CIT. Journal of Computing and Information Technology, Vol. 26, No. 4, December 2018, 235–250
doi: 10.20532/cit.2018.1004174

Multi-point Security by a Multiplatform-compatible Multifunctional Authentication and Encryption Board

Ravdeep Singh Boparai, Anastasios Alexandridis and Zeljko Zilic

McGill University, Montreal, Canada

Securing the access in networks is a first-order concern that only gains importance with the advent of Internet of Things (IoT). In this paper, a security system is presented for password-free access over the secured link. It makes the connection faster than manual authentication and facilitates Machine-to-Machine (M2M) secure interactions, as required for IoT. The authentication procedure includes the exchange of certificate and challenge/response pairs, which are stored and computed in an external security coprocessor. The system enforces the authentication protocol, includes error detection, and handles multiple devices according to their Operating Systems (OS) through their connections/disconnections. It also performs encryption, if necessary. It is applicable on application level for devices, including IoT based devices, sensors, Android, and iOS-based smartphones. The devices that have the correct certificate and can solve the challenge can connect to the network linked with the security system. The system security is hardened because the sensitive authentication elements such as keys, certificates, and challenge responses are invisible to users and are exchanged only using strong hashing algorithms that are irreversible. The proposed hardware security system can augment any supporting network, converting the entire insecure network into a secured one, as well as retrofit existing insecure Bluetooth devices for secure access. The system incurs low overhead in time and energy by performing security operations in an ASIC coprocessor, and can be shared to secure access to multiple devices, which reduces both energy and cost.

ACM CCS (2012) Classification: Security and privacy → Security in hardware → Embedded systems security

Computer systems organization → Embedded and cyber-physical systems → Sensor networks

Keywords: multiplatform, secure channel, IoT, wireless, multiple devices, Bluetooth

1. Introduction

In 2020, billions of Internet of Thing (IoT) devices will be available [1], turning this world into a hub of wireless networks consisting of "things" such as sensors, home appliances, and computing devices. In turn, security becomes an important factor, as any such thing becomes accessible and vulnerable to attacks.

The existing methods, such as authentication by passwords can mitigate the issues only partially. The passwords can be cracked through computation and by means of deceiving the users. Passwords are also tedious as they must be remembered and entered by users, making them time-consuming in most cases. More importantly, the IoT world is becoming essentially a Machine-to-Machine (M2M) domain, in which intervention by humans is prohibitively expensive, if not impossible.

In certain critical cases, additional devices can produce authentication tokens, often in a way applicable to the specific authentication scenario. In the past decade, two-factor authentication systems have been introduced. They employ an additional personal device [2], such as a phone or a tablet, to produce an authentication token. The scheme is fairly simple to use; however, it also comes with disadvantages. The use of passwords is still one of the problems, and the time it takes to authenticate becomes worse than when using the passwords alone.

To address these issues, a hardware-based security system that can authenticate and encrypt the communication between multiple devices or between devices and peripherals is proposed. The proposed system performs password-free authentication, which increases the authentication speed and eliminates human errors. It is compatible across different computing platforms and is capable of securing peripherals such as sensors that are either wireless or wired. Similar to two-factor authentication, there are two steps to execute. With the system, a protocol was designed that allows the authentication for accessing multiple devices and encrypting the raw data of peripherals or sensors, in a way compatible with connection-oriented or connectionless communications. Hence the channel between user devices and security system is authenticated and encrypted. The details of packet formation, handling the timeout, acknowledgments, sequences, size, payloads, and checksums, are all carefully addressed in the proposed solution as well.

The hardware-based security system is designed and deployed for both leading smartphone platforms that can perform authentication compatible with an X.509v3 certificate-based authentication, and a challenge-response exchange based on hash-based message authentication code (HMAC) – Secure Hash Algorithm (SHA256) was used as the underlying hashing algorithm, which is deemed secure for the time being [3]. For encryption, an Elliptic Curve Cryptography (ECC) is used, with dynamically generated asymmetric keys used to establish a secure connection. Using this connection, the system exchanges a shared key used to perform Advanced Encryption Standard (AES) encryption, as specified by the National Institute of Standards and Technology (NIST).

