

A Novel Trusted Boot Model for Embedded Smart Device without TPM

Rui Wang*, Yonghang Yan**

*School of Computer Science and Technology, Xidian University, Xi'an, China

**School of Computer Science and Information Engineering, Henan University, Kaifeng, China, **Corresponding Author

ruigelwang@163.com, yanyonghang@henu.edu.cn

Abstract— Embedded smart devices are widely used in people's life, and the security problems of embedded smart devices are becoming more and more prominent. Meanwhile lots of methods based on software have been presented to boot the system safely and ensure the security of the system execution environment. However, it is easy to attack and destroy the methods based on software, which will cause that the security of the system cannot be guaranteed. Trusted Computing Group proposed the method of using Trusted Platform Module (TPM) to authenticate the credibility of the platform, which can solve the disadvantages of using methods based on software to protect the system. However, due to the limited resource and volume of embedded smart devices, it is impossible to deploy TPM on embedded smart devices to ensure the security of the system operating environment. Therefore, a novel trusted boot model for embedded smart devices without TPM is proposed in this paper, in which a device with TPM provides trusted service to realize the trusted boot of embedded smart devices without TPM through the network and ensure the credibility of the system execution environment.

Keywords— Trusted Boot; Embedded Smart Devices; Trusted Services; TPM

I. INTRODUCTION

With the rapid development of Internet of Things and computing technology, the application of embedded smart devices is more and more popular and extensive in people's daily life. People use embedded smart devices to process all kinds of daily work, business communication and store all kinds of information. These devices are interconnected through the network and cooperate with each other, thus providing more powerful task execution ability, which bring great convenience to human life. However, due to the openness of network and unavoidable system vulnerabilities, embedded smart devices face more and more prominent security threats and suffer more and more attacks.

In order to ensure the security of embedded smart devices and avoid malicious attacks, many methods have been presented by researchers. Jiang et al. [1] proposed an ARM-based secure boot scheme for the isolated execution environment, which is based on TrustZone technology. Kumar et al. [2] proposed a method called Post-Quantum Secure Boot (PQSB) that was implemented by fully hardware for reasons of security and performance consideration to ensure the integrity

of the trusted computing base of a security system. Kim et al. [3] proposed a method based on platform integrity verification for activating a mobile trusted module (MTM). Ehret et al. [4] designed a security focused low power SoC architecture with a hardware root-of-trust for use in edge devices. The architecture alleviates deployment time threats faced by edge devices and meets the power and energy constraints. G. Pocklassery et al. [5] proposed a Self-Authenticated Secure Boot method (SASB) for FPGAs. Zimmo et al. [6] proposed trusted boot method for embedded systems using hypothesis testing benchmark. This method contains two parts: achieving the hypothesis testing benchmark values and the trusted boot method based on the hypothesis testing benchmark values. Profentzas et al. [7] presented a software-based secure boot method for embedded devices. This method relies on several assumptions. The first is that the security of the boot stages before bootloader U-Boot cannot be modified. The second is that U-Boot has to be placed in read-only memory since there is no prior verification of the booting process. Chakraborty et al. [8] presented a software based trusted boot method called simTPM for mobile devices. The simTPM was implemented on a Hikey960 using a Gemalto Multos card as SIM card. However, the above methods are software-based solutions that cannot be used once the device system is tampered. TPM proposed by TCG provides a powerful guarantee for the protection of confidentiality, integrity and authenticity of information systems. At present, the security protection technology of general computing platform based on TPM has been studied and applied deeply. However, a large number of embedded smart devices cannot directly deploy TPM due to resource constraints and strict volume requirements. In view of various problems, existing solutions are not suitable for embedded smart devices with limited resource and volume. A new trusted boot model for embedded smart devices is proposed in this paper. In this scheme, the devices that deploy TPM are used to provide trusted services to the devices without TPM protection through the network, so as to guarantee the trusted boot of embedded smart devices.

The rest of the paper is organized as follows. In the next section the system model is introduced and detailed. Section III presents our algorithm design. In Section IV, the presented scheme is proved by BAN logic, and the validity of the trusted

boot model is proved by security analysis. Section V is the summary of the full paper.

