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

Bo Yang, Dengguo Feng, Yu Qin, Yingjun Zhang, and Weijin Wang

Trusted Computing and Information Assurance Laboratory
Institute of Software, Chinese Academy of Sciences, Beijing, China
{yangbo,Feng,qin_yu,zhangyingjun,wangweijin}@tca.iscas.ac.cn

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 especially for ICS to provide a complete defense against data transmission 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 the background of application scenarios, TMSUI is designed based on security chip to achieve authoring a certain USB storage device to only access some exact protected terminals in ICS for a particular period of time. The issues about digital forensics and revocation of authorization are discussed. The prototype system is finally implemented and the evaluation on it indicates that TMSUI effectively meets the security goals with high compatibility and good performance.

Keywords: Trust Management, USB Storage Device, Industrial Control System, Industrial Security, Security Chip.

1 Introduction

The simplicity to access all kinds of computers and the easiness to carry around are two of the significant advantages of USB storage devices. USB flash disk, memory card, mobile HDD, smartphone and embedded devices can all have free access to computers as USB storage devices. 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 the chances to attach USB storage devices to computers of ICS. However, USB storage devices are becoming the terrible media to threaten the security of ICS.

By the end of June 2014, totally 549 vulnerabilities of ICS are publicly reported by American Common Vulnerabilities and Exposures (CVE) [16], the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) of US Department Homeland Security [19] and China National Vulnerability Database (CNVD) [14]. From the report[18], 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 virus, worms, spyware and Trojan horse, are able to invade ICS together with USB-based hack tools through USB storage devices [23]. Especially, the exclusive core part of ICS is relatively (or absolutely somewhere) isolated from externally general 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 [22]. 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 [9].

On the other hand, launching social engineering attacks using USB storage devices is considered to cause more damage than malicious codes themselves in ICS [21]. The unauthorized access behavior of USB storage devices taken by a corrupt engineer, an evil staff 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 access ICS are the reasons that lead to these risky situations. Therefore, building up a trust management scheme of USB storage devices can help to keep the hazardous devices away from ICS. To our best knowledge, there is no complete solution specially designed for the architecture of ICS.

In this paper, TMSUI, a trust management scheme combined with related security rules is proposed. The scheme is able to authorize some certain USB storage devices to have limited permission to access specific protected computers such as engineer stations (ES) and supervisory control and data acquisition (SCADA) servers in ICS, while other unauthorized devices are directly ejected by the target computers. Actually, the scheme can be executed flexibly and at a fine granular level. Our contributions can be summarized as follows:

- We design TMSUI expressly for ICS. Based on security chip and digital signature technique, the administrators are guaranteed to have the privilege to authorize USB storage devices, and meanwhile they are held accountable for their respective misconduct. The authorization is time-limited.
- We enable an offline whitelist mechanism. The corresponding revocation strategy is presented in case the authorized devices are stolen.
- According to TMSUI, security rules are given to guide the operations, which helps to establish the complete protection system from both aspects of technology and management.

– We implement a prototype system with full functions of TMSUI and the evaluation shows its good effectiveness with high compatibility and low performance overhead.

The remainder of this paper is organized as follows. Section 2 introduces the related work of ICS security, technology of trusted computing and the research on USB devices. Section 3 provides the system overview. Our TMSUI is specified in Section 4. Section 5 describes the implementation and evaluation of the scheme. Finally, Section 6 concludes the whole paper.

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 [7]. 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 [2]. Infrastructure vulnerability assessment model [6] is conductive to quantify the vulnerabilities in ICS networks. TCG-based integrity measurement architecture [12] provides a method to control the untrusted processes in operating system and establish the trusted execution environment for software in ICS. [3] emphasizes that the application of technology alone will not provide solutions, and human factors can cause the insider threat and even easily destroy the secure environment of ICS. The latest NIST guide to ICS security [8] standardizes both the security technology used in ICS architecture and the ICS-specific security policies that regularize employees' behaviors. Nonetheless, few literatures specially 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 thoroughly.

