TMSUI: A Trust Management Scheme of USB Storage Devices for Industrial Control Systems

Bo Yang[1](✉), Yu Qin[1], Yingjun Zhang[1], Weijin Wang[1], and Dengguo Feng[1,2]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
{yangbo,qin_yu,zhangyingjun,wangweijin,Feng}@tca.iscas.ac.cn
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract. The security of sensitive data and the safety of control signal are two core issues in industrial control system (ICS). However, the prevalence of USB storage devices brings a great challenge on protecting ICS in those respects. Unfortunately, there is currently no solution specially for ICS to provide a complete defense against communication between untrusted USB storage devices and critical equipment without forbidding normal USB device function. This paper proposes a trust management scheme of USB storage devices for ICS (TMSUI). By fully considering application scenarios, TMSUI is designed based on security chip to flexibly achieve authorizing a certain USB storage device to only access some exact protected terminals in ICS for a particular period of time. The scheme enables administrators to revoke authorized devices. We analyze six security properties of TMSUI. The prototype system is finally implemented. The evaluation results indicate that our scheme meets the security goals with high compatibility and good efficiency.

Keywords: Trust management · USB storage device · Security chip · Industrial control system · Industrial security · TPM/TCM

1 Introduction

The easiness to carry around and the simplicity to access a variety of computers are two significant advantages of USB storage devices. USB flash disk, memory card, mobile HDD, smartphone and many other embedded devices can all be treated as specific forms of USB storage devices. With regard to them, the main functions cover data storage and offline communication between remote terminals. Undoubtedly, the prevalence of USB storage devices benefits the masses, including the engineers who manage and operate the modern ICS. Updating software, uploading and downloading industrial data give the engineers chances to attach USB storage devices to ICS. However, USB storage devices are becoming the dangerous media to potentially threaten the security of ICS.

By the end of June 2014, a total of 549 vulnerabilities of ICS were publicly reported by American Common Vulnerabilities and Exposures (CVE) [23], the

S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 152–168, 2016.
DOI: 10.1007/978-3-319-29814-6_13

Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) of US Department Homeland Security [26] and China National Vulnerability Database (CNVD) [21]. From the report [25], the number of security incidents of ICS shows a rapid growth trend. These incidents involve several key industry areas. Sadly, many serious malicious codes, including evil scripts and varied malwares such as viruses, worms, spyware and Trojan horses, are able to invade ICS together with USB-based hack tools through USB storage devices [30]. Especially, the exclusive core part of ICS is relatively (or absolutely somewhere) isolated from the external network, which dramatically reduces the risks of attacks via Internet, but therefore, USB storage devices become the more dangerous tools potentially used by attackers to spread malicious codes into ICS. Known as Autorun malwares, some malicious codes exploits operating system (OS) vulnerabilities to replicate via USB drives and automatically activate themselves. Both the notorious Stuxnet, which attacked Iranian nuclear power plant, and the latest Havex, which could shut down the nationwide power grid, try to access ICS originally via USB storage devices [29]. Although the local firewall, anti-virus software and intrusion detection system (IDS) protect ICS from attacks to some extent, 0-day and unknown vulnerabilities of ICS still leave opportunities for attackers. Additionally, in Black Hat USA 2014, BadUSB is presented as a new form of malware that operates from controller chips inside USB devices which can be reprogrammed to spoof various other device types in order to take control of a computer, exfiltrate data, or spy on the user [15].

On the other hand, launching social engineering attacks using USB storage devices is considered to cause more damage than malicious codes themselves in ICS [28]. The unauthorized access behavior of USB storage devices taken by a corrupt engineer, an evil staff member or an infiltrator is able to easily upload malicious codes to ICS or unimpededly steal valuable data from ICS. In fact, the lack of authentication and restriction on USB storage devices and permitting every device to easily access ICS are the reasons that lead to these risky situations. Therefore, building up a trust management scheme of USB storage devices is necessary to keep the hazardous devices away from ICS.

Nevertheless, compared with traditional computer and network systems, ICS architecture has its own characteristics:

- **Isolated.** Not only ICS is partly isolated from external network, but also the different sections inside ICS are isolated from each other to some extent. The network communication between terminals is limited and costly.
- **Complicated.** In order to be well managed, controlled and supervised, ICS is divided to several system or network sections which make it complicated.
- **Stable.** For guaranteeing continuous running with no mistake, ICS architecture and equipment require to remain both reliable and stable, and cannot tolerate large-scale changes.

For these characteristics, applying some existing comprehensive technologies to ICS, for example using Public Key Infrastructure (PKI) to issue certificates to USB devices, is not appropriate because they are inflexible and may further increase ICS's complexity, instability and performance overhead. Likewise, the real-time online solutions such as Online Certificate Status Protocol (OCSP) are hardly applicable for ICS either. Furthermore, the existing access control mechanisms in Windows and Linux do not fulfill the enough fine-grained control over USB devices. To our best knowledge, there is no complete solution specially designed for the architecture of ICS.

Our Contributions. In this paper, we propose TMSUI, a trust management scheme combined with related security rules. The scheme is able to authorize some certain USB storage devices to have limited permission to access specific protected terminals such as engineer stations and supervisory control and data acquisition (SCADA) servers in ICS, while other unauthorized devices are directly ejected by the target terminals. As a result, the use of USB storage devices in ICS is meticulously controllable. Our work is summarized as follows.