The remainder of this paper is outlined as follows. Section 2 presents the previous related work on hardware-based security, as well as authentication and encryption of IoT based systems. In Section 3, the proposed system architecture and design, as well as the parts that it is composed of, can be found, whereas in Section 4 the core of the system, a coprocessor, is outlined. System advantages and applications are given in Section 5. The implementation and evaluation of the security system can be seen in Section 6. The last section concludes the paper and discusses any potential future work for the system.

2. Related Work

Wireless sensor networks (WSN) for IoT require strong cryptography to protect data from the attacks in which intruders could be anywhere in the world. The most common methods of protection will be outlined here. The *Two-factor authentication system* developed by Das [4] needs the user to login during the Registration phase, followed by Authentication phase that employs most commonly another consumer electronic device. The *Controller-based* security includes a security manager, a link and network layer security – there is additional need for the security at application layer. TinyPK, an authentication protocol based on RSA and Diffie-Hellman algorithms was proposed by Watro *et al.* [5]. The credentials are stored in the device, therefore they can be extracted using a stolen verifier attack. The use of user login exposes the keys as well as makes it vulnerable to the *man-in-the-middle* (MITM) attack, and significantly slows down the authentication procedure. A protocol was proposed by Shi *et al.* [6] for WSNs that uses elliptic curve cryptography for providing a secure communication between sensor devices and user devices. The solution presented by Salman *et al.* in [7] uses the Montgomery multiplier hardware and pairing software which requires additional computation and power for pairing-based cryptography.

Gao and Gang [8] used fingerprinting, digital watermarking along with device authentication, and encryption, which increases the security using hardware, but at the same time needs additional (*e.g.*, biometric) hardware devices. Another approach by Lesjak *et al.* in [9] used hardware-rooted snapshots for transparent and secure communication for Industrial IoT. It uses schemes such as Broker-based messaging infrastructure and hybrid encryption.

The security attacks can take many forms, such as physical, side-channel, network, or software attacks. Certain security frameworks were built to provide security against some of such attacks [10]. Wireless systems have been more vulnerable as the secret encryption key can be extracted from the packets through statistical analysis. By using attack methods such as side-channel attacks, various approaches were made by Zhen *et al.* [11]. Blockchain-based frameworks were also proposed that provide a decentralized security system having additional features, such as fault tolerance and scalability by Varshney *et al.* [12].

The YubiKey commercial product implements "Security keys" as a second factor device to protect the user from MITM, or phishing, attacks [13]. The device acts as a hardware-based login system that has been deployed by Google, Dropbox and many more. It follows the Universal Second Factor (U2F) open standard for the Security key protocol, which uses key pairs explicitly tied to systems in order to authenticate. YubiKey also supports simple passwords to automate input, and one-time challenge or time-based responses. Once registered and configured for two-factor authentication, the YubiKey is inserted into USB and a button on the key is pressed, or it is simply tapped against Near-field communication (NFC) enabled Android smartphones to perform authentication. In the software, the YubiKey first registers in which a pair of Security keys is generated, and public key is returned which is now linked to user account. There is also a key handle, which ties the key to a specific origin, for example a server. When authenticating, the key will match the key handle to the correct pair of keys, and send a signature created using the private key, which is verified by the server to authenticate the user [14].

However, the YubiKey is limited to only one type of interface, a USB connector, with NFC also supported in one model. Moreover, YubiKey can perform authentication on only a single device at any given moment, with physical action required to authenticate a different device (that is, connecting it to the other device). Further, the YubiKey does not support securing the other concurrently used devices and peripherals, such as sensors. It requires a fully functional system in place that both supports U2F and features device logic that can handle data formatting and communication. Last but not least, even though YubiKey initially featured open-source components, it has transitioned to closed-source in the latest models, which cannot be reviewed independently for security vulnerabilities [15]. This approach that reintroduces the 'security through obscurity' concept eventually led to a flaw in the underlying cryptographic library, which generated primes with low entropy. A practical factorization attack (ROCA vulnerability, which stands for Return Of Coppersmith Attack), based on Coppersmith's factorization was discovered by a research team, which allows attackers to compute a private key from a public key in a realistic amount of time [16], [17], [18]. For reference, a 512 bit key generated with a low entropy prime can be factorized in less than two CPU hours, whereas a 1024 bit key can take less than 100 CPU days, both very feasible with rented cloud CPU instances, whereas a 2048 bit key could take just over 140 CPU years, a rather large but still not infeasible requirement [18].