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 [1]. TC can effectively protect platform from software-based attack from malwares. Using TC for x86-based hardware platform, Trusted Computing Group (TCG) proposes Trusted Platform Module (TPM) [24], while China proposes Trusted Cryptography Module (TCM) [15]. Nowadays, the widely used TPM chip is designed according to TPM v1.2 specification. TCG has already released the latest TPM v2.0 specification, which absorbs some techniques of TCM including SM serial algorithms. TPM has been generally employed to construct trusted execution environment for security-sensitive computers in the similar way of [12]. TPM and TCM are the alternative security chips to build up our TMSUI. The unique endorsement key of the chip fixed on the mainboard can be leveraged to identify the computer of ICS and its tamper-resistant cryptographic algorithm library maintains the secure communication as well as trust relationship between the administrators and the protected computers.

Apart from system and network security, some research pays attention to the security issues of USB storage devices. Sinan et al. design and implement a security platform for USB flash disks [5]. The scheme needs to deep modify the kernel drivers, and only targets at Windows OS and resisting some known malwares. A method of USB device management in Linux is proposed by [4], which concerns the access control to USB device and defends against the attacks from computers. Some configure policies are given by [11] for Windows to block malwares and hack tools from USB devices. But it is considered vulnerable without concrete security technology. [13] 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 [17] are recommended to protect against BadUSB malware using signed firmware. However, the specially-made devices with high cost are impossibly 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 also imposes fine-grained restrictions on USB storage devices.

3 Overview

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

3.1 System Architecture

On the basis of ICS deployment diagram [21], we abstract general ICS architecture and omit some irrelevant components according to the USB storage devices application scenarios. Fig. 1 illustrates the system architecture of our TMSUI. The blue area 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 yellow area is part of process control network which optionally contains ES, 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 well protection from theft through USB storage devices. Thus, these terminals are our protected objects (PO) and equipped with security chip. The red area consists in

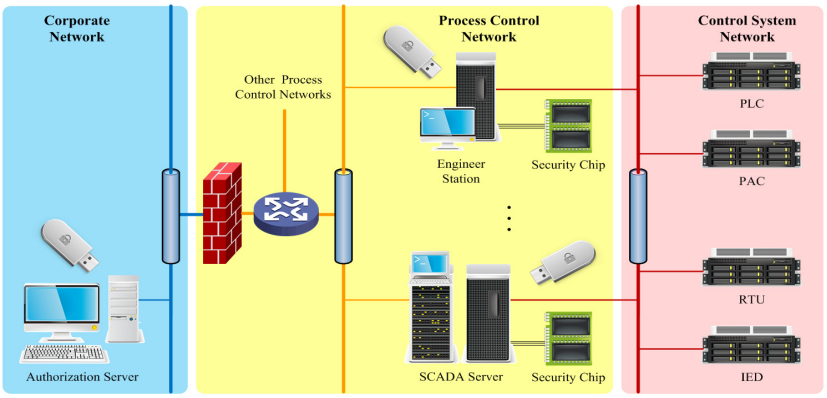

Fig. 1. System architecture of TMSUI.

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 blue and yellow adopt general structure of Ethernet, while the red one often uses filed bus structure or industrial Ethernet with specified protocols such as Modbus [20] and Profibus [26].

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. Actually 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 their own USB storage devices with the similar information of the formal USB devices used inside the ICS. More powerfully, the adversary may untargetedly 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.

We do not consider network and system attack itself, which is beyond the protection scope in this paper. Moreover our technical solution cannot absolutely stop the social engineering attack such as the scene in which the corrupt engineer "steals" his own authorized USB storage device to exactly access the PO which he often operates.

For improving the overall protection of our TMSUI, it is assumed that the communication between AS and PO builds on secure transport protocols such

[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.

as TLS/SSL or VPN. The traditional safeguards for network of ICS and system of computers like IDS, firewall, anti-virus software or isolated execution environment schemes [12,10] are recommended. These are easily and necessarily established and maintained by most ICS. To further enhance security, we compulsively prohibit OS on PO from running USB storage devices automatically and lock the permission to change the OS time. These can be achieved by setting the OS and BIOS on computers.

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 appropriated 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 demand for other special equipment except for security chip. The security chip like TPM/TCM has currently become the standard components of many computers and is easily later equipped at low price in varied ways such as fixing chip via PCI.