- We design TMSUI expressly for ICS without changing its architecture. Based on security chip and digital signature technique, the administrators have the privilege to authorize USB storage devices flexibly and at a fine granular level. Meanwhile, the administrators are held accountable for their respective misconduct. The authorization is time-limited.
- We elaborate an offline whitelist mechanism with no need for each terminal to download and hold a huge whitelist. A lightweight revocation strategy is presented for remediation in case the authorized devices are stolen.
- We analyze six security properties achieved by our TMSUI.
- We implement a prototype system with full functions of TMSUI and the evaluation results show its good effectiveness with high compatibility and low performance overhead.

2 Related Work

There are many studies focusing on security issues of ICS and several approaches are proposed against threats to ICS. The protection measures in SCADA networks provided by access control, firewalls, IDS, protocol vulnerability assessment and cryptography are discussed in [13]. Artificial immune algorithm, which borrows the ideas from the modeling of human immune system, is used in ICS in order to detect and stop the intrusion [7]. Attack tree model [19] is leveraged to evaluate the vulnerabilities of cybersecurity in SCADA system. Infrastructure vulnerability assessment model [12] is conductive to quantify the vulnerabilities in ICS networks. TCG-based integrity measurement architecture [18] provides a method to control the untrusted processes in operating system and establish the trusted execution environment for software in ICS. [9] emphasizes that the application of technology alone will not provide solutions, but human factors can cause the insider threat and even easily destroy the secure environment of ICS. The latest NIST guide to ICS security [14] standardizes both the security technology used in ICS architecture and the ICS-specific security policies that regularize employees' behaviors. However, few publications explicitly mention the defense on the threats from USB storage devices and existing security measures hardly forbid USB storage devices accessing to steal sensitive data from ICS or upload malicious codes to ICS unless prohibiting USB devices completely.

Trusted computing (TC), as one security support technology, aims at constructing a specific integrity measurement mechanism to prevent untrusted codes from executing on the computing platform [36,37]. TC can report on the hash-value of software modules and help platform to build up trust chain. Using TC for x86-based hardware platform, Trusted Computing Group (TCG) proposes Trusted Platform Module (TPM) [31], while China proposes Trusted Cryptography Module (TCM) [22]. Nowadays, the widely used TPM chip is designed according to TPM v1.2 specification [32]. TCG has already released the latest TPM v2.0 specification [33], which absorbs some techniques from TCM including serial algorithms of Standard Commercial Cryptography (SM). TPM has been generally employed to construct trusted execution environment for security-sensitive computers in the similar way of [18]. TPM and TCM are the alternative security chips to build up our TMSUI. The unique endorsement key of the chip fixed on the mainboard is tamper-resistant and can be leveraged to identify the terminal of ICS. The chip's tamper-resistant cryptographic algorithm library crucially maintains the trust relationship between the administrators and the protected terminals.

Apart from system and network security, some research pays attention to the trust management and security issues of USB storage devices. Sinan et al. design and implement a security platform for USB flash disks [11]. The scheme only targets Windows OS and needs to deeply modify the kernel drivers. Moreover, it protects against only some of the known malwares. A method of USB device management in Linux is proposed by [10], which concerns the access control to USB device and defends against the attacks from computers. Some configure policies are given by [17] for Windows to block malwares and hack tools from USB devices. But it is considered vulnerable without concrete security technology. [20] details the forensic evidence for USB devices as a supplementary approach to trace the devices that behave abnormally. Physical information fixed in the firmware of USB device is treated as the identification which is hardly tampered. And recently, IronKey Secure USB devices [24] are recommended to protect against BadUSB malware using signed firmware. Nonetheless, the specially-made devices with high cost are unlikely to be accepted in large scale by enterprises. In general, all of these approaches cannot singly achieve a trust management scheme that is directly applied to the architecture of ICS and imposes fine-grained restrictions on USB storage devices.

3 Overview

This section gives an overview of our design, including the system architecture, our threat model and assumptions, and the design principles.

3.1 System Architecture

On the basis of ICS deployment diagram [28], we abstract general ICS architecture and omit some repeated or irrelevant components according to the USB storage devices application scenarios. Figure 1 illustrates the system architecture of our TMSUI. It consists of three sections. The left section belongs to corporate network which serves for inner-enterprise applications such as ERP system and top monitoring system. It provides the only interface of the whole enterprise to access Internet, while some enterprises disable the external access interface for more security needs. Authorization server (AS) locates this area as one part of enterprise management systems for administrators to authorize USB storage devices. There can be several ASs with several administrators. The middle section is part of process control network which optionally contains engineer station, SCADA server and many other control and supervising terminals such as operation station. Established on general computers, these terminals are responsible for programing industrial controllers, collecting industrial operation data and automatically altering controllers based on exceptional data. Their security is so vital that some errors intentionally caused by malicious codes may lead to faulty operation of controllers and finally cause an accident. Moreover some confidential data on these terminals indeed need good protection from theft through USB storage devices. Thus, these terminals are our protected objects (PO) and equipped with security chip. The right section consists in control system network where some specialized control devices run. These devices involve programmable logic controller (PLC), programmable automation controller (PAC), remote terminal unit (RTU) and intelligent electronic device (IED). They are the only components of ICS accessing field devices and rarely attacked directly. The networks in the left and the middle sections adopt general structure of Ethernet, while the right one often uses filedbus structure or industrial Ethernet with protocols such as Modbus [27] and Profibus [35].