II. SYSTEM MODEL

A. Model Assumption

In this paper, a novel trusted boot model for embedded smart devices is studied, and the model assumptions used are as follows:

Assumption 1: embedded smart devices without TPM (NTESD) and their network environment are not trusted, and may be attacked by attackers at any time.

Assumption 2: attackers invade NTESD in order to eavesdrop on intelligence, falsify data or add some of their own information modules to the system disguising as an effective node for data monitoring. Therefore, it is assumed that the attackers will not replace the operating system files in this model.

Assumption 3: embedded smart devices with TPM (TESD) are always trusted and the trusted services they provide are also trusted. The authentication key in TSED and the shared key used to send the message are secure and cannot be accessed by attackers.

Assumption 4: the NTESD system update and the initial integrity measurement of the system should be carried out in a secure environment. This assumption can be implemented when the network is not connected or when the connected network is secure. The measurement value of NTESD in the trusted state is stored in the non-volatile memory of TPM through USB disk transmission, or the trusted value is transmitted through the network when NTESD is in the secure environment.

Assumption 5: the private key stored in the kernel of NTESD and the shared key between TSED and NTESD will not be obtained by the attacker. If the attackers can obtain the private key stored in the kernel for authentication and the symmetric key used for information encryption, so the security of this model will not be guaranteed.

B. Trusted Boot Model

The trusted boot model of NTESD is shown in Fig.1, which consists of two parts: TSED and NTESD. TSED is the provider of trusted service, and NTESD is the requester of trusted service. They communicate through the network to send the request and the response of trusted service.

TSED contains three entities: TPM, TCG Software Stack (TSS) and Trusted Service Module (TSM). TPM and TSS are key components for building trusted platforms and providing trusted services. TSM is a user module of TSED through the network providing trusted service to NTESD and this module uses the TPM's non-volatile memory to store NTESD's trusted state measurement value to provide trusted services for each NTESD. As a trusted service server (TS-Server), TSM receives messages about trusted service request sent from NTESD and will supply certification results to the corresponding NTESD.

NTESD contains three entities: positive trusted authentication chain first stage (PTAC1), reverse trusted authentication chain (RTAC) and positive trusted

authentication chain second stage (PTAC2). As a part of U-Boot, PTAC1 is used to guarantee the trusted boot of the system kernel. As a part of the kernel, RTAC is used to measure the status value of the currently booted kernel, U-Boot and platform hardware. As a trusted service client (TS-Client), RTAC sends trusted service requests to TS-Server in TSED through the network to authenticate the current system status value calculated by RTAC. After authenticating the kernel, U-Boot and platform hardware, PTAC2 as a part of the kernel continues to boot the system to start the Init process credibly.

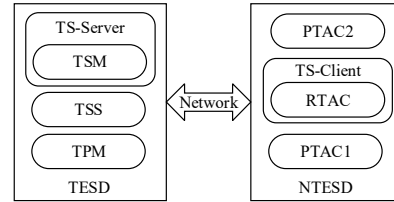


Figure 1. Trusted boot model for embedded smart devices without TPM.

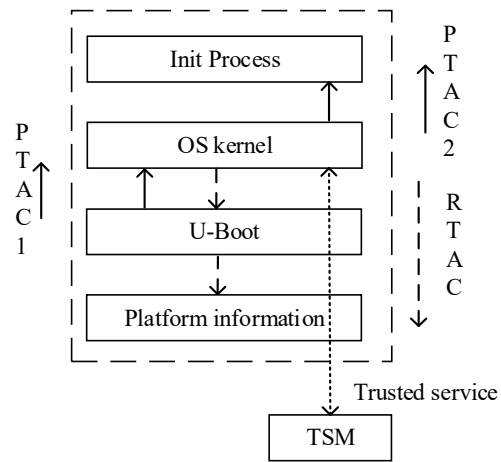


Figure note: Arrows are executed from left to right

- ↑ stand for performing positive trusted authentication chain
- ↓ stand for performing reverse trusted authentication
- ⇕ stand for performing Trusted service interaction process

Figure 2. Trusted boot model for embedded smart devices without TPM.