USB Type-C supports authentication through the C-AUTH specification. C-AUTH is an authentication specification standard that provides one-way authentication to authenticate USB devices, power supplies, and cables. Authentication is performed with the use of the certificates which are signed using private keys and hashed for means of verifying them by the authenticating party. The cryptographic methods used are widely available, such as X.509v3 certificate format and SHA256 hashing algorithm. Certificate chains may be used if needed, such as a device certificate that is being signed by the manufacturer, who relies on a root certificate signed by a trusted certificate authority [19]. C-AUTH allows only a single device authentication at a time.

In the Web software world, the W3C consortium is introducing a Web Authentication (WebAuthn) API, which is an add-on for web browsers to generate public key-based credentials which can be accessed by only the origin of key [20]. The WebAuthn performs signature and attestation of a key with the relying party. These are the two steps followed:

(i) registration, in which the authenticator generates the key using its user account and associates with the reply party, and

(ii) authentication, in which the key is offered by the replying party to the authenticator.

The device is registered by signing in an account or creating a new one in a website opened in the browser; the user can add login methods such as passwords/PINs or biometric. The WebAuthn can establish a connection between web browser and server systems, but the connection between sensor and user device has not been explored. Furthermore, the WebAuthn needs a browser whereas IoT based systems usually interface at lower protocol layers, the authentication system should be able to run over the firmware. The proposed system allows in the core of the sensor interfacing hence, the data along the processing pathway can be kept more secure. Further, the authentication system is in the firmware of the control module, making it invisible or inaccessible to user.

The security systems studied and reviewed above require human intervention or are limited to a particular network, either Internet or WSN. The IoT not only connects computers but also connects smartphones with peripherals or sensors. The system should be capable of securing any kind of platform such as Android, iOS, Windows; therefore, it has to be above the OS, at the application layer, while at the same time the authentication procedure should be automatic. If no human intervention is required, then the process is invisible and fast, which protects the system from several types of attacks. It should have a low overhead, as well as it should work with low-powered devices.

To the best of our knowledge, there is no other hardware system providing simultaneous multi-point security between a wide range of devices and technologies, and without human input. YubiKey for instance offers some variety in technologies (USB, NFC), however it is point to point and requires human input [14]. C-AUTH on the other hand is more automated and supports M2M, however it is exclusive to USB-C and also point to point [19]. Our proposed system can act between multiple devices, including connectivity with sensor nodes that often appear in IoT. The presented system performs the authentication, controls the connectivity, and builds an IoT based secured network, surpassing the existing systems that deal with only one device.

3. System Architecture

In this section, the system architecture of a device designed to facilitate secure connection of multiple types of devices in IoT setups is described.

3.1. System overview

The proposed security system is composed of several modules. The main part is the security coprocessor module, which is a "coprocessor", that offers authentication and encryption. The system is also composed of a communication module, which enables the different systems to communicate with each other, while it follows the defined protocol. In addition, there is a control module that controls all modules and enables the system to function. It is responsible for the information exchanged, the protocol being followed, and performs authentication and encryption as required, by controlling the coprocessor.

Finally, a dedicated application is installed on the user device based on the platform such as iOS, Android or Windows. It performs the protocol specific steps with the security system. Figure 1 presents the architecture of the system.

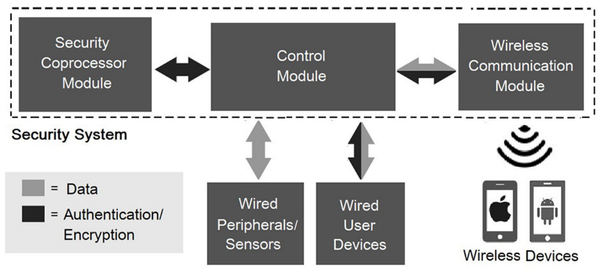

Security Coprocessor Module
Control Module
Wireless Communication Module
Security System
= Data
= Authentication/ Encryption
Wired Peripherals/ Sensors
Wired User Devices
Wireless Devices

Figure 1. System architecture.