3.2 Threat Model and Assumptions

We seek to protect the confidentiality and integrity of industrial data on PO and keep threats from USB storage devices away. More precisely, our core goal is to forbid unauthorized (untrusted) USB storage devices to access PO. An adversary is assumed to use arbitrarily extraneous USB storage devices to attempt accessing PO. Malicious codes including malwares and evil scripts can be injected into these USB devices. The adversary is also able to tamper the physical information of his own USB storage devices with the similar information of the formal USB devices used inside the ICS. More powerfully, the adversary may randomly steal an authorized USB storage device to access arbitrary PO[1], and modify its physical information or format it in order to access his ideal attacking target.

Our solution does not consider the direct attack itself on network, system, databases or integrity of our software, which is beyond the protection scope in

[1] The adversary has a very low probability in ICS to find and access the right one of PO that is just his ideal target and could be accessed by the stolen USB device.

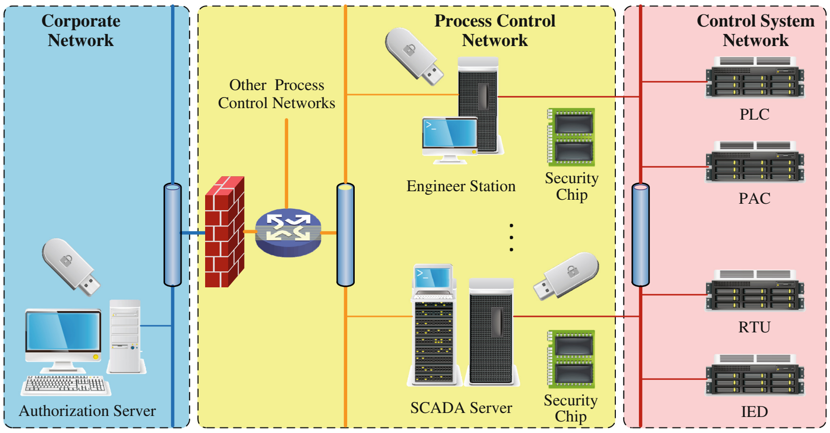

Fig. 1. System architecture of TMSUI.

this paper. The traditional safeguards like IDS, firewall, anti-virus software or isolated execution environment schemes [16,18] could be deployed along with TMSUI. Moreover our technical solution cannot absolutely stop the social engineering attack. For example, it is hard to prevent that a corrupt engineer "steals" his own authorized USB storage device to exactly access the PO which he often operates.

We explicitly lock the permission to change the OS time. It can be achieved by setting the OS and BIOS on terminals.

3.3 Design Principles

We propose TMSUI according to following desired design principles.

Strong Compatibility. The scheme is designed for a majority of ICSs. It executes on general computers with x86 architecture and is suitable for both Windows and GNU/Linux. The scope of USB storage devices covers mainstream devices including USB flash disk, memory card, mobile HDD, smartphone and embedded devices.

Low Cost. The scheme does not change the system architecture of ICS or require other special equipment except for security chip. On the one hand, many manufacturers have already equipped computers with TPM/TCM chips as standard components. On the other hand, it is easy and inexpensive to install an additional security chip into the legacy hardware through inserting chip into its PCI slot.

Centralized and Efficient Management. The scheme guarantees administrators have the privilege to authorize USB storage devices, manage trust relationship and revoke authorization. The administrators are regarded as proficient and faithful employees on the side of ASs. The model of rights allocation and

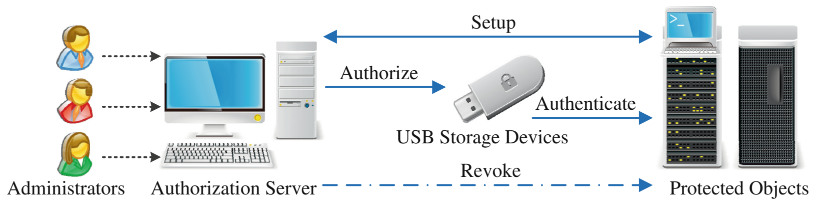

Fig. 2. Execution process of TMSUI.

trust transfer forms the centralized and efficient management, ensures the scheme security and meets the requirement of ICS supervision principle.

High Security. The scheme is designed to realize the goal described in Sect. 3.2 and provide protection with sound security properties. The attacks on the whitelist scheme and USB storage device cannot effectively threaten the security of our PO.

4 Trust Management Scheme

In this section, we detail our scheme. The execution process is first illustrated. Then we specify the phases of TMSUI. The security rules are finally presented.

4.1 Execution Process

The proposed scheme is composed of four phases: Setup, Authorize, Authenticate and Revoke. And there are mainly four kinds of entities participating in the execution process, including administrator, AS, PO and USB storage device. Setup only needs to securely execute at the very first time when the scheme is installed and launched in ICS or some considerable changes take place in ICS. After Setup, Authorize and Authenticate execute repeatedly for normally using USB storage devices. Revoke is a supplementary phase that can execute at any time after Setup. Figure 2 shows the execution process of the proposed scheme. To simplify the description of our TMSUI, we assume only one AS here with several administrators. In practice, another synchronization server could be added for coordinating data sharing among all the ASs.