The trusted boot process of NTESD includes the startup of platform hardware and the startup of operating system software. As shown in Fig.2, the platform hardware starts from the power-up of the device to the end of the U-Boot initialization of the device. The execution of the PTAC1 starts from the beginning of U-Boot. U-Boot authenticates the kernel image. If the result of authentication is not trusted, the system boot is terminated, or else to start the kernel. Then the kernel starts to execute the RTAC. The implementation of the RTAC needs to first measure the integrity of the kernel image being started, and then expand to measure the integrity of the executable file of U-Boot that has been started. Then the kernel collects the platform hardware information of the system, such as MAC address and hardware serial number, then calculate the

hash of the kernel, U-Boot and the platform hardware information. Finally, the hash value of the reverse integrity measurement is obtained.

The system kernel passes the measurements calculated during the reverse authentication process to the TSM in the remote TESD. When the TSM receives the request message, it compares the received measures with the trustworthiness values prestored in the TPM on the platform on which the TSM resides. If the comparison results are consistent, the current state of the system is identified as trusted state. If the comparison results are inconsistent, the current state of the system is identified as untrusted state and the result is returned to the trusted service client of requesting NTESD. After the kernel module RTAC of NTESD receives the authentication result returned from TSM on the remote TESD, RTAC will start PTAC2, if the current system state is trusted. PTAC2 first measures and authenticates the integrity of the Init process. If the authentication status is credible, the Init process will be started.

III. ALGORITHM DESIGN

This section proposes three protocols: forward authentication protocol (FAP), reverse authentication protocol (RAP) and trusted service protocol (TSP) to implement the above trusted boot model for embedded smart devices without TPM.

A. Forward Authentication Protocol

The FAP contains two stages. The first stage is executed using the U-Boot module of NTESD to ensure the reliable boot of the system kernel. After the RAP authenticates the credibility of the currently booted system through the trusted service, the second stage is executed to ensure the trusted boot of the started Init process. The implementation of this protocol involves three entities: NTESD (D_N), PTAC1 in the U-Boot of the D_N and RTAC in the kernel of the D_N .

The implementation of the protocol is mainly based on the trust chain technology, which measures the state value of the system successively, and the system can be booted only if the state is trusted. In the process of FAP protocol implementation, D_N powers up and then U-Boot gets the control. At this time, PTAC1 is started. D_N 's PTAC1 check whether the state value of the system kernel is consistent with the state of the trusted kernel prestored in the U-Boot. If the result is inconsistent, the current kernel state is untrusted and the boot is terminated. If the results are consistent, the system kernel is started and control of the system is handed to the kernel. Then the RTAC is started and it executes the RAP and the TSP. Through reverse authentication, the integrity of the current system status value is measured and verified. If the reverse authentication fails, the boot process will terminate, or else the PTAC2 is started. PTAC2 checks whether the system Init process has been tampered. If the integrity measurement of the system Init process is consistent with the integrity measurement prestored in the kernel, then the Init process can be trusted to boot. If the Init process is not trusted, the boot terminates. The pseudocode for this protocol is shown in Table 1.

B. Reverse Authentication Protocol

The RAP will be executed after the NTESD kernel is started. The protocol reversely measures the integrity of the system that has been initially started to guarantee the credibility of the U-Boot program and the hardware platform, and it authenticates the measured hash value through the trusted service protocol TSP. The implementation of this protocol mainly involves two entities: the NTESD (D_N), and the RTAC module in the D_N 's kernel module.

TABLE 1. ALGORITHM 1: FORWARD AUTHENTICATION PROTOCOL ALGORITHM

Input: OS kernel image, Init

Output: success or failure

Hash_Ker = CheckKernel(OS kernel image);

if (IsHash_Ker())

RAP(); TSP();

if(IsTrusted_RAP_TSP())

Hash_Init = CheckInit(Init);

if (IsHash_Init()) success;

else failure;

else failure;

TABLE 2. ALGORITHM 2: REVERSE AUTHENTICATION PROTOCOL

Input: U-Boot, OS kernel image

Output: success or failure

Sys_Firm = getFirm();

Hash_Reverse = CheckReverse(U-Boot, OS kernel image, Init Sys_Firm);

Result_TSP = TSP()

if (IsResult_TSP()) success;

else failure;

This protocol obtains the platform information firstly, then calculates the hash value of the OS kernel, U-Boot and the platform information. After that, RTAC will send the hash value representing the current state of the system to trusted TSP, and eventually receive authentication result of TSP. If the current system state value is consistent with the state value prestored in TPM, it indicates that the system is trusted and the system is trusted to boot. If the authentication status values are inconsistent, the system has been tampered. The pseudocode for this protocol is shown in Table 2.

