. The configurations and performance of different platforms are demonstrated as shown in Table 4. In practice, SS-GW will not be deployed on the Raspberry Pi due to its lack of performance. Therefore the transmission speed on Raspberry Pi is not given. The result indicates that the bottleneck of transmission speed for WB-GWS is the hardware limitations of Raspberry Pi.

Table 4. The performance of WB-GWS on different platforms.

Platform	Configuration		Transmission speed (Mbit/s)	Encryption speed (Mbit/s)
	CPU	software		
Raspberry Pi	ARMv7 Processor rev @1.20 GHz	Linux Raspberry Pi 5.10.17-v7+	–	2.48
mac OS	Intel Core i5 @1.6 GHz	macOS Big Sur 11.5.2	0.97	20.82
Linux	Intel (R) Xeon (R) Platinum 8255 C CPU @2.50 GHz	Linux VM-0–13- centos 3.10.0–11 27.19.1.e17.x86_64	1.32	22.24
Windows	Intel (R) Core (TM) i7-11800 H @2.30 GHz	Windows 10 20 H2	1.58	52.83

This paper does not provide a comparative analysis of efficiency with existing schemes, due to the following reasons. 1) There are few studies on the application of white-box cryptography to secure data transmission in the IoT, and there is no mature solutions for the time being. 2) Compared with solutions such as SSL VPN, this research scheme has too many variables to make an effective comparison.

8 Conclusions

Device authentication and secure data transmission play a vital role in secure communication between smart IoT objects. WB-GWS provides a solution for

secure data transmission in IoT systems by using WBC in combination with tunneling technology. The proposed solution of WB-GWS can compatible with legacy IoT systems which have lack of cryptographic function support. Also, it can manage keys efficiently with the help of WBC. The disadvantage of WB-GWS is that it is currently only applicable to gateway-based IoT architectures, not to the architectures where devices can directly connect to cloud services. In future work, it is challenging to improve the performance of WB-GWS and to provide a secure WBC instance for the implementation of WB-GWS.

Acknowledgments. We are grateful to the anonymous reviewers for their insightful comments. This work was supported in part by National Natural Science Foundation of China (62072192) and National Defense Technology 173 Basic Improvement Project (2121-JCJQ-JJ-0931).

References

1. Aboubakar, M., Kellil, M., Roux, P.: A review of IoT network management: current status and perspectives. Elsevier (2021)
2. Ammar, M., Russello, G., Crispo, B.: Internet of things: a survey on the security of IoT frameworks. **38** (2018). https://doi.org/10.1016/j.jisa.2017.11.002
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. **54** (2010)
4. Aziz, B.: A formal model and analysis of an IoT protocol. Ad Hoc Netw. **36**, 49–57 (2016). https://doi.org/10.1016/j.adhoc.2015.05.013
5. Balfanz, D., Smetters, D.K., Stewart, P., Wong, H.C.: Talking to strangers: authentication in ad-hoc wireless networks. In: NDSS. Citeseer (2002)
6. Berger, T.: Analysis of current VPN technologies. In: Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, 20–22 April 2006, Vienna University of Technology, Austria. IEEE Computer Society (2006). https://doi.org/10.1109/ARES.2006.30
7. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 63–84. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_4
8. Bock, E.A., Amadori, A., Brzuska, C., Michiels, W.: On the security goals of white-box cryptography. IACR Trans. Crypt. Hardware Embed. Syst. **2020**, 327–357 (2020)
9. Bogdanov, A., Isobe, T.: White-box cryptography revisited: space-hard ciphers. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015, pp. 1058–1069. ACM (2015). https://doi.org/10.1145/2810103.2813699
10. Bogdanov, A., Isobe, T., Tischhauser, E.: Towards practical whitebox cryptography: optimizing efficiency and space hardness. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 126–158. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_5

11. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: hiding your white-box designs is not enough. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 215–236. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_11
12. Chow, S., Eisen, P., Johnson, H., Van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_17
13. Combs, G.: Wireshark: go deep. https://www.wireshark.org/. Accessed 14 Nov 2021
14. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-04722-4
15. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45203-4_23
16. elink: Wifi temperature and humidity sensor DHT11 version (support MQTT, Baidu Cloud IoT, etc.). https://item.taobao.com/item.htm?spm=a1z09.2.0.0.3ce72e8d8radF8&id=622084892800&_u=fvrdd9137b5. Accessed 14 Nov 2021
17. Granjal, J., Monteiro, E., Silva, J.S.: Security for the internet of things: A survey of existing protocols and open research issues. **17** (2015). https://doi.org/10.1109/COMST.2015.2388550
18. Juma, M., Monem, A.A., Shaalan, K.: Hybrid end-to-end VPN security approach for smart IoT objects. **158** (2020). https://doi.org/10.1016/j.jnca.2020.102598
19. Kouicem, D.E., Bouabdallah, A., Lakhlef, H.: Internet of things security: a top-down survey. **141** (2018). https://doi.org/10.1016/j.comnet.2018.03.012
20. Lee, S., Jho, N., Kim, M.: Table redundancy method for protecting against fault attacks. **9** (2021). https://doi.org/10.1109/ACCESS.2021.3092314
21. Lee, S., Kim, M.: Improvement on a masked white-box cryptographic implementation. **8** (2020). https://doi.org/10.1109/ACCESS.2020.2993651
22. Li, P., Su, J., Wang, X.: ITLS: lightweight transport-layer security protocol for IOT with minimal latency and perfect forward secrecy. IEEE Internet Things J. **7**(8), 6828–6841 (2020). https://doi.org/10.1109/JIOT.2020.2988126
23. Mazzocchi, S.: Apache JMeter. https://github.com/apache/jmeter. Accessed 22 Feb. 2022
24. Muir, J.A.: A tutorial on white-box AES (2013). http://eprint.iacr.org/2013/104
25. De Mulder, Y., Roelse, P., Preneel, B.: Cryptanalysis of the Xiao – Lai white-box AES implementation. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 34–49. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_3
26. Nguyen, K.T., Laurent, M., Oualha, N.: Survey on secure communication protocols for the internet of things. Ad Hoc Netw. **32**, 17–31 (2015). https://doi.org/10.1016/j.adhoc.2015.01.006
27. Noor, M.B.M., Hassan, W.H.: Current research on Internet of Things (IoT) security: a survey. **148** (2019). https://doi.org/10.1016/j.comnet.2018.11.025
28. Pan, S., et al.: Universense: IoT device pairing through heterogeneous sensing signals. In: Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications, pp. 55–60 (2018)
29. Radoglou-Grammatikis, P.I., Sarigiannidis, P.G., Moscholios, I.D.: Securing the internet of things: challenges, threats and solutions. Internet Things **5**, 41–70 (2019). https://doi.org/10.1016/j.iot.2018.11.003

30. Rahman, R.A., Shah, B.: Security analysis of IoT protocols: a focus in CoAP. In: 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC). IEEE (2016)
31. Saha, A., Srinivasan, C.: White-box cryptography based data encryption-decryption scheme for IoT environment. In: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). IEEE (2019)
32. Tang, Y., Gong, Z., Sun, T., Chen, J., Zhang, F.: Adaptive side-channel analysis model and its applications to White-box block cipher implementations. In: Yu, Yu., Yung, M. (eds.) Inscrypt 2021. LNCS, vol. 13007, pp. 399–417. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88323-2_22
33. Xiao, Y., Lai, X.: A secure implementation of white-box AES. In: 2009 2nd International Conference on Computer Science and its Applications, pp. 1–6. IEEE (2009)
34. Yang, G., Xu, J., Chen, W., Qi, Z.H., Wang, H.Y.: Security characteristic and technology in the internet of things, vol. 30. Nanjing University of Posts and Telecommunication (2010)
35. Yuan, R., Strayer, W.T.: Virtual Private Networks: Technologies and Solutions. Addison-Wesley Longman Publishing Co., Inc. (2001)