4.2 Detailed Phases

The scheme takes one USB storage device as example. The physical serial number sn[2] and vendor ID vid of USB storage device could together uniquely identify the USB device [8,34]. The security chip owns a unique endorsement key ek

[2] On the market, a few USB storage devices do not have their serial numbers. It is reasonable and feasible to forbid using these devices here. In fact, these devices cannot pass the authentication of PO and will be ejected directly.

bound with the legal and secure identity for everyone of PO in ICS. This key is a pair of asymmetric keys in the form of (epk, esk), in which epk is the public key and esk is the private key. For the sake of security, esk never leaves the security chip. AS has an original database table that records all the legal epk of the security chips with which the enterprise has equipped PO. The information of the security chips could come from the chip manufacturer or be exported by the enterprise using a specific tool. In addition, respective administrator ID number $adminID$ with password $pswd$ for login in the AS is assigned to each administrator. The four phases of our TMSUI are described in detail as follow.

Setup. This phase initializes several keys and transmits original data between AS and PO for future use.

1. AS generates public parameters $param$ based on cryptography rules and randomly chooses a master key seed mks for creating other keys. Particularly, mks must be securely stored in AS. The revocation list $RevokeList$ is also generated by AS even if it is blank now.
2. There could be many administrators participating in the scheme. We take the administrator α for instance. The administrator α uses his ID number and password to login the AS and generates his authorizing key, which is a pair of asymmetric keys (apk_α, ask_α) by a key derivation function (KDF) with inputs of $param$, mks, ctr and $\mathsf{H}(pswd_\alpha)$. ctr is a monotonous counter value that ensures generating an unique key pair. $\mathsf{H}()$ represents a hash function. The generating process runs as

$$pdigest_\alpha \leftarrow \mathsf{H}(pswd_\alpha), \quad (apk_\alpha, ask_\alpha) \leftarrow \mathsf{KDF}(param, mks, ctr, pdigest_\alpha).$$

   The private key ask_α represents α's identity in the later phases and becomes the forensic evidence when inside attacks occur. AS automatically generates storage protection key k_α by a pseudorandom number generator (PRNG) and then saves α's authorizing key after encryption action ENC. These two steps are

$$k_\alpha \leftarrow \mathsf{PRNG}(mks, pdigest_\alpha), \quad blob_\alpha \leftarrow \mathsf{ENC}_{k_\alpha}(apk_\alpha, ask_\alpha).$$

   Likewise, all the administrators generate their own authorizing keys, storage protection keys and encrypted key blobs in this phase.
3. One of the PO ρ calls the driver of security chip to export the public key epk_ρ, and then sends it with more information, such as its plant area ID, production line ID and its usage, to AS for registering its legal identity. All the PO do this procedure similarly. If using TCM chip, for example, the epk_ρ exporting method is like

$$epk_\rho \leftarrow \mathsf{Tspi\_TCM\_GetPubEndorsementKey}().$$

   Tspi_TCM_GetPubEndorsementKey is TCM API for exporting public ek.
4. After receiving epk_ρ, AS queries the original database table for checking the key's legitimacy. Passing the check, AS recodes the related information of ρ bound with its unique epk_ρ into a registration database table. At the end

of Setup, AS sends *param*, *RevokeList* and all the effective administrators' public authorizing keys (*apk*) with their respective identifications (*adminID*) back to the successful registrants of PO. PO calls security chip to use its inside storage root key *srk* to seal each *apk* and save the ciphertext *EncData* in the local hard disk:

$$EncData \leftarrow \mathsf{Tspi\_Data\_Seal}_{srk}(apk).$$

Authorize. In this phase, an unauthorized USB storage device is taken to AS. After confirming security of the device, one administrator operates to build up the trust relationship between it and some exact PO for a certain period of time.

1. In the light of the requirements for using USB storage device in ICS, engineers, staff members or operators could ask an administrator α to authorize the device. Plugging into the AS, USB storage device is first scanned and checked for malicious codes and other threats by traditional anti-virus software. Relying on rich experience, the administrator will judge whether the USB storage device is secure enough to be used in ICS.
2. After detection, the physical information of the USB storage device, including *sn* and *vid*, is extracted by AS. And then, with the physical information, some additional information such as brand, type, usage and holder are together recorded in a device management database table.
3. The administrator α uses his ID number and password to login the AS. Automatically, AS decrypts his authorizing key using decryption function DEC as

$$pdigest_\alpha \leftarrow \mathsf{H}(pswd_\alpha), \quad k_\alpha \leftarrow \mathsf{PRNG}(mks, pdigest_\alpha),$$

$$(apk_\alpha, ask_\alpha) \leftarrow \mathsf{DEC}_{k_\alpha}(blob_\alpha).$$

4. From the user of the USB storage device, α confirms which terminal of PO the user is going to access. For the selected terminal ρ, the corresponding epk_ρ is obtained from the registration database table. Then, the expiry date *exp* of authorization for USB storage device is determined by the negotiation between the user and α. The authorization for the device is in the form of a digital signature σ signed by α. The signature is generated at AS by calling function SIG using the private authorizing key of α. The detailed process of generating σ runs as follow.

$$sdigest \leftarrow \mathsf{H}(exp, epk_\rho, sn, vid), \quad \sigma \leftarrow \mathsf{SIG}_{ask_\alpha}(sdigest, param).$$