C. Trusted Service Protocol

The trusted service protocol uses trusted service provided by TESD to compare the system status value of NTESD with the

one prestored in the TPM of TESD and return the comparison result to the RTAC of RAP protocol. The implementation of this protocol mainly involves four entities: the NTESD (D_N), the TESD (D_T), the RTAC module designed in the kernel module of D_N and the TSM designed in the D_T platform.

The key to realize this protocol is to ensure the integrity, confidentiality, authenticity, validity and other characteristics of messages during the communication between D_N and D_T . To ensure that the TSP has these characteristics, the request message R_{D_N} is designed as follows:

- 1) Random number N_{D_N} is used to prevent replay attacks.
- 2) The timestamp TS_{D_N} is used to ensure the freshness of the message.
- 3) D_N uses the shared key between D_T and D_N to encrypt the status value of the current system state information to obtain E_{D_N} , and encrypts the system state value with the shared key between D_T and D_N , which can satisfy the confidentiality of the transmitted message.
- 4) Valid signatures Sig_{D_N} of the hash value of the message R_{D_N} .

Once TSM of D_T receives trusted service request message R_{D_N} from D_N , it will firstly authenticate the freshness and integrity of the message, and then it decrypts Sig_{D_N} to get the hash value H_{D_N} of the message. At the same time, the TSM calculates hash value $H_{D_N}^* = \text{Hash}(N_{D_N} \parallel TS_{D_N} \parallel E_{D_N})$, then compares H_{D_N} and $H_{D_N}^*$. If H_{D_N} and $H_{D_N}^*$ is not equal, the authentication fails. If it is equal, then authenticates N_{D_N} to check whether the message has been received before. If so, the message will be discarded. And then the timestamp TS_{D_N} is authenticated to check whether the message has timed out. And if so, RTAC is required to resend the message. If authentication fails L times, D_N is considered an untrusted node.

After checking the integrity and effectiveness of the message, the TSM decrypts E_{D_N} to get the integrity measurement value of D_N . At the same time, the TSM reads the trusted status information H_{N_V} of D_N from nonvolatile memory of TPM and compares H_{N_V} with the integrity measurement value of D_N decrypted from E_{D_N} to judge whether D_N is destroyed. And then response message generated by TSM is sent to D_N . The response message RSP_{D_T} contains the N_{D_T} to prevent replay attacks, the timestamp TS_{D_T} to ensure the validity of the time and the E_{D_T} which is a system status value encrypted by the shared key between D_T and D_N . And the response message also includes a valid signature for the hash value of RSP_{D_T} , which is used by D_N to check the integrity and authenticity of the message. When D_N receives the response message, the same authentication process is performed as for D_T . After the RTAC of D_N gets the authentication message of the system status, it performs the same operation as TSM when it receives the message to get the confidential data (authentication result). If the result is trusted, PTAC2 will be started and the operating system will continue to be booted forwardly. If the result is not trusted, the boot will

be terminated. The pseudocode of this protocol is shown in Table 3 and 4.

TABLE 3. ALGORITHM 3: TRUSTED SERVICE PROTOCOL ALGORITHM -- D_N SENDS MESSAGE TO D_T

```

 $N_{D_N} = \text{random}(); TS_{D_N} = \text{time}();$  list value;
 $E_{D_N} = \text{Enc}(\text{value}, \text{key});$ 
list  $R_{D_N} = \{N_{D_N}, TS_{D_N}, E_{D_N}\};$ 
 $H_{D_N} = \text{Hash}(N_{D_N} \parallel TS_{D_N} \parallel E_{D_N});$ 
 $Sig_{D_N} = \text{Sig}(H_{D_N}, \text{Priv}_{D_N});$ 
message =  $\{R_{D_N}, Sig_{D_N}\};$ 
Send(message);

