A Smart Card Web Server in the Web of Things

Lazaros Kyrillidis*, Sheila Cobourne†, Keith Mayes‡ Konstantinos Markantonakis§
Smart Card Centre, Information Security Group
Royal Holloway, University of London
Egham, Surrey, UK, TW20 0EX
Email: *lazaros.kyrillidis.2011@live.rhul.ac.uk, †sheila.cobourne.2008@live.rhul.ac.uk, ‡keith.mayes@rhul.ac.uk,
§k.markantonakis@rhul.ac.uk

***Abstract*—The establishment of the Internet of Things (IoT) is gathering pace. The "things" will be counted in their billions, however interoperability problems may compromise the interconnectivity aspect. Isolated "things" are common and often make use of proprietary communication and security protocols that have not been subject to public scrutiny. By contrast the World Wide Web has well established technology and protocols and so there is interest in the so-called Web of Things (WoT) that would allow the "things" to communicate using standard web protocols. However, with so many readily accessible nodes we considered that the WoT should be underpinned by attack/tamper-resistant security modules that are compatible with the WoT protocols. This paper considers the use of the Smart Card Web Server (SCWS) capability to practically secure the WoT. Finally, the use of a SCWS is extended to provide a means of secure, local Single Sign-On (SSO).**

***Keywords*—SCWS; Smart Card Web Server; IoT; WoT**

I. Introduction

The World Wide Web (WWW) is an almost ubiquitous technology. Since its introduction, it has changed the way we communicate, shop or even socialize and is now considered a vital part of everyday life. In recent years there has also been work towards technology that aims to connect everything in a communications network, the so-called "Internet of Things (IoT)". The IoT promises integration of all things in one large network, whether they be smart phones, computers, sensors, cars, meters, household appliances or tagged objects. The goal is not just communication, but widespread interaction to provide new and/or improved services. However, interoperability problems are currently inhibiting progress towards the IoT vision, resulting in isolated islands of connectivity using proprietary protocols. One approach to this problem is to create a "Web of Things (WoT)", in which standardised web protocols are used to achieve the widespread connectivity required by IoT. Core to the WoT approach is the fact that TCP/IP stacks can be accommodated in devices with relatively poor computing power and memory. Tiny webservers inside WoT devices can be accessed by standard browsers and associated protocols.

However, the WoT approach comes with security concerns. The emphasis to date has been primarily on functionality, with security as an add-on. Using web protocols gives assurance about protocol design, but does not address attacks on implementation security, which is extremely important with so many accessible deployed nodes.

This paper proposes a way to deal with such problems, by leveraging standardisation from mobile communications in the form of the Smart Card Web Server (SCWS), originally intended to provide a tamper-resistant (attack resistant) webserver capability on Subscriber Identity Modules (SIM). For WoT we propose a strongly tamper-resistant SCWS security module based on a JavaCard v3.0 Connected Edition chip, supporting a TCIP/IP protocol stack. Finally, the paper also proposes a way for local WoT Single Sign-On using a smartphone with a SIM card hosting a standardised SCWS.

The rest of the paper is structured as follows: Section II provides a background to relevant work in this area, while Section III introduces our proposal. Section V outlines the main characteristics of the proposed architecture, Section VI provides calculations for the execution times of the protocol, while Section VII considers how it would be used for local WoT Single Sign-On. Section VIII gives a preliminary security analysis of the proposal and also discusses its advantages and disadvantages. Section IX concludes the paper and gives directions for future work.

II. Related Work

The are various definitions for the Internet of Things (IoT) and perhaps the first time that a definition was heard in public was during a presentation at Procter & Gamble [1]. However, if we interpret IoT using our own words, we could say that the IoT is the idea of everyday objects (fridges, ovens, cars, etc.) having network connectivity so that they can communicate with each other. These objects may have a range of functional abilities and be "aware" of their own capabilities and in some cases of their environment, and can act individually (e.g. fridge) or as a group to provide a useful service (e.g. sensors in a house). The objects can range from everyday items with tags and/or sensors to major devices such as cars or houses. Recent surveys estimate the number of these objects to reach many billions by 2020 [2]. The emergence of the IoT is motivating significant research. A useful survey describing the technologies behind the IoT, the main applications that can be derived from its use, as well as concerns and issues can be found in [3].

In more recent years the WoT, a subcategory of the IoT has also drawn the attention from the research community. In [11], [12] and [13] the authors describe how they can integrate devices into the Web using the Representational State Transfer (REST) architecture, giving an even more detailed presentation in [14]. Another presentation on WoT issues including technologies, security, privacy and implementation can be found in [15]. Implementation discussions can also be found in [16] where the authors describe a number of

prototypes they created. In [19] the authors describe the use of a gateway in order to access sensor nodes that cannot use web protocols. In [22] the idea of using web technologies with home appliances for home automation is presented, while in [23] the researchers describe a framework for the WoT, illustrated by a prototype motion detector system.