For the purpose of clarifying, the practical implementation of the system comprises the control module that is a microcontroller, the security coprocessor module is simply a coprocessor, peripherals are devices whose data is to be secured connected to microcontroller via I2C & UART such as sensors, and the communication module is Bluegiga Bluetooth chip. USB based user systems are wired user devices and wireless devices are the user smartphones including wireless sensors. For consistency purposes, the modules will be referred to with their specific names rather than the abstracted module names, unless necessary. The entire arrangement is termed as a security system that can connect to any device, node, or peripheral that needs data authentication and encryption features. It can be used in several use cases and configurations, such as authenticating peripherals for devices, devices for devices, or even devices for different systems, on either one or both ends. If the system authenticating another system is trusted, then it may not be necessary to authenticate on both ends. In any case, an end to end encryption will be performed if necessary.

3.2. Coprocessor Module

The coprocessor found in the system can authenticate and encrypt communications [21]. The authentication is performed in two steps. First, an X.509v3 certificate is sent to the authenticating device from the coprocessor. The authenticating device can verify the certificate owner by checking the certificate, and verifying that the signer of the certificate is an entity that is trusted by both. This concludes the first step.

For the second step of the authentication, a randomly generated challenge is submitted by the authenticating device. The challenge, together with a secret key known to both parties, undergoes a hash-based message authentication code (HMAC) procedure, based on the SHA256 hashing algorithm, which produces an irreversible response.

The combination of these two steps ensures the device or peripheral is secure and can be trusted, as successful authentication may only be performed by the genuine device or peripheral that is connected through the security system. Following authentication, if the information exchanged between devices and peripherals needs to remain private, the coprocessor can additionally encrypt it.

3.3. Communication Module

The communication module in the system can support various wired interfaces, *e.g.*, Universal Serial Bus (USB) or Universal Asynchronous Receiver-Transmitter (UART), and wireless links, such as Bluetooth, between wireless devices and control module. The communication module includes any hardware that is necessary for the communications, such as UART controllers or Bluetooth radios, and extends into the control module. For the purpose of evaluation, a wireless device with Bluetooth functionality (Bluegiga) was used in conjunction with the UART interface in the microcontroller. Typically, the security system will connect to one or more devices wirelessly and, on the other side, it connects with peripherals or sensors and forwards traffic between each other, by forming packets for the defined protocol, and does all the authentication and encryption.

The defined protocol mandates that communication may take place over any existing link that can send data in a raw form, be it in one or multiple packets, as long as all the data is preserved and reaches the destination. It will construct packets that must contain a header, and optionally a payload, whereas both the header and payload need to be followed by a checksum of the bytes, for error detection purpose. The header of the packet contains information such as packet length, or sequence and acknowledgement numbers. The payload of the packet is optional, for example, the packet may be sent only to acknowledge and needn't contain one. The payload itself contains its own header, which features information such as message type, and may contain one or multiple sub-packets.

3.4. Control Module

The microcontroller is responsible for the system operation, and ensures that everything goes smoothly. The control module is an STM32L432 microcontroller which has all the required interfaces (UART, I2C) to communicate with the communication module, coprocessor, external nonvolatile memory and other interfaces. It performs authentication and encryption when and as needed, by communicating with the coprocessor. It follows the communication protocol and forms packets which it then orders the communication module to transmit, or receives packets from it, which it then decapsulates and processes. It is also responsible to start the systems operating and communicating. The microcontroller follows a sequence, from power up to communicating meaningful information between devices. The sequence of the commands sent and received via the microcontroller to the user device and coprocessor are shown in Figure 2(a) and 2(b). The sequence of steps is as follows:

1. Initialize system and check with all the peripherals for their successful boot-up.
2. Initialize communication module: Start interfaces, perform configurations such as setting transmission power on wireless radios, or data packet size configurations.
3. Check wired and wireless interfaces for incoming connections.

4. Advertise to allow new connections.
5. Connect to the available paired devices and peripherals.
6. Detect devices/peripheral types requiring authentication and those that need to authenticate.
7. Initialize the protocol and configure it for the connected devices and peripherals.
8. Initialize authentication procedure.
9. Certificate exchange and validation.
10. Challenge exchange and response verification.
11. Establish trusted connection.
12. Configure secure link and encryption if necessary.
13. Tabulate each connected device and peripheral status and characteristics.
14. Poll for data between connected user devices and peripherals in multiplexing mode.
15. Generate packets for protocol and parameters supported by each device.
16. Parse and decapsulate the incoming packets, and respond accordingly.

The security system acts as a gateway between the devices and the peripherals, therefore handles the connections, packet formation and parsing. All the authentication validation and response generation and encryption of data are done by the coprocessor.