   AS does not sign the data content of the USB storage device, which allows staff to use the authorized device repeatedly and flexibly after this phase. The enough security is guaranteed and controlled by the expiry date.
5. An authorization item is finally created and written into the authorization whitelist saved in a document in the USB storage device. The attribute of the document is set to "readonly" and "hidden" for preventing the whitelist from being modified unintentionally. The pattern of the item is a quadruple form like $(exp, epk_\rho, adminID, \sigma)$. *adminID* identifies which administrator

Algorithm 1. Whitelist Verification Algorithm

```
Input: $I$: set of items in whitelist and each $item$ contains $(exp, epk, adminID, \sigma)$
    $clt$: current local time; $E$: set of $EncData$; $epk_\rho$; $sn$; $vid$; $param$
Output: $result$
 1: $result \leftarrow false$
 2: for $item \in I$ and $result \neq true$ do
 3:     $i \leftarrow adminID$
 4:     if not_expired$(exp, clt)$ and $epk = epk_\rho$ and $EncData_i \in E$ then
 5:         $sdigest \leftarrow$ Tspi_Hash_GetHashValue$(exp, epk_\rho, sn, vid)$
 6:         $apk \leftarrow$ Tspi_Data_Unseal$_{srk}(EncData_i)$;    Tspi_Key_LoadKey$(apk)$
 7:         $result \leftarrow$ Tspi_Hash_VerifySignature$_{apk}(\sigma, sdigest, param)$
 8:     end if
 9: end for
10: if $result = false$ then eject_USB_storage_device()
11: end if
12: return $result$
```

issues the authorization item and actually implies which public authorizing key is applicable to verify the signature σ. If the user wants to access several different PO, a few items could be simultaneously created using different *epk*. All the data used for generating items are saved in the registration database table as backup for revocation and forensics.

Authenticate. When a USB storage device is taken to one of PO such as ρ, this phase is triggered on ρ for authenticating whether the device is authorized legitimately and effectively. Some actions will be taken depending on the authentication result.

1. As a USB storage device is plugged into ρ, *sn* and *vid* are firstly extracted similar to the way in Authorize phase. And then, epk_ρ is exported from the security chip as ρ did in Setup phase.
2. Authentication process checks whether the authorization of USB storage device is revoked by AS. ρ tries to match *sn* and *vid* with each item in *RevokeList*. If one item is matched, the USB storage device is immediately ejected and Authenticate phase is terminated.
3. In this step, ρ reads out the whitelist in the USB storage device and uses security chip to check it. Algorithm 1 shows the specific verification algorithm for checking the whitelist. Four TCM APIs are called including Tspi_Hash_GetHashValue for computing the digest, Tspi_Data_Unseal for unsealing public authorizing key, Tspi_Key_LoadKey for loading the key into TCM and Tspi_Hash_VerifySignature for verifying the signature. The returned value *false* indicates the ejection of USB device and the termination of Authenticate phase. Conversely, the value *true* indicates the approval for normal use of USB device on ρ.

Revoke. This phase allows a revocation of some still effective authorizations of the lost USB storage devices. The lightweight strategy acts as a kind of remedial measure.

1. When a staff member notices his authorized USB storage device μ is missing, he would report the situation to one of administrators.
2. After receiving the report, the administrator looks up the corresponding sn_μ and vid_μ in the device management database table relying on the description of the lost USB storage device.
3. The administrator operates AS to make sure that there are really some authorization items of μ within its validity period. Using sn_μ and vid_μ to search in the registration database table, every related items exp is selected out and checked. If there is an unexpired authorization item, the following steps are executed.
4. AS adds sn_μ and vid_μ to *RevokeList*. In the meantime, AS validates every revoked information item again in the current *RevokeList*. The revoked information item is checked whether the corresponding authorization is expired and if it is, the revoked information item is deleted from *RevokeList*. In this way, *RevokeList* is updated into a new version.
5. According to the specific needs, AS could distribute the new updated version of *RevokeList* to all the PO with a certain frequency such as once a month or half a year, which would not cost much network traffic.

4.3 Security Rules

The technical solution of TMSUI could dramatically reduce the threat from externally unknown USB storage devices. However, the authorized devices may be infected later and unwittingly carry malicious codes to access PO. We propose the following security operation rules recommended as a complementary part of TMSUI:

– authorize USB storage devices with the period of validity as short as possible,
– carefully scan USB storage devices every time before authorizing it.

5 Security Analysis

In this section, we give an informal security analysis on TMSUI scheme. The proposed scheme satisfies the following security properties based on the assumptions in Sect. 3.2.

Correctness. TMSUI does not aim at any certain USB-based attacks, but its Authenticate phase theoretically prevents the unauthorized access and the invasion of most malicious codes from USB storage devices, as long as it is not authorized. The security chip uniquely identifies each of PO, and the unite of sn and vid uniquely identifies each of USB storage devices. Consequently, the flexible and specific whitelist scheme achieves our design goal.

Difficulty in Faking. If the adversary tries his best to tamper the physical information of his USB device and employ it to access the ideal target PO, he has to obtain the ideal *sn* and *vid* of the corresponding device for the target PO. It is definitely no easier than directly stealing a both authorized and ideal USB device from the inside of ICS, i.e., launching a complete social engineering attack.