The above solutions define the idea around the WoT, however none of them addresses the issue of authentication in the WoT realm and how to enable secure communications between entities participating in the WoT. With our current paper we aim to propose an answer to this problem through a Single Sign On (SSO) solution and the use of established protocols like TLS. We acknowledge that TLS is not suitable for sensors and miniature tags, but the devices that we are considering in this paper have significant power to support protocols that require bigger amounts of power and our aim is to research the feasibility of using such protocols in the realm of the WoT.

For this proposal we will use the Smart Card Web Server (SCWS). The SCWS is standardised to run inside the SIM card and is able to serve content to clients running on a reader that interacts with the SIM card (usually the browser on the mobile phone). In this case the SCWS is said to run in "server mode", whereas when the SCWS is accessed by a remote administration entity (most probably in the Mobile Network Operator's (MNO) premises) it is running in "client mode" (for an explanation on SCWS see [27]) . As with any web server, the SCWS's main purpose is to serve static or dynamic web content. The reader can find more information in [28] and [29]. For this paper we are extending the idea of the SCWS by integrating it with objects that participate in the WoT.

III. Proposal Overview

Our proposal is built around the use of strongly tamper-resistant "security modules" embedded in all communicating objects that collectively provide the security foundation of the entire WoT. Fundamental to this is the ability for entities to securely authenticate each other prior to transactions. A security evaluated smart card chip supporting JavaCard v3.0 Connected Edition functionality is used as our security module. We chose this chip, as it can be integrated in an environment where secure access and storage of information is needed, but at the same time is easily accessible to external entities (readers, phones) without the need for proprietary protocols. Communication may take place over TCP/IP and, so standard HTTP clients may be used, notably web browsers. Additionally, Java is one of the most popular programming languages and the required development would be very similar to application development on conventional Java smart cards. To briefly recap we are proposing an attack resistant chip, hosting an SCWS that can serve (via the WoT context), trusted web content via standardised web protocols and is relatively simple to program and manage. For the rest of the paper, these tamper-resistant SCWS chips will be called security modules. For the communication channel we chose to examine Bluetooth Low Energy and WiFi Direct, although other candidates included Near Field Communication (NFC) and cellular. Bluetooth LE is the main candidate as its requirements in power are minimal compared to other technologies.

We have to note that our security modules are in general much more powerful than traditional "things". So, the overall estimates and assumptions of this paper will not apply to an environment where predominantly lightweight sensors/tags are used. In our use case the smart card chips can be attached to an electronic assembly and then be integrated to items of higher value, thus allowing for better performance, safer storage and execution times and easier access from readers. This electronic assembly with the security module can then be attached to items of relatively high value, e.g. a painting or a large box full of items of smaller value, thus making our solution not suitable for use with tags or minimal sensors.

IV. Example Use Cases

An example use case could be a product logistics scenario and in particular the interaction of an employee's smartphone with the security modules within a warehouse. The employee can authenticate to his phone's SIM card (this SCWS will be called "local server" for the rest of the text) and then be able to extend this authentication to other devices with SCWS capability (offered by the security module) in order to access data and functionality spread throughout the local area. We have to note though that the use of a smartphone is not mandatory and any other reader can be used instead - the only requirement is that the reader should be able to communicate over TCP/IP. Another use case could be a logistics warehouse, containing a number of crates, where each of the crates has attached one of these security modules. The modules host all necessary information about the contents of the crates and this information is accessible to only legitimate employees and without the need for internet (or other) connection. A third use case can be when these modules are attached to items of high value, so for example if an item needs to be transferred from one location to the other, the details of the recipient (address, phone, etc.) can be accessible to the transport employee in a secure way, without any further equipment and/or infrastructure needed (like internet connection).

A. Assumptions

For our proposal to work properly we have to consider the following assumptions:

- The SIM cards cannot be directly controlled by any other entity besides the MNO. Thus we assume that the MNO is trusted and that it allows access to the SIM card (for applets that will be used by the SCWS to run inside the local server).
- We do not consider managerial issues in this paper, but rather we anticipate that there is a secure procedure for the development, deployment and revocation of certificates and key pairs on the various phones.
- As mentioned above, we assume that the security modules are installed on items that have a significant value, e.g. a fridge, a painting, etc. or any other expensive item and they are not used with tags or other minimal sensors.
- We assume that the security module receives accurate external time from an external timer found on the electronic assembly.

V. ARCHITECTURE-PROTOCOL

Our tamper-resistant SCWS chips may be deployed to provide authenticated access to sensitive information about the products found inside a warehouse. We rely on the attack resistance of these security modules to provide assurance that the chip's stored data, access control and functionality will remain as intended. Furthermore any additional application dependent functionality/data hosted on the security module is expected to be simple enough to be rigorously security evaluated e.g. against common criteria. Since the security module is going to be serving sensitive information only to trusted parties, there must be a way of authenticating entities requesting to access this information - the solution should be able to fulfill the requirements described below.