4. Security Coprocessor

The security coprocessor is a fairly complex module – its structure, its components, and functionality regarding the incorporation in our board are outlined in this section. A more detailed architecture of the coprocessor design is contained in [21]. The coprocessor is implemented in an ASIC, which is synthesized using the SIMC 65nm CMOS technology.

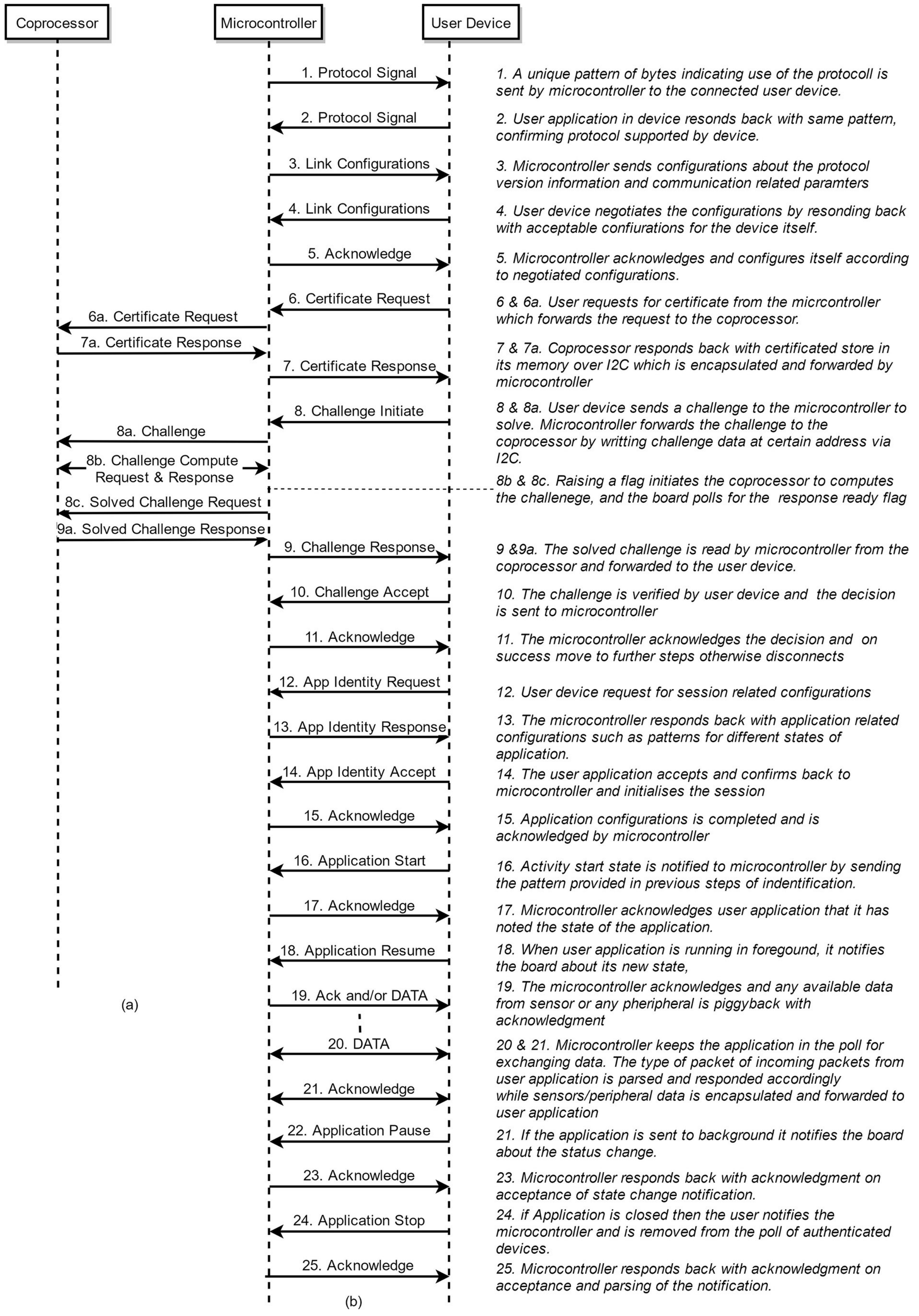
4.1. Coprocessor Overview

The coprocessor comprises two main sub-modules: the authentication and the encryption modules. The sub-modules may operate independently or together, depending on the security requirements of the communication. One sub-module performs the authentication in two steps, whereas the other sub-module performs encryption online as needed. The coprocessor is interfaced directly to the microcontroller and operates with command codes. An Inter-Integrated Circuit (I2C) is used for communication between the two. The steps of interactions between the coprocessor and the microcontroller are shown in Figure 2(a), while the complete interactions of microcontroller with a remote mobile device are presented in Figure 2(b). Further, Figure 3 shows a more detailed description of the coprocessor, which will also be described in the subsections below.