```

TABLE 4. ALGORITHM 4: TRUSTED SERVICE PROTOCOL ALGORITHM -- D_T RESPONSES TO D_N

```

if  $N_{D_N}$  is invalid then drop message; num = 0;
while (num < L)
{
  if  $TS_{D_N}$  timeout
    Send_Retry(); num ++;
  else break;
}
if (num == L) Return_false();
 $H_{D_N} = \text{Verify}(Sig_{D_N}, \text{Pub}_{D_N});$ 
 $H_{D_N}^* = \text{Hash}(N_{D_N} \parallel TS_{D_N} \parallel E_{D_N});$ 
if ( $H_{D_N} == H_{D_N}^*$ ) {
  value = Dec( $E_{D_N}$ , key);
  if (value == read_NV())
    Return_true();
  else Return_false();
}
else Return_false();

```

IV. SECURITY ANALYSES

A. BAN logic proof

BAN logic [9] is used to prove the security of protocols presented in this paper. Before proof of the protocol, the communication parties in the protocol, the assumption of the protocol and the goal to be proved are defined firstly.

In this paper, the security protocol involved in the trusted service sharing mechanism of embedded smart devices proposed in this paper is formally proved by BAN logic analysis method. subject A is the TESD, while subject B is the NTESD.

(1) Security protocols are described in BAN logic terms.

message 1: $A \triangleleft \text{message} = \{R_{D_N}, Sig_{D_N}\}$

message 2: $A \triangleleft R_{D_N} = \{N_{D_N}, E_{D_N}\}$
 message 3: $A \triangleleft Sig_{D_N} = \langle H_{D_N} \rangle_{K^{-1}}$
 message 4: $A \triangleleft E_{D_N} = \{value\}_Y$

(2) Safety goal

goal 1: $A \equiv (\text{Hash}_{value})$

goal 2: $A \equiv (value)$

(3) Initial assumption

assumption 1: $A \equiv A \xleftrightarrow{Y} B$

assumption 2: $A \equiv B \xleftrightarrow{K}$

assumption 3: $A \equiv \#(N_{D_N})$

assumption 4: $A \equiv B \Rightarrow \{R_{D_N} = \{N_{D_N}, E_{D_N} = \{value\}_Y\}, Sig_{D_N} = \langle H_{D_N} \rangle_{K^{-1}}\}$

(4) Analysis and conclusion

① proof $A \equiv (H_{D_N})$

equation (1) can be obtained from assumption 3, message 2 and the fresh rule.

$$A \equiv \#\{N_{D_N}, E_{D_N}\} \text{ that is } A \equiv \#(R_{D_N}) \quad (1)$$

equation (2) can be obtained from equation (1), message 1, message 3 and the fresh rule.

$$A \equiv \#(Sig_{D_N}) \text{ that is } A \equiv \#(H_{D_N}) \quad (2)$$

equation (3) can be obtained from assumption 2, message 3 and the message meaning rules (for public keys).

$$A \equiv B \sim (H_{D_N}) \quad (3)$$

equation (4) can be obtained from equation (2), equation (3) and the nonce verification rule.

$$A \equiv B \equiv (H_{D_N}) \quad (4)$$

equation (5) can be obtained from assumption 4, equation (4) and the jurisdiction rule.

$$A \equiv (H_{D_N}) \quad (5)$$

② proof $A \equiv (value)$

equation (6) can be obtained from assumption 3, message 2, message 4 and the fresh rule.

$$A \equiv \#\{N_{D_N}, E_{D_N}\} \text{ that is } A \equiv \#(value) \quad (6)$$

equation (7) can be obtained from assumption 1, message 4 and the message meaning rules (For shared keys).

$$A \equiv B \sim (value) \quad (7)$$

equation (8) can be obtained from equation (6), equation (7) and the nonce verification rule.

$$A \equiv B \equiv (value) \quad (8)$$

equation (9) can be obtained from assumption 4, equation (8) and the jurisdiction rule.

$$A \equiv (value) \quad (9)$$

The above proofs show the effective information value and the H_{D_N} that guarantees the integrity of the message are not tampered and still trusted, when TESD receives the message from NTESD. When the NTESD receiving the message from the TESD, the proof procedure is the same as the above proof procedure.

B. security analysis

The security protocol used in the trusted service sharing scheme of embedded smart devices is analyzed.