1) The local server should authenticate the user[1].
2) The security module should authenticate the local server and vice versa.

A high level diagram of our proposal can be seen in Figure 1.

In principle a user could be authenticated directly to all security modules via a unique password or a PIN that is stored in each of these modules, although this would rapidly become impractical as the number of modules and users grow. Our proposal assumes that each user has a single PIN/password. This PIN/password allows the user to authenticate to the SIM card local server and the authentication result is later used for authenticating with security modules located around the area. A security module will accept the local authentication result, since it trusts the tamper-resistant local server that authenticated the user in the first place. Procedures would be used to ensure that the user PIN/password was chosen, stored and updated in accordance with security best practice. The local server will create an authentication token that will pass over to the security module for the latter to provide access for a specific duration of time. The authentication token will have the following attributes:

- An attribute specifying the level of access granted by the authentication token (represented as *L* in Figures 2 and 3), e.g. administrative or standard access.
- An expiration date/time, represented as *T*.
- A value provided that will identify a unique SIM card, so that the authentication token will not provide access to other SIM cards. This value can be the *Integrated Circuit Card Identifier (ICCID)* that uniquely identifies a SIM card, represented as *I*.

Thus, an authentication token could be represented as the concatenation of the above attributes ($L||T||I$).

The above can be seen in Figures 2 and 3.

We can summarise the authentication process in the following steps (Figure 2 represents steps 1 to 6, while Figure 3 represents steps 7 to 12):

1) The user starts the web client (browser) on the reader and starts negotiating a TLS connection to the local server (this is the first TLS handshake) - for TLS see [30].
2) The local server prompts the user for her password (or PIN). This is the "PasswordRequest" HTML page.
3) The user provides the password/PIN to the local server.
4) The local server calculates the hash of the user's password and compares it to a pre-stored value.
5) The user is notified about the result. This is the "PasswordResult" HTML page.
6) If the password/PIN is correct, the local server creates an authentication token, signs it and stores both inside the SIM card. The signature is created by the private key that is part of the key pair already present in the SIM card where the local server is host and can be later used to verify the integrity and authenticity of the token.
7) The user initiates a wireless connection with the security module using her reader. The two entities negotiate a TLS connection.
8) After the second handshake is finished, the security module asks for the authentication token, from the local server. This is the "AuthTokenRequest".
9) The authentication token and the accompanying signature are transferred to the security module over the established wireless connection.
10) The security module checks the validity of the new authentication token, i.e makes sure that the token has not expired. It also verifies the signature of the token using the public key of the local server that it received during the TLS handshake.
11) If both are legitimate, then the security module grants access to the user to the information stored on it. If not, then the token and the signature are discarded and the user is notified.
12) The requested information is sent to the browser over the established connection.

We have to note, that the first part of the protocol can be avoided in case the token is still valid i.e. the user does not need to provide her PIN/password every time, but rather the process can proceed to the second part directly. This check is taking place inside the SIM card/local server - so the user needs to resubmit the PIN/password only when the token stored inside the SIM card has expired. The trust relationship between the local server and the security module means that the latter will accept the token's validity.

By always submitting the authentication token during the second part of the protocol, we ensure that the security module always gets and checks the freshest available token. Between a successful authentication attempt and a subsequent one, the user may have re-authenticated to the local server (thus expanding the validity of the authentication token) or her authentication level may have changed, so that now she may access less or more information. Furthermore in the case of a lost or stolen reader, the authentication token could be erased from the SIM card/local server, so that the non-authorized entity that may try using it to access a security module, will have to re-authenticate (in order for a new token to be created). However, the malicious entity will not know the PIN/password,

[1]For a more complicated solution there may be mutual authentication between the local server and the user's browser through the exchange of certificates. We will not consider this option for this paper.