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 trust transfer forms the centralized and efficient management and ensures the scheme security.

High Security. The scheme is designed to provide protection with following security properties:

- **Correctness.** The scheme itself fully realizes the security goals and executes without faults that could cause exceptions in ICS.
- **Unforgeability.** The effective authorization whitelist in USB storage devices cannot be created by the adversary along.
- **Tamper Resistance.** Tampering authorization whitelist issued by administrators cannot bring them into effect.
- **Non-repudiation.** The administrators cannot deny their authorizing operation on the exact USB storage devices.
- **Revocability.** The scheme enables revoking the authorization of the authorized USB storage devices and reducing the loss once the devices are stolen.

4 Trust Management Scheme

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

4.1 Executing Process

The proposed scheme is composed of four phases: Setup, Authorize, Authenticate and Revoke. And there are manly four kinds of entities participating in the executing process, including administrator, AS, PO and USB storage device. Setup only need to 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. Fig. 2 shows the executing 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.

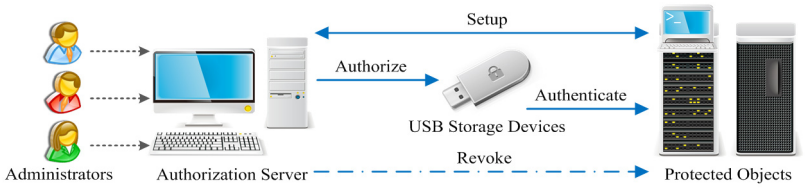

Fig. 2. Executing process of TMSUI.

4.2 Detailed Phases

The scheme takes one USB storage device as example. The physical serious number sn and vendor ID vid of USB storage device could together uniquely identify the USB device [25]. For security chip, it owns a unique endorsement key ek 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 and $H(pswd_\alpha)$. $H()$ represents a hash function. The generating process runs as

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

   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

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

   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

$$epk_\rho \leftarrow Tspi\_TCM\_GetPubEndorsementKey(ValidationData).$$

   $Tspi\_TCM\_GetPubEndorsementKey$ is TCM API for exporting public ek, and $ValidationData$ is the validation information secretly generated inside the chip with the initialization of TCM for purpose of calling TCM API.
4. After receiving epk_ρ, AS queries the original database table for checking the keys legitimacy. Passing the validation, 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.

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, staffs or operators could ask an administrator α to authorize the device. Plugging into the AS, USB storage device is firstly scanned and detected for

ruling out 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. The device uniformly purchased and distributed by enterprise is recommended here (see Sec. 4.3).
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

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

$$(apk_\alpha, ask_\alpha) \leftarrow DEC_{k_\alpha}(blob_\alpha).$$

4. From the user of the USB storage device, α confirms which computer of PO the user is going to access. For the selected computer ρ, the corresponding epk_ρ is found out 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.

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

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 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. Once the authorized USB device behaves maliciously, the related user and administrator could be sought out based on the relevant evidence.

Authenticate. When a USB storage device is taken to one of PO such as ρ, this phase is triggered in 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 and checks the whitelist in the USB storage device. Firstly, exp in each item is compared with the current local time. Secondly, if the item is not out of date, epk in item is compared with ρ's epk_ρ. The equal outcome indicates that the corresponding item is issued exactly for ρ. Thirdly, for the selected item, ρ finds out the corresponding administrators public authorizing key apk_α according to $adminID$ and then uses it to verify the signature σ by calling function $VRFY$ as

$$sdigest \leftarrow H(exp, epk_\rho, sn, vid), \quad result \leftarrow VRFY_{apk_\alpha}(\sigma, sdigest, param).$$

   If $result$ is $true$, the USB storage device is approved for normal use in ρ by the authentication process.
4. In the above step, ρ will eject USB storage device immediately under five situations: no whitelist read out successfully, no item within its validity period, no epk equal to epk_ρ, no applicable apk found out and $false$ assigned to $result$. These also cause the termination of Authenticate phase.

Revoke. This phase enables to revoke some still effective authorizations of the lost USB storage devices.