Unforgeability. A valid whitelist in USB storage devices can only be created by a legal administrator through Authorize phase. The whitelist is signed using *ask* which is encrypted and saved only in AS. The whitelist is verified using *apk* which is protected by security chip in PO. Adversaries hardly forge or tamper a valid whitelist without stealing *ask*, cracking security chip or breaking cryptographic algorithms. Preventing these attacks is beyond the scope of this paper.

Copy Protection. The adversary may attempt to copy an ideal and legal whitelist from the authorized USB device to his illegal device and then use it furtively in ICS. Nevertheless, the signed whitelist contains the original *sn* and *vid* which are not match the illegal device. Therefore, PO would not admit the replicate but unmatched whitelist during Authenticate phase.

Non-repudiation. The administrators cannot deny their authorizing operation on the exact USB storage devices because of the signature associated with their unique *adminID* and authorizing keys. Once the authorized USB device is stolen or malicious act is revealed, the related user and administrator could be sought out based on the relevant evidence in the registration database.

Revocability. The scheme considers revocation issue and enables revoking the authorization of the authorized USB storage devices. Revoke phase could reduce the threat and loss to a certain extent once the devices are stolen. PO are able to refuse the access of adversaries who try to use the revoked devices.

6 Implementation and Evaluation

This section describes the prototype system and evaluation of TMSUI on it.

The implementation of TMSUI is divided into a server part for AS and a client part for PO. In our prototype system, Windows platform is the target OS. Microsoft Windows device development kit (DDK) is used to develop bottom driver and achieve capturing, deleting and ejecting actions for USB storage devices. The bottom drive could load the public parameters to verify the whitelist in USB devices. We acquire the physical information of USB storage devices by reading some Windows registry entries mainly under the path: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\. This method is generally feasible for different USB protocol standards including 1.1, 2.0 and 3.0. Besides, we accomplish to capture the event of devices' access by means of registering a daemon as system service running in background on both the server side and the client side. MySQL 5.6 is adopted to construct database on the server. In regard to cryptographic algorithms, we employ high-performance SM serials algorithms [22], also supported by TCM chip. We implement or set ENC, DEC, Tspi_Data_Seal and Tspi_Data_Unseal using symmetric cryptographic algorithm SMS4, SIG and Tspi_Hash_VerifySignature using digital signature algorithm SM2, and the hash function using SM3. In the client, these algorithms are directly provided by the security chip. The server of TMSUI totally consists of 3500 lines of C++ code, while the client has 2500 lines. In fact, TMSUI is not merely designed for one OS platform. The technic skills in [10] can help us implement TMSUI also on GNU/Linux system.

To establish the prototype system environment, we use a DELL OptiPlex 990 as AS. It has a 3.3 GHz Intel i3-2120 processor, 4 GB memory and USB 2.0 ports, and runs Windows 7 SP1. For PO, we use two of Lenovo ThinkCentre M8500t equipped with a 3.4 GHz Intel i7-4770 processor, 8 GB memory and USB 3.0 ports. These two PO respectively run Windows XP SP3 and Windows Server 2003 SP2, as well as some industrial control software for simulating an engineer station and a SCADA server. Moreover, we choose TCM 1.1 by Tongfang Microelectronics Company as the security chip fixed inside the two PO. As PLC, a Honeywell PKS C200 is connected with the two PO for testing the impact of the potential threat from USB storage devices.

Table 1 shows the different kinds of USB storage devices we select to test compatibility of TMSUI on the prototype system. Only two devices cannot be applicable to TMSUI. For Apple device, it is unable to directly access PO without its dedicated driver. And for SD card, it works well when using the same card reader (reader1) to be authorized and authenticated, while it is invalid when using different card reader (reader2) during those two phases. This is because the physical information correlated with SD card is actually from the card reader.

Table 1. Compatibility of TMSUI for USB storage devices

Device	Toshiba 16 GB U-disk	Kingston 32 GB U-disk	WD 2TB mobile HDD	SanDisk 64 GB SD card + reader1
USB standard	2.0	3.0	3.0	2.0
Applicable	✔	✔	✔	✔
Device	Apple iPad mini 16 GB	BeagleBone-Black demo board	Samsung S4 smartphone	SanDisk 64 GB SD card + reader2
USB standard	2.0	2.0	2.0	2.0
Applicable	✘	✔	✔	✘

On the other hand, the executing cases of TMSUI definitely attest that it is impossible to download data from and upload data to PO through an unauthorized USB storage device or a wrongly authorized device. In order to evaluate protective property on defending malicious codes in unauthorized devices,

Table 2. Security of TMSUI for defending malwares in unauthorized devices

Malware	USBC [6]	RavMonE [3]	Stuxnet [5]
Defensible	✔	✔	✔
Malware	Havex [2]	Red Oct. [4]	BadUSB [1]
Defensible	✔	✔	partly

we pick out six typical malwares, including virus, Trojans, worm for ICS and BadUSB, and we save their samples in an unauthorized USB disk to access our POs for the test. Some malware samples are downloaded from public web pages [1,2,5] with source codes, while the others [3,4,6] are selected from our previous collection. By detecting whether POs are invaded by these malwares and whether PLC is disordered, TMSUI's effectiveness is examined. Especially, although Stuxnet only targets Siemens PLC and Havex hunts for SCADA systems using OPC communication standard, we can still identify their invasions on our experimental POs by monitoring their preliminary behaviors such as malicious scanning. Table 2 illustrates the test results. TMSUI is effective to prevent invasion of almost all the test malwares except BadUSB. With regard to BadUSB, defensibility of TMSUI depends on different situations. If a USB device is tampered into a storage device, or it first identifies itself as other devices like USB keyboard or network adapter and then requests reinitialization as a storage device, TMSUI could deny the unauthorized storage device successfully. But if a USB storage device is fully tampered into other kinds of devices, it is hardly recognized by TMSUI. This test result does not violate our security goal which aims at controlling the access in the exact form of USB storage device. Overall, TMSUI fulfills the design principles of compatibility and security.