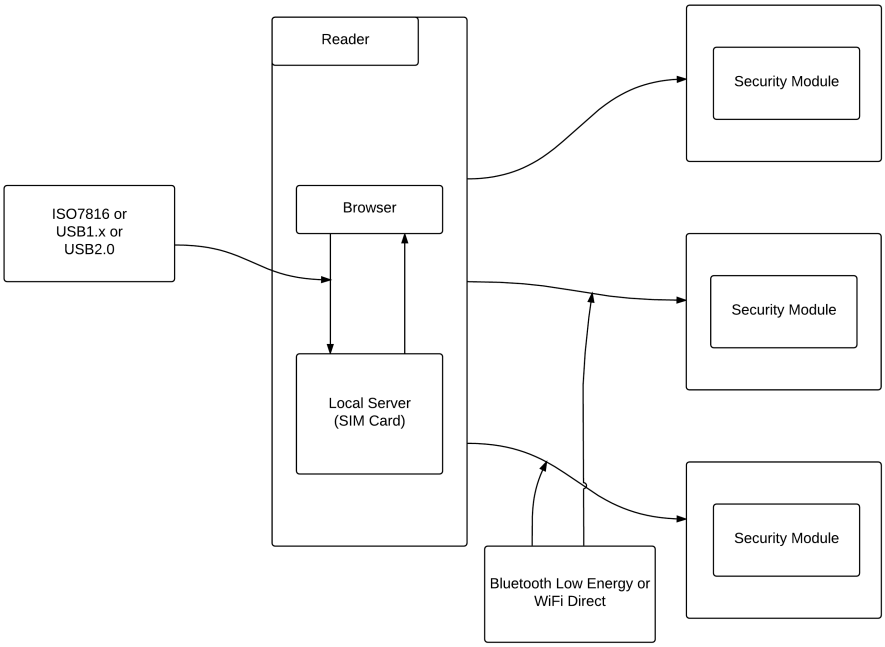

Figure 1: High Level Diagram

thus adding another layer of defense for our solution. If the token had to be stored in each of the security modules that the user requested access to, then the tokens would have to be erased from all these modules to block malicious users.

VI. Process Duration Time Estimation

For the purpose of calculating the overall time needed for the whole process to finish and the user to start receiving information/data in her browser, we wished to consider a modern JavaCard v3.0 Connected Edition. Unfortunately at the time of writing, these cards were not available for the study (thus limiting the practical part of our research) and for our estimated calculations we had to use figures based on JavaCards of the previous generation (v2.2). Our aforementioned calculations are based on results that we gathered during other experiments (for example time needed for an encryption of X bytes to occur), as well as on data that we received from a leading smart card chip manufacturer. This manufacturer provided real figures (for the most powerful of the smart cards that they produce) for a number of actions like RSA signing/verifying, SHA-256 hashing, etc. using a crypto co-processor. For the communication between the browser and the local server, we assumed the use of three different standardised communication channels: conventional ISO 7816, USB1.x and USB2.0 [31]. For the (wireless) communication channel between the local server and the security module we used Bluetooth Low Energy and WiFi Direct.

We separated our calculations into two phases: the first phase is between the browser and the local server (thus, the SCWS inside the SIM card), while the second phase is taking place between the reader/SIM card and the security module. As can be seen in Figures 2 and 3, the messages are presented with the handshake broken down to its basic parts. For each of these messages we calculated the time for the message to be transmitted and/or the time for the basic action to occur, such as an encryption of certain amount of data or the verification of a signature. We did not include the setup time for Bluetooth LE or WiFi direct since setting up a bluetooth connection varies among different technologies and products and even users (i.e. how easy is for a person to setup a bluetooth connection). The amounts of data are based on an average of data that we collected using a network sniffing tool called Wireshark when trying to establish an HTTPs connection with either remote websites or a local website running on a laptop under our control. For all our calculations we assumed the use of the algorithms found in Table I. The figures for each algorithm are based on information that we received from a leading smart card manufacturer for an ARM core running at 30 MHz.

Table I: Algorithm Speeds

Action	Algorithm	Timings
Symmetric Encryption/Decryption	AES-128	0.2ms/block
Asymmetric Signing	RSA-2048	150ms for 2048bits
Asymmetric Verification	RSA-2048	9ms for 2048bits
Hashing	SHA-256	0.5ms/block