1. When a staff notices his authorized USB storage device μ missing, he should report the situation to one of administrators.
2. After receiving the report, the administrator finds 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 $RevokeList$. At 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 promptly distribute the new version of $RevokeList$ to all the PO, or distribute it with a certain frequency such as once a week or once a month.

4.3 Security Rules

The technical solution of TMSUI could address most of pivotal security issues of USB storage devices for ICS, except that the authorized device unwittingly

carries malicious codes to access PO. In case this extreme condition and other unpredicted situations happen, we propose the following security rules recommended as a complementary part of TMSUI:

- purchase USB storage devices carefully in bulk, especially after brand selection and security detection,
- manage and keep dedicated USB storage devices centrally in a certain admin department and forbid taking them out of working ground,
- authorize USB storage devices with the period of validity as short as possible,
- scan the USB storage devices regularly and weed out the malicious codes timely,
- maintain and update the firmware of USB storage devices regularly,
- update the protection software in PO regularly.

In reality, some of these security rules have become parts of the basic guidelines in many industrial enterprises.

5 Implementation and Evaluation

This section describes the prototype system and the 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 controlling the deleting and ejecting actions for USB storage 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 implement ENC and DEC using symmetric cryptographic algorithm SMS4, SIG and $VRFY$ using digital signature algorithm SM2, and the hash function using SM3[15]. In the client, these algorithms are directly provided by the security chip. Finally, 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 [4] can help us implement TMSUI also on GNU/Linux system.

To establish the prototype system environment, we apply a DELL OptiPlex 990 as AS. It owns a 3.3GHz Intel i3-2120 processor, 4GB memory and USB 2.0

Table 1. Compatibility of TMSUI for USB storage devices

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

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

Malware	USBC	RavMonE	UsbSpy.a	Nimaya	Stuxnet	Havex	Red Oct.	BadUSB
Defensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Partly

ports, and runs Windows 7 SP1. For PO, we apply two of Lenovo ThinkCentre M8500t equipped with a 3.4GHz Intel i7-4770 processor, 8GB memory and USB 3.0 ports. These two PO respectively run Windows XP SP3 and Windows Sever 2003 SP2, as well as some industrial control software for simulating ES and SCADA server. Moreover, we choose TCM 1.1 by TMC as the security chip fixed in 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.

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, we pick out eight typical malwares including virus, Trojan, worm and BadUSB for the test. Table 2 illustrates the test result. TMSUI is effective to prevent almost all the malwares except BadUSB. If a USB device is tampered into a storage device, TMSUI could deny the unauthorized device successfully. But if a USB storage device is tampered into other devices like USB keyboard or network adapter, it is hardly recognized by TMSUI, which does not essentially violate our design principles. Overall, TMSUI achieves the design goals of compatibility and security.

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 ES and SCADA server for both

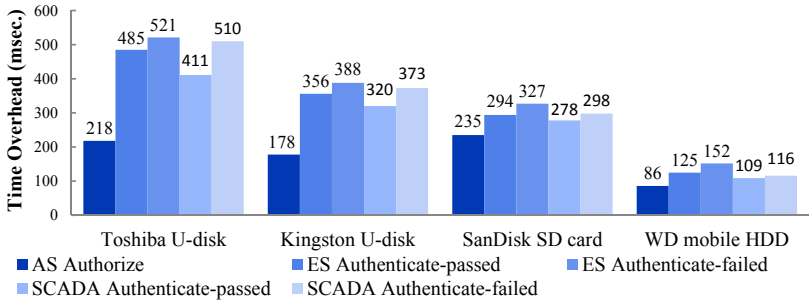

Fig. 3. Time overheads of 4 USB devices executing Authorize and Authentication

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. Fig. 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 usage 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.

6 Conclusion

In this paper, we investigate the security issues on USB storage devices especially used in ICS, and propose a trust management scheme to 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 computers. Digital forensic and authorization revocation are enabled. The evaluation on the prototype system shows that our scheme is universal, effective and efficient. Our next step is to design the function of USB behavior auditing and integrate it into TMSUI for improving defense and trace capability in the matter of malicious acts.

Acknowledgment. We would like to thank the anonymous reviewers. This work was supported in part by grants from the National Natural Science Foundation of China (91118006) and the National 973 Program of China (2013CB338003).