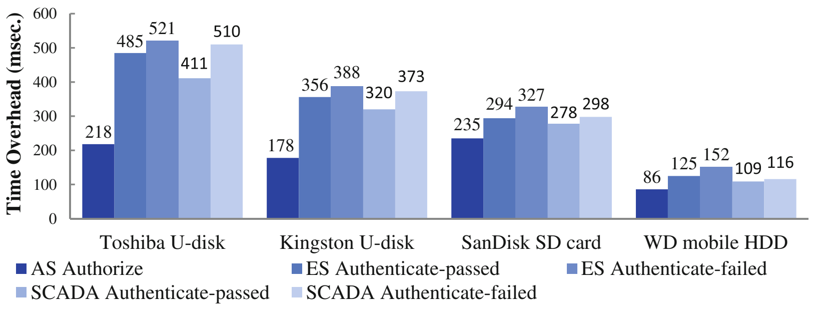

Fig. 3. Time overheads of 4 USB devices executing authorize and authenticate

Furthermore, we evaluate the performance of TMSUI by using four kinds of prevalent USB storage devices in Table 1 to execute Authorize phase and Authenticate phase. The Authenticate is tested on engineer station (ES) and

SCADA server for both passing the authentication and not passing. 10 administrators, 50 authorization items in each USB device and 10 items in *RevokeList* are assumed in the experiment. Every failed authentication result is given after attempting all test branch paths. Figure 3 illustrates the average time overheads of each test case running 20 times. Evidently, all the operations spend less than 600 milliseconds. This does not affect the normal use of USB storage devices and endows TMSUI more security on account of the speediness of ejecting dangerous devices. In practice, the prototype system of TMSUI has been tested in depth and got positive feedback from an automatic control system company and some industrial enterprises.

7 Discussion

In this section, we discuss some practical issues about deploying TMSUI.

Feasibility for Large-Scale ICSs. For large-scale ICSs with many POs, TMSUI operation overheads might overwhelm some absolutely centralized administrators. However, the issue can be solved through a well-designed hierarchy of administrators with the divided rights of jurisdiction. In fact, one or two administrators may only manage a specific region of ICSs, for example just involving a certain manufacturing process. Thus, the number of USB devices used within the region is quite limited for the administrators to focus on handling the TMSUI operations. The hierarchy does not violate the principle of centralization because the USB devices are still under the control of relatively centralized administrators who are also under the higher-level supervision.

Auditability for Administrators. With the help of backup proofs and logs in TMSUI, the administrative hierarchy enables the top-down audits on administrators' behaviors. On the one hand, once any POs are found invaded, the logs could provide clues to correlate them to the malicious USB devices and the corrupt administrators' misconduct. On the other hand, the regular audits may contribute to revealing the administrators' errors or oversights to prevent possible losses.

8 Conclusion

In this paper, we investigate the security issues on USB storage devices used in ICS, and propose a trust management scheme to specially deal with them. With support of security chip, the scheme achieves customizing offline whitelist for authorized USB storage device accessing exactly protected terminals in ICS. The management and control mechanisms prevent data transmission between unknown devices and sensitive terminals. Digital forensic and authorization revocation are enabled. The evaluation on the prototype system shows that our scheme is universal, effective and efficient. Our future work will concern formal proofs related to the security properties of TMSUI, and building trust chain on PO using security chip.

Acknowledgment. We thank the anonymous reviewers for their valuable comments. This work was supported in part by grants from the National Natural Science Foundation of China (No. 91118006, No. 61202414 and No. 61402455) and the National 973 Program of China (No. 2013CB338003).