The amount of data exchanged for the handshakes, the

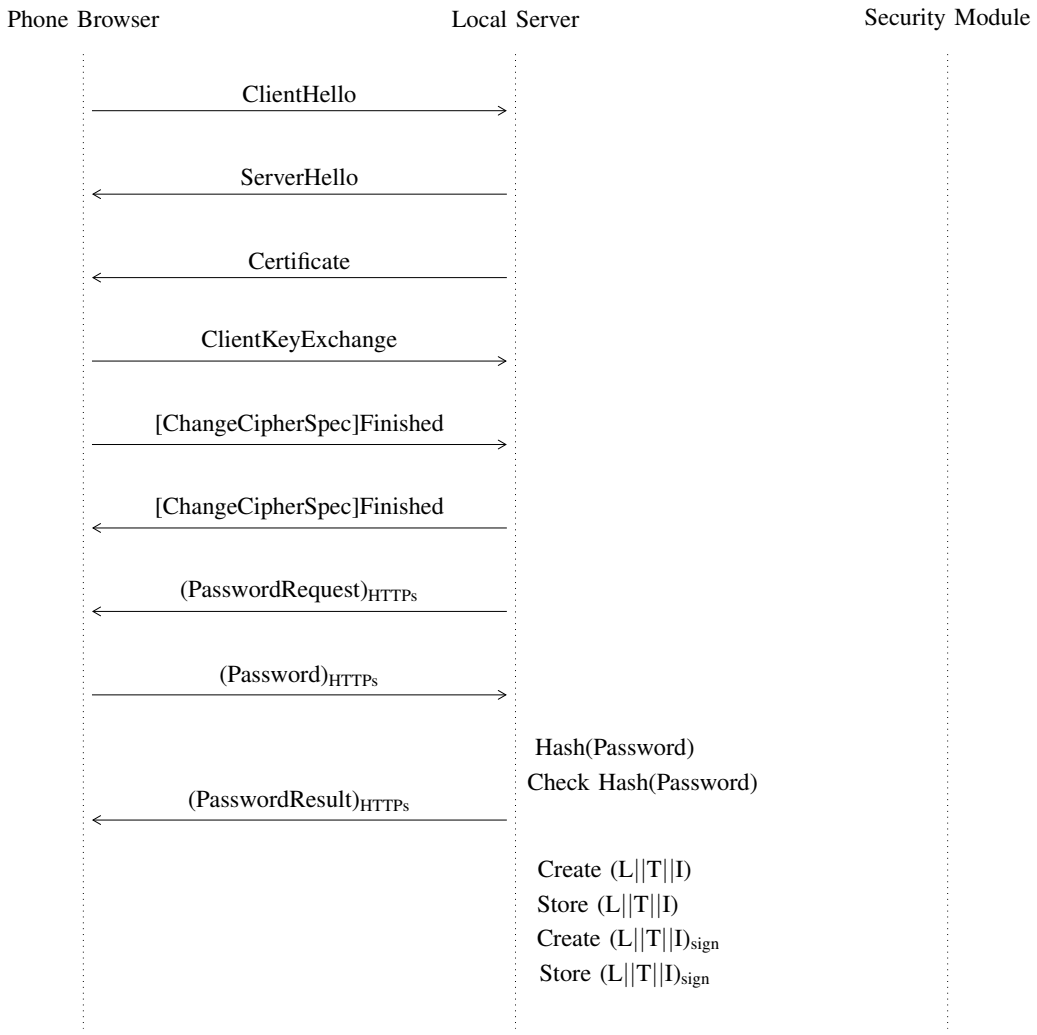

Figure 2: User Authentication and Token Generation

size of password, password hash and authentication token that were processed at each step can be found in Table II - please refer to Figures 2 and 3 for detailed information about the contents of each of the protocol steps. For messages exchanged over HTTPs we included a further overhead for the TLS headers. The page size represents the HTML page that was created by the local server and send to the phone's browser. The HTML page is represented as PasswordRequest which is a page created by the local server for the user to input her password and PasswordReturnResult, where the user is informed about the password checking result.

Before running the calculations we anticipated that the bottleneck for the whole procedure might be the wireless interface between the reader and the security object and (if used) the conventional ISO 7816 interface between the phone's browser and the SIM card. Thus, we concentrated our attention to the comparison of the various available communication channels that we could use, i.e. ISO 7816, USB1x and USB2.0 for the communication between the SIM card and the browser ("wired" communication), and Bluetooth LE and WiFi Direct for the "wireless" interface.

Table II: Amount of Data

Data	Amount (in bytes)
First TLS handshake	5149
Second TLS handshake	9399
Password	9
Password hash	32
Authentication token	36
HTML page (incl. TLS headers)	70

Times for data transit of all messages between the entities can be seen in Table III (calculations in milliseconds). This table does not include the processing time for actions on the data (can be found in Table IV), but the times depicted are the summary transit times for both parts of the protocol (e.g. transit over ISO7816 for the first part of the protocol and transit over Bluetooth LE for the second part). Next to each of the options the reader can find the assumed communication speed