4.2. Authentication Sub-module

The authentication sub-module performs authentication in two steps. The first step involves an X.509v3 certificate, which is stored in a read-only memory. The certificate is sent to the authenticating device, where it is verified and validated. The authenticating device achieves validity by using the certification path validation algorithm specified in RFC 5280 [22]. The algorithm requires that a certificate is signed by the authorities of higher trust, which is then trusted by both parties. Should that not be the case, there will be a certificate chain, where each certificate signs another, at the end of which an entity trusted by both parties will be present. In that case, the authenticated party will need to provide all the certificates leading up to the known and trusted one. By following the chain, the authenticating party can ensure that the certificate of the authenticated party is legitimate, and the device or peripheral is genuine.

The second step involves the challenge/response exchange. A random, 64-bit long challenge, is generated by the authenticating party, and sent to the authenticated party where it is processed together with a 256-bit long secret key, also stored in the ROM. The secret key is known to both parties, and the procedure followed is HMAC-SHA256 that utilizes SHA256 which produces irreversible messages. The HMAC-SHA256 procedure computes a 256-bit long response, which is transmitted to the authenticating device. As the procedure is irreversible, the authenticating device has to compute the



1. A unique pattern of bytes indicating use of the protocoll is sent by microcontroller to the connected user device.

2. User application in device resonds back with same pattern, confirming protocol supported by device.

3. Microcontroller sends configurations about the protocol version information and communication related paramters

4. User device negotiates the configurations by resonding back with acceptable confiurations for the device itself.

5. Microcontroller acknowledges and configures itself according to negotiated configurations.

6 & 6a. User requests for certificate from the micrcontroller which forwards the request to the coprocessor.

7 & 7a. Coprocessor responds back with certificated store in its memory over I2C which is encapsulated and forwarded by microcontroller

8 & 8a. User device sends a challenge to the microcontroller to solve. Microcontroller forwards the challenge to the coprocessor by writting challenge data at certain address via I2C.

8b & 8c. Raising a flag initiates the coprocessor to computes the challenege, and the board polls for the response ready flag

9 &9a. The solved challenge is read by microcontroller from the coprocessor and forwarded to the user device.

10. The challenge is verified by user device and the decision is sent to microcontroller

11. The microcontroller acknowledges the decision and on success move to further steps otherwise disconnects

12. User device request for session related configurations

13. The microcontroller responds back with application related configurations such as patterns for different states of application.

14. The user application accepts and confirms back to microcontroller and initialises the session

15. Application configurations is completed and is acknowledged by microcontroller

16. Activity start state is notified to microcontroller by sending the pattern provided in previous steps of indentification.

17. Microcontroller acknowledges user application that it has noted the state of the application.

18. When user application is running in foregound, it notifies the board about its new state,

19. The microcontroller acknowledges and any available data from sensor or any pheripheral is piggyback with acknowledgment

20 & 21. Microcontroller keeps the application in the poll for exchanging data. The type of packet of incoming packets from user application is parsed and responded accordingly while sensors/peripheral data is encapsulated and forwarded to user application

21. If the application is sent to background it notifies the board about the status change.

23. Microcontroller responds back with acknowledgment on acceptance of state change notification.

24. if Application is closed then the user notifies the microcontroller and is removed from the poll of authenticated devices.

25. Microcontroller responds back with acknowledgment on acceptance and parsing of the notification.

Figure 2. Protocol steps between microcontroller and (a) coprocessor (b) user devices.