Preventing attackers from attacking the message content sent by the sender. In the security protocol of this paper, the designed message sent by the sender mainly contains two parts:

the first part includes the secret information encrypted by the shared key between the sender and the receiver, the random number used to prevent replay attack, the time stamp and other information to prove the validity of the message. The second part includes calculating the hash value of the first part and signing the hash value with the sender's signature private key to ensure the authenticity of the message. If the attacker modifies the contents in the first part, the receiver will find that the message has been tampered when verifying its integrity after receiving the message. If the attacker modifies the contents in the second part, the attacker does not have the signature private key of the sender of the message. So, it cannot legally sign the message after forging the message. Therefore, the receiver will find that the message has been modified when verifying the authenticity of the message after receiving the message.

Preventing the attacker from replaying the message. In the security protocol in this paper, a random number of anti-replay attack is placed in each message. When receiving the message, the receiver will judge whether the message is valid or not according to whether the number has appeared before.

Preventing attackers from eavesdropping and illegal reading of messages. In the security protocol of this paper, when the sender sends confidential data that cannot be eavesdropped, the confidential data will be encrypted with the shared key of the sender and the receiver. Because the attacker does not have the shared key, so it cannot be eavesdropped.

Preventing the attacker from sending the message to the receiver after hijacking. In the security protocol of this paper, the message sent by the sender contains a time stamp to ensure the freshness of the message. If the message exceeds the time limit, it will be considered invalid and the receiver will demand the sender to resend the valid information. If the attacker changes the timestamp of the delayed message after the intercept, the receiver will find out when the message is received for integrity authentication because the timestamp message was hashed before it was sent. If the timestamp is changed, this change will be found when the hash integrity is authenticated.

Preventing attackers from forging attacks. When the attacker wants to forge the sender to send a message to the receiver, it needs to use the private key of the sender for identity authentication and the shared key of the sender and the receiver to encrypt the confidential data. However, the sender's private key and shared secret key cannot be obtained by the attacker, so the security protocol in this paper can prevent the attacker from forging attack.

V. CONCLUSION

For embedded smart devices without TPM, traditional software-based methods are often used to ensure the security of the system. However, once the security software is attacked, the security of the system will not be guaranteed. A novel trusted boot model to provide trusted service for NTESD is presented in this paper, which uses TPM in TESD to make NTESD boot credibly. The security protocols presented in this

paper are proved by BAN logic, which can prove that they are credible.

REFERENCES

- [1] Hang Jiang et al. "Implementing a ARM-Based Secure Boot Scheme for the Isolated Execution Environment." 2017 13th International Conference on Computational Intelligence and Security (CIS).
- [2] Vinay B. Y. Kumar et al. "Post-Quantum Secure Boot" 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE).
- [3] Daewon Kim et al. "A method based on platform integrity verification for activating a mobile trusted module" 2015 International Conference on Information and Communication Technology Convergence (ICTC).
- [4] Alan Ehret et al. "A Hardware Root-of-Trust Design for Low-Power SoC Edge Devices" 2020 IEEE High Performance Extreme Computing Conference (HPEC).
- [5] G. Pocklassery et al. "Self-authenticating secure boot for FPGAs" 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST).
- [6] Sara Zimmo et al. "Trusted Boot for Embedded Systems Using Hypothesis Testing Benchmark" 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).
- [7] Christos Profentzas et al. "Performance of Secure Boot in Embedded Systems" 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS).
- [8] Dhiman Chakraborty et al. "simTPM: User-centric TPM for Mobile Devices." 28th USENIX Security Symposium, page:533-550.
- [9] Burrows M, Abadi M, Needham R M.: A logic of authentication. proceedings of the Royal Society A Mathematical Physical & Engineering Sciences, 1989, 426(1871):1-1.



Rui Wang (M'2021, Luoyang, 1995) received B.S. degree in computer science and technology from Henan University at 2018. And now she is a postgraduate in computer science and technology from Xidian University, Xi'an, China. Her research interest contains trusted computing.



Yonghang Yan (Zhoukou, 1981) received B.S. degree in computer science and technology from Zhengzhou University at 2004, M.S. degree in computer science and technology from Beijing Institute of Technology at 2007 and Ph.D degree in computer science and technology from Beijing Institute of Technology at 2014. His research interests include computer network, Mobile Ad hoc and Sensor Network, UAV network, mobile computing network and QoS.

Now he is an associate professor in the School of Computer and Information Engineering at Henan University, Kaifeng, China. He is the head of advanced network technology laboratory.