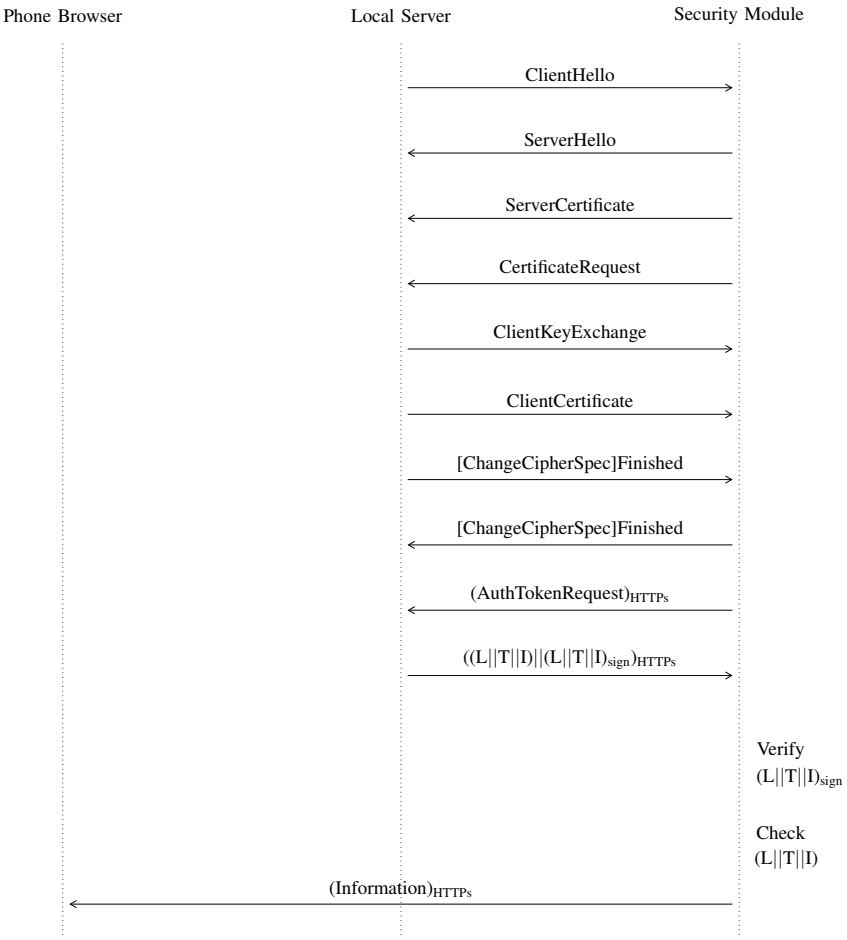

Figure 3: Object Authentication and Access

for each. ISO7816 has a broad range of communication speeds depending on the hardware and the vendor, so we selected an average figure. The same principle was applied to the rest. It must be noted that the theoretical speed for each of the interfaces is larger, but we chose to use a figure representing an average used/found in practice.

Table III: Transit times per communication channel in ms

Wired \ Wireless	Bluetooth LE (0.1 MBit/s)	WiFi Direct (54 MBit/s)
ISO 7816 (0.1 MBit/s)	1967.04	1208.69
USB1.x (12 MBit/s)	769.82	11.47
USB2.0 (80 MBit/s)	761.27	2.92

Table IV depicts the time needed for the most important calculations to finish. This processing includes the TLS handshake with the verification of certificate signatures, hashing, encryption/decryption of data to be transmitted over TLS and authentication token signature creation (we assume that both chips, in the SIM card and the security module, are of equivalent power).

Table IV: Calculations times

Action	Size of data (in bytes)	Time (in ms)
Signature Verification	256	9.00
Data Hashing (per certificate)	1500	12
Auth.Token Signing	256	150
Data Encryption/Decryption (SIM card and Security Module)	52	0.8
Password Hashing (for 9 bytes password)	32	0.5

Table V depicts the overall time for each of the phases, i.e. it is the summary of all calculations to occur.

Table V: Total calculation times per phase

Phase	Time (in ms)
User Authentication and Token Generation	216.40
Object Authentication and Access	143.60

Table VI gives the overall time needed for the whole process to conclude. Our estimated calculations show that the communication speed is dominating the duration of the process (compared to the processing), especially when the combination is for ISO 7816 along with Bluetooth LE (for the wireless interface between the security module and the reader).

We can see in Table VI that the combination of Bluetooth LE with ISO 7816 is almost 7 times slower than the equivalent combination of WiFi direct and USB2.0.

Table VI: Total times in ms

Wireless / Wired	Bluetooth LE	WiFi Direct
ISO 7816	2327.04	1568.69
USB1.x	1129.82	371.47
USB2.0	1121.27	362.92

VII. Local Single Sign-On

At the beginning of the paper we discussed about how our proposal provides an SSO solution that will allow users to authenticate in a de-centralised fashion in their phones/readers and then be able to access information found on security modules. We now describe how this can be accomplished.

The idea of Single Sign-On (SSO) is not new: in fact it is a well established technology and there are many, free and commercial, SSO product solutions. While products have different features, the basic idea behind SSO is that a user authenticates to just a single central server, which then handles the user's authentication to other participating servers. One of the most prominent of these solutions is OpenID [32] that is being used by a large number of websites and most recently, social media solutions like Facebook Connect allow users to authenticate anywhere using their social media account [33]. One of the most important protocols for SSO is the Kerberos protocol developed by MIT [34]. Kerberos is used with many solutions/platforms, the most prominent of which are the Windows OSes developed by Microsoft [35]. In a nutshell, Kerberos allows users and systems to obtain a "ticket" from a Kerberos Key Distribution Center (KDC) and use this ticket as their credentials when required to access other entities within the network. These entities trust the Kerberos ticket credentials and so allow access from the users/systems that were previously authenticated by the KDC.

Our proposal uses ticket credentials, but is otherwise a different approach. The user is now the "owner" of the authentication server, within the SCWS/local server of the SIM card. The credentials of the user are stored inside the SIM card (which is a tamper resistant device). The user authenticates to the local server, and the SCWS then creates a token that the user can use anytime to access a security module. An advantage of this approach is that there is no centralised equivalent to the KDC, so no single point of failure or attack target for Denial of service (DoS). Of course, there should be a general trust relationship between all the entities used in the system and this can be achieved by a conventional X.509 certificate approach. Once the local server and the security module mutually authenticate, the security module retrieves the token stored in the SIM card and the user is authenticated if the token is valid and fresh.