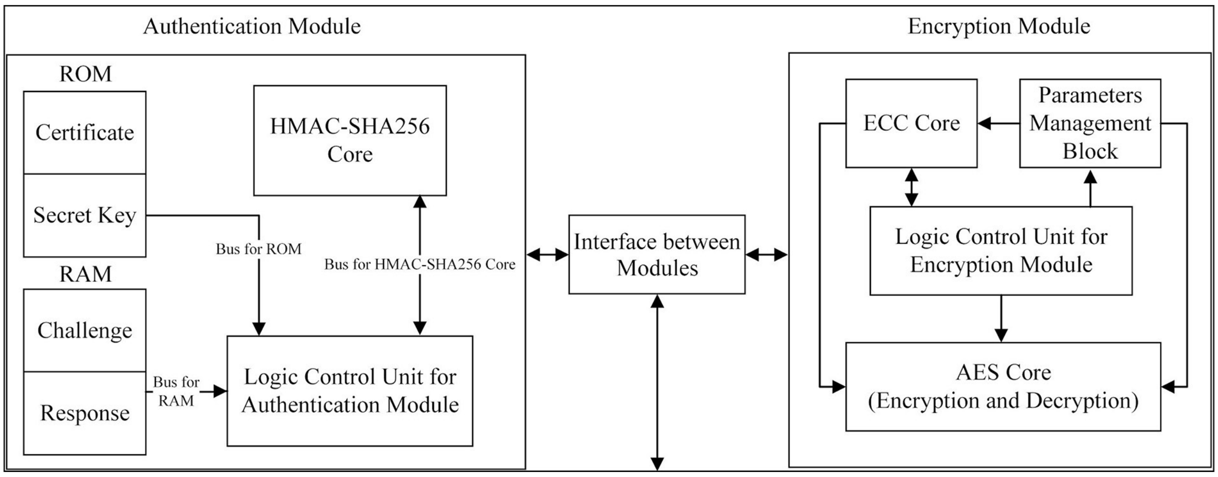

Authentication Module
ROM
Certificate
Secret Key
RAM
Challenge
Response
HMAC-SHA256 Core
Bus for ROM
Bus for HMAC-SHA256 Core
Bus for RAM
Logic Control Unit for Authentication Module
Interface between Modules
Encryption Module
ECC Core
Parameters Management Block
Logic Control Unit for Encryption Module
AES Core (Encryption and Decryption)

Figure 3. The security coprocessor.

response as well, using the known secret key, then compare it to the received one [21]. If the two match, then the secret key is correct, while the hashed response ensures that the communication is immune to the "man in the middle" MITM attacks, and the secret key cannot be extracted. The random challenge generation also ensures that reproducing the response to a new challenge will not be correct, thus countering replay attacks.

4.3. Encryption Sub-module

The encryption sub-module utilizes Elliptic Curve Cryptography (ECC) that achieves results similar to older asymmetric key cryptography methods using a shorter key size [23], [24], [25]. Using ECC as an alternative security approach was proposed decades ago, the reason being that popular approaches such as Rivest, Shamir, and Adelman (RSA) and Digital Signature Algorithm (DSA) rely on the integer factorization and discrete logarithms, which might take sub-exponential time. As a result, to ensure that an encryption is sufficiently secure, an ever-increasing key size is needed. Currently, the National Institute of Standards and Technology (NIST) suggests the use of 1024-bit long keys [26], which are equivalent to 160-bit long keys when using ECC [27]. For resource-constrained devices and peripherals, this advantage is critical.

However, symmetric key cryptography is more efficient and fast, and for this reason, fast Advanced Encryption Standard (AES) encryption ensues. Therefore, a combination of ECC and AES is utilized in the encryption. The procedure performed is as follows: at first, an ECC private key is dynamically generated and the equivalent public key is calculated and exchanged. Using the secure channel, a shared key is then generated and exchanged. Finally, with the shared key, fast encryption is performed and the communication is secured.

5. System Advantages and Applications

While the existing systems are mainly limited to Wi-Fi or wired access, the proposed module can connect via Bluetooth as well. The Bluetooth-based networks are less secure, especially for Android, so the system is capable of connecting sensors and Android devices using a secure protocol at the same time. Further, Bluetooth connections utilize a combination of Simple Secure Pairing (SSP) and LE Secure Connections (LE SC) for pairing. Pairing by itself uses some form of encryption relying on elliptic curves, however it has been recently proven that it is vulnerable to fixed coordinate invalid curve attacks [28]. Therefore, pairing by itself is not as secure, and, if intercepted, the communication will be vulnerable to MITM attacks on certain devices. Further authentication can prevent this.