References

1. Bo, Y., Dengguo, F., Yu, Q.: A lightweight anonymous mobile shopping scheme based on DAA for trusted mobile platform. In: 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2014. Beijing. pp. 9–17. IEEE (2014)
2. Cai, N., Wang, J., Yu, X.: SCADA system security: Complexity, history and new developments. In: 6th IEEE International Conference on Industrial Informatics, 2008. INDIN. pp. 569–574. IEEE (2008)
3. Colwill, C.: Human factors in information security: The insider threat–who can you trust these days? Information security technical report 14(4), 186–196 (2009)
4. Deroncelé, E.B., Fuentes, A.P., Hernández, D.C.T., Navarro, H.C., Donestévez, A.A.F., Parker, M.E.F.: Usb device management in GNU/Linux systems. In: Open Source Software: Mobile Open Source Technologies, pp. 218–225. Springer (2014)
5. Diwan, S.A., Perumal, S., Fatah, A.J.: Complete security package for USB thumb drive. Computer Engineering and Intelligent Systems 5(8), 30–37 (2014)
6. Ezell, B.C.: Infrastructure vulnerability assessment model (I-VAM). Risk Analysis 27(3), 571–583 (2007)
7. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in SCADA networks. Computers & Security 25(7), 498–506 (2006)

8. Keith, S., Suzanne, L., Victoria, P., Marshall, A., Adam, H.: Guide to industrial control systems (ICS) security. NIST Special Publication 800, 82 (2014)
9. Nohl, K., Kribler, S., Lehl, J.: BadUSB–on accessories that turn evil. https://srlabs.de/badusb (2014), (last accessed October 15, 2014)
10. 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 the 2013 ACM SIGSAC conference on Computer & communications security. pp. 13–24. ACM (2013)
11. 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)
12. 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, pp. 223–238 (2004)
13. Tetmeyer, A., Saiedian, H.: Security threats and mitigating risk for USB devices. IEEE Technology and Society Magazine. 29(4), 44–49 (2010)
14. China National Vulnerability Database (CNVD): http://www.cnvd.org.cn (2014), (last accessed September 20, 2014)
15. China State Password Administration Committee: Functionality and interface specification of cryptographic support platform for trusted computing. http://www.oscca.gov.cn (2007), (last accessed November 28, 2013)
16. Common Vulnerabilities and Exposures (CVE): http://www.cve.mitre.org (2014), (last accessed September 15, 2014)
17. Imation Corporation: IronKey secure USB devices. http://www.ironkey.com/en-US/solutions/protect-against-badusb.html (2014), (last accessed November 25, 2014)
18. Industrial Control Systems Cyber Emergency Response Team (ICS-CRET): ICS-CERT monitor for October-December 2013. https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2013.pdf (2013), (last accessed October 10, 2014)
19. Industrial Control Systems Cyber Emergency Response Team (ICS-CRET): ICS-CERT monitor for January-April 2014. https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2013.pdf (2014), (last accessed September 20, 2014)
20. Modbus Organization: Modbus application protocol specification v1.1b3. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf (2012), (last accessed November 10, 2014)
21. NSFOCUS Information Technology Co., Ltd: Research report on ICS and its security. http://www.nsfocus.com (2013), (last accessed February 5, 2014)
22. NSFOCUS Information Technology Co., Ltd: 2014 ICS security report. http://vdisk.weibo.com/s/r1DAFAovsYVH (2014), (last accessed September 28, 2014)
23. NSFOCUS Information Technology Co., Ltd: Research and practice on security of industry control system. http://www.nsfocus.com/report/NSFOCUS_ICS_Security_Report_20140311.pdf (2014), (last accessed July 28, 2014)
24. Trusted Computing Group: TCG specification architecture overview. http://www.trustedcomputinggroup.org (2004), (last accessed October 25, 2013)
25. 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, 2008. pp. 60–66. IEEE (2008)
26. Tovar, E., Vasques, F.: Real-time fieldbus communications using Profibus networks. IEEE Transactions on Industrial Electronics 46(6), 1241–1251 (1999)