VIII. Security analysis

A. Advantages

We based our security module on a smart card chip because such chips can be strongly tamper/attack resistant. Modern high-quality chips will have effective defenses against physical probing/modification, side-channel and fault attacks and can be independently evaluated against common criteria. In short, providing weak devices are avoided, we expect the module to protect the integrity and access of sensitive stored data and to function as per design/expectation, even when subjected to aggressive attack.

In any case no sensitive data is available without authentication and no sensitive transmissions are in clear. The same levels of protection are offered by the local server.

One of the biggest advantages in favour of our proposal is achieving interoperability, without compromising on the tamper-resistant security of the solution. All involved entities communicate over TCP/IP channels, thus making them accessible from any platform that supports this protocol suite i.e. practically everything in the modern Internet. An attack resistant web server stored inside a security module (chip) implementing a TCP/IP stack means that is easily accessible using standardised protocols and can be incorporated within an WoT architecture that requires all the involved entities to be accessible over "web" standards. The lack of need for special protocols and proprietary communication methods allows for easier integration in any environment, and application development is straightforward for anyone familiar with Java and/or smart card programming. The attack resistance of the security module and the equivalent attributes of the SIM card (where the initial authentication is taking place), protects the data and functional implementation (contained within the chip). Logical attacks (against the design) cannot be avoided, but the small attack surface of the web servers and the applets involved, provide a certain assurance level that the programmers can develop robust and security verified code. Another important factor is that all communication is taking place over secure channels and more specifically using HTTPs. Attacks against HTTPs exist [36], however the vast majority of web servers throughout the Internet rely on HTTPs to secure communication to and from them. Until proven otherwise, HTTPs is still the most secure way to browse the Internet and as such should be acceptable for WoT.

Another potentially important factor (which we consider a strong privacy advantage of our proposal) is that the user's password/PIN stays within the local device and is checked by the local server (within the trusted SIM card). The PIN is exposed to the browser (when it is submitted by the user), however even if malware captures the password/PIN, a malicious user also has to be in possession of the exact SIM card in order for her to use it (because the password/PIN can be used with one SIM card only as it is stored in it and checked by it). So, even if an attacker steals the password/PIN, she will need not be able to use it effectively, unless she is in possession of the SIM card as well. Also, if the attacker steals the authentication token, she will not be able to use it, since the authentication token can be used by one SIM card only (identified by the ICCID). Furthermore, the attacker needs to be in possession of a X.509 certificate installed within the

"malicious" SIM card; this certificate must belong to the CA authority that governs the entire system.

The security modules contain all the necessary information about the product(s) that the user wants to access. Thus, a product can provide this information when tapped by an authentic user/phone, and without the need for on-line communication. This stand-alone ability of the chip is very important when an Internet connection is not present. Finally, a web enabled security module has enough processing power to run applets and process data that it stores/receives. Other WoT entities like sensors possess limited (if any) processing power and in most cases they are just simple conduits pushing forward information to more capable entities.

In summary we can say that the main advantages of this proposal are:

1) Interoperability using standardised/proven open protocols.
2) Attack resistance using proven attack-resistant devices.
3) Communication exclusively over secure channels via HTTPs.
4) Web protocols that allow the entities in this proposal to seamlessly form part of the WoT.
5) Stand-alone entities that may provide information without the need for a back end server/database nor a connection to the Internet.
6) De-centralised Single Sign On, thus avoiding single points of failure.
7) PIN/passwords and authentication tokens are useless without the SIM storing them. Thus the attacker has to be in possession of the PIN/password or authentication token plus the SIM card in question, in order for her to get access to a security module.

B. Disadvantages

The importance of mutual authentication cannot be underestimated, however installing and maintaining a number of certificates with all the accompanying tasks (updating, revoking) could prove cumbersome. We anticipate that a process of updating the certificates will be down to policies mandated by the individual case/scenario with the attack resistance of the chips permitting a longer period of certificate validity, depending on the value of the information stored on the chips.

Another fact is the cost of the security modules. An assembly with an advanced smart card chip may be too expensive for use with "cheap" tags/things. So, for our scenario we assumed the use of these objects with items of generally higher value. For example, an assembly with a security module can be attached on a painting, on a fridge or on a palette containing a large number of items. It can then store and protect information about these items (information for the painting or the contents of a box/palette) in a tamper-resistant way protecting it from malicious entities.

IX. Conclusion-Future Work

The proposal described in this paper uses a Smart Card Web Server both in an attack-resistant SIM card (for which it was initially standardised), but also in a similarly protected smart card chip (security module) embedded within an electronic assembly. The two entities communicate over a wireless communication channel mutually authenticating each other. This way the web server running inside the security module accepts the authentication results for the user that was previously authenticated from the SIM card local server. This SSO solution has the advantage of avoiding a single centralised SSO server as it distributes the authentication process over a number of trusted local servers found inside SIM cards.