References

1. BadUSB: http://github.com/adamcaudill/Psychson. Accessed 20 July 2015
2. Havex: https://www.f-secure.com/weblog/archives/00002718.html. Accessed 15 May 2015
3. Mon E.R.: https://en.wikipedia.org/wiki/RavMonE.exe. Accessed 9 June 2015
4. Red Oct.: http://heavy.com/news/2013/01/red-october-virus-cyber-attack/. Accessed 15 May 2015
5. Stuxnet: https://github.com/micrictor/stuxnet. Accessed 10 June 2015
6. USBC: http://www.mu-43.com/threads/20289/. Accessed 9 June 2015
7. Cai, N., Wang, J., Yu, X.: SCADA system security: complexity, history and new developments. In: 6th IEEE International Conference on Industrial Informatics, INDIN, pp. 569–574. IEEE (2008)
8. Carvey, H., Altheide, C.: Tracking USB storage: analysis of windows artifacts generated by USB storage devices. Digital Invest. **2**(2), 94–100 (2005)
9. Colwill, C.: Human factors in information security: the insider threat-who can you trust these days? Inf. Secur. **14**(4), 186–196 (2009). technical report
10. Deroncelé, E.B., Fuentes, A.P., Tejera Hernández, D.C., Cáceres Navarro, H., Fírvida Donestévez, A.A., Febles Parker, M.E.: USB Device Management in GNU/Linux Systems. In: Corral, L., Sillitti, A., Succi, G., Vlasenko, J., Wasserman, A.I. (eds.) OSS 2014. IFIP AICT, vol. 427, pp. 218–225. Springer, Heidelberg (2014)
11. Diwan, S.A., Perumal, S., Fatah, A.J.: Complete security package for USB thumb drive. Comput. Eng. Intell. Syst. **5**(8), 30–37 (2014)
12. Ezell, B.C.: Infrastructure vulnerability assessment model (I-VAM). Risk Anal. **27**(3), 571–583 (2007)
13. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in SCADA networks. Comput. Secur. **25**(7), 498–506 (2006)
14. Keith, S., Suzanne, L., Victoria, P., Marshall, A., Adam, H.: Guide to industrial control systems (ICS) security. NIST Spec. Publ. **800**, 82 (2014)
15. Nohl, K., Kribler, S., Lehl, J.: BadUSB-on accessories that turn evil (2014). https://srlabs.de/badusb. Accessed 15 October 2014
16. Owusu, E., Guajardo, J., McCune, J., Newsome, J., Perrig, A., Vasudevan, A.: OASIS: on achieving a sanctuary for integrity and secrecy on untrusted platforms. In: Proceedings of 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 13–24. ACM (2013)
17. Pham, D.V., Halgamuge, M.N., Syed, A., Mendis, P.: Optimizing windows security features to block malware and hack tools on USB storage devices. In: Progress in Electromagnetics Research Symposium (2010)
18. Sailer, R., Zhang, X., Jaeger, T., Van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: USENIX Security Symposium, vol. 13 (2004)
19. Ten, C.W., Liu, C.C., Govindarasu, M.: Vulnerability assessment of cybersecurity for SCADA systems using attack trees. In: IEEE Power Engineering Society General Meeting, pp. 1–8. IEEE (2007)

20. Tetmeyer, A., Saiedian, H.: Security threats and mitigating risk for USB devices. IEEE Technol. Soc. Mag. **29**(4), 44–49 (2010)
21. China National Vulnerability Database (CNVD: 2014). http://www.cnvd.org.cn. Accessed 20 September 2014
22. China State Password Administration Committee: Functionality and interface specification of cryptographic support platform for trusted computing (2007). http://www.oscca.gov.cn. Accessed 28 November 2013
23. Common Vulnerabilities and Exposures (CVE: 2014). http://www.cve.mitre.org. Accessed 15 September 2014
24. Imation Corporation: IronKey secure USB devices (2014). http://www.ironkey.com/en-US/solutions/protect-against-badusb.html. Accessed 25 November 2014
25. Industrial Control Systems Cyber Emergency Response Team (ICS-CRET): ICS-CERT monitor for October-December 2013 (2013). https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2013.pdf. Accessed 10 October 2014
26. Industrial Control Systems Cyber Emergency Response Team (ICS-CRET): ICS-CERT monitor for January-April 2014 (2014). https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_%20Jan-April2014.pdf. Accessed 20 September 2014
27. Modbus: Modbus application protocol specification v1.1b3 (2012). http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Accessed 10 November 2014
28. NSFOCUS Information Technology Co., Ltd: Research report on ICS and its security (2013). http://www.nsfocus.com. Accessed 5 February 2014
29. NSFOCUS Information Technology Co., Ltd: 2014 ICS security report (2014). http://vdisk.weibo.com/s/r1DAFAovsYVH. Accessed 28 September 2014
30. NSFOCUS Information Technology Co., Ltd: Research and practice on security of industry control system (2014). http://www.nsfocus.com/report/NSFOCUS_ICS_Security_Report_20140311.pdf. Accessed 28 July 2014
31. Trusted Computing Group: TCG specification architecture overview, version 1.4 (2007). http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14/. Accessed 25 October 2013
32. Trusted Computing Group: TPM main specification version 1.2, revision 116 (2011). http://www.trustedcomputinggroup.org/resources/tpm_main_specification. Accessed 25 October 2013
33. Trusted Computing Group: Trusted platform module library, family 2.0 (2014). http://www.trustedcomputinggroup.org/resources/tpm_library_specification. Accessed 10 December 2013
34. Thomas, P., Morris, A.: An investigation into the development of an anti-forensic tool to obscure USB flash drive device information on a Windows XP platform. In: Third International Annual Workshop on WDFIA, pp. 60–66. IEEE (2008)
35. Tovar, E., Vasques, F.: Real-time fieldbus communications using profibus networks. IEEE Trans. Industr. Electron. **46**(6), 1241–1251 (1999)
36. Yang, B., Feng, D., Qin, Y.: A lightweight anonymous mobile shopping scheme based on DAA for trusted mobile platform. In: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 9–17. IEEE (2014)
37. Yang, B., Yang, K., Qin, Y., Zhang, Z., Feng, D.: DAA-TZ: an efficient DAA scheme for mobile devices using ARM trustZone. In: Conti, M., Schunter, M., Askoxylakis, I. (eds.) TRUST 2015. LNCS, vol. 9229, pp. 209–227. Springer, Heidelberg (2015)