The entire solution works even when there is no network connectivity available, as everything is happening offline. The user still needs to remember a password/PIN, although capture of this credential is of little use without the corresponding SIM card.

This project is on-going and future work will involve a formal analysis of the protocol and the development of a first working prototype to assess the comparative performance of protocol implementation options. Future work can also include an analysis of managerial concerns (mentioned in the assumptions) of this proposal and investigate the feasibility of direct access and authentication of the user to the security modules, without the need to previously authenticate the user to the local server/SCWS. Finally a future paper could analyse in much bigger detail one of the use cases presented previously, to further assess the feasibility of our solution.

References

[1] K. Ashton. (2009, June) That 'internet of things' thing. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[2] Gartner. The internet of things enables digital business. [Online]. Available: http://www.gartner.com/technology/research/internet-of-things/

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] M. Kranz, P. Holleis, and A. Schmidt, "Embedded interaction: Interacting with the internet of things," *Internet Computing, IEEE*, vol. 14, no. 2, pp. 46–53, 2010.

[5] S. Haller, S. Karnouskos, and C. Schroth, *The internet of things in an enterprise context*. Springer, 2009.

[6] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[7] F. Mattern and C. Floerkemeier, "From the internet of computers to the internet of things," in *From active data management to event-based systems and more*. Springer, 2010, pp. 242–259.

[8] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the internet of things," in *The Internet of Things*. Springer, 2010, pp. 389–395.

[9] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[10] R. H. Weber, "Internet of things–new security and privacy challenges," *Computer Law & Security Review*, vol. 26, no. 1, pp. 23–30, 2010.

[11] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "The web of things: interconnecting devices with high usability and performance," in *Embedded Software and Systems, 2009. ICESS'09. International Conference on*. IEEE, 2009, pp. 323–330.

[12] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, 2009, p. 15.

[13] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[14] D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the internet of things*. Springer Science & Business Media, 2011, ch. 5, pp. 97–129.

[15] D. Zeng, S. Guo, and Z. Cheng, "The web of things: A survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[16] D. Guinard, V. M. Trifa, and E. Wilde, *Architecting a mashable open world wide web of things*. ETH, Department of Computer Science, 2010.

[17] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "Web of things: Description, discovery and integration," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. IEEE, 2011, pp. 9–15.

[18] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber–physical systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 9, pp. 905–923, 2011.

[19] V. Trifa, S. Wieland, D. Guinard, and T. M. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," *Submitted to DCOSS*, 2009.

[20] B. Christophe, V. Verdot, and V. Toubiana, "Searching the'web of things'," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 308–315.

[21] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010, pp. 702–707.

[22] M. Kovatsch, M. Weiss, and D. Guinard, "Embedding internet technology for home automation," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. IEEE, 2010, pp. 1–8.

[23] B. Ostermaier, F. Schlup, and K. Romer, "Webplug: A framework for the web of things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010, pp. 690–695.

[24] B. Traversat, M. Abdelaziz, D. Doolin, M. Duigou, J.-C. Hugly, and E. Pouyoul, "Project jxta-c: enabling a web of things," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE, 2003, pp. 9–pp.

[25] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth *et al.*, "Spitfire: toward a semantic web of things," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 40–48, 2011.

[26] N. Zhong, J. H. Ma, R. H. Huang, J. M. Liu, Y. Y. Yao, Y. X. Zhang, and J. H. Chen, "Research challenges and perspectives on wisdom web of things (w2t)," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 862–882, 2013.

[27] L. Kyrillidis, K. Mayes, B. Chazalet, and K. Markantonakis, "Card-present transactions on the internet using the smart card web server," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*. IEEE, 2013, pp. 611–619.

[28] OMA, *Smartcard-Web-Server Approved Version 1.2.1 OMA-TS-Smartcard_Web_Server-V1_2_1-20130913-A*, Open Mobile Alliance Std., 13 Sep 2013.

[29] *ETSI TS 102 588 V9.1.0 (2011-01) Smart Cards; Application invocation Application Programming Interface (API) by a UICC webserver for Java Card platform; (Release 9)*, Std.

[30] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[31] *ETSI TS 102 600 V7.2.0 (2008-06) Smart Cards; UICC-Terminal interface; Characteristics of the USB interface (Release 7)*, Std.

[32] OpenID. [Online]. Available: http://openid.net/

[33] Facebook. (2008, May) Announcing facebook connect. [Online]. Available: https://developers.facebook.com/blog/post/2008/05/09/announcing-facebook-connect/

[34] Kerberos: The network authentication protocol. [Online]. Available: http://web.mit.edu/kerberos/

[35] Microsoft. Microsoft kerberos. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa378747%28v=vs.85%29.aspx

[36] Tracking the freak attack. [Online]. Available: https://freakattack.com/