The microcontroller is aware of the device types (peripheral or user device), OS (Android, iOS, Windows) and of a link (wired or wireless). Consequently, communication channels are established to comply with the protocols, such as SPP for Bluetooth connectivity with Android, iAP2 for iOS based devices, or TCP/IP followed by protocol-based communication for WiFi, File Transfer Protocol (FTP) or USB-based user devices.

The coprocessor has the certificate and key which are permanently programmed by the manufacturer, therefore the chip has to be replaced if the key or certificate has to be changed and it is one of the advantages of having the authentication/encryption task outside the main microcontroller. The multipurpose smartphones always have dedicated processors for special tasks such as pedometer step count. The microcontroller has to deal with interrupts and peripherals, Input/Output registers, buffers which require switching between tasks frequently, therefore the performance and processing is improved if the real time tasks are handled by coprocessors.

This system can find its use in many applications, including the securing of pico networks, and in poor coverage areas, such as the basements or underground metros. In a hospital, a doctor can continuously monitor sensors placed over the patient's body from anywhere in the network over the secured link. Here the sensors send data to the security system which authenticates and encrypts the data and forwards it to the user application on the doctor's smartphone. The data can only be accessed by the user who has the Protocol Specific Application which responds to the challenge and unique key. Another application of the security system is in underground metros, where the satellite signals are extremely weak, therefore the security system with GPS sensor as peripheral can be placed where it can have line-of-sight with satellites while it extends the range by forwarding the data to designated user devices within the area over the wireless link.

6. The Implementation

To evaluate the system, an implementation where IoT sensors transmit data to user personal devices is demonstrated. All IoT based devices require a wireless link to connect to the network, but the sensors provide raw data which is required to be encapsulated. Along with it, the setup of a secured link is also a crucial part before the exchange of data. Therefore, implementation of the system involves sequential steps which are required to be followed before transmission of data from peripherals to the user devices.

The test was done using the Bluetooth technology which does not implement security aspects that wireless networks do. The system acts as a bidirectional gateway between multiple nodes. It implements a star topology, and all the devices follow the authentication protocol. The sensors or peripherals can be connected to the gateway via a wired or wireless link. The Bluetooth technology used in the system is the most compatible Bluetooth 2.1 + EDR, using the Silicon Labs Bluegiga radio with a line of sight range of up to 1000 meters, the highest of all Bluetooth radios [29]. Furthermore, it is based on Enhanced Data Rate (EDR) mode that has a higher throughput, as a requirement for multi-node systems.

The system is completely automatic – once powered, it automatically connects and configures itself with all available paired devices. The user needs to perform pairing between wireless user devices and security system for the first time and the security system stores the authenticated devices into non-volatile auto-connect list, while wired devices need plugging in. Upon security system power up, it connects with each auto-connect listed device available within wireless range and the ones connected over wired link simultaneously, and performs authentication and encryption using the protocol procedure provided, with a dedicated protocol compatible application installed on user devices. The data from all the connected peripherals is sent to the user devices over the same link saving a huge bandwidth. The packet headers and payload headers contain the information of the peripheral device from which the data is received. Addition or elimination of nodes does not affect other devices connected to the system. The devices receive notifications about the eliminated node and the fact that no more data will be exchanged, or the addition of a new node.

6.1. Connection Capabilities

The system can support 7 simultaneous wireless connections, and multiple master-slave based devices can be connected via UART and

I2C. As seen in Figure 4, a wired sensor was connected to the system using UART while the coprocessor and a non-volatile memory were connected via I2C multi-master/slave mode. The very first step when a new device connects is the exchange of a protocol signal packet as shown in Figure 2(b), first packet between microcontroller and user device. This packet defines that both ends follow the same protocol, failing of which leads to termination of the connection.

The protocol signal is sent by the microcontroller and is responded by the application installed on the user device. The acknowledgment by user device defines the availability of protocol dedicated application on the user device. It follows a configuration of all the channels, and negotiation of data-related requirements, which include protocol details, payload sizes, timeouts, and cumulative acknowledgments. The configuration is exchanged between connected nodes, and feasible configurations are followed throughout the communication. Thereafter, the authentication procedure shown in Figure 2(a) is performed, in which certificate and challenge are exchanged. The user devices request challenges from the microcontroller and the microcontroller removes the protocol header and sends the payload to the coprocessor. In response, the coprocessor generates the challenge by hashing a key, as explained previously. The response is encapsulated in the protocol packet by the microcontroller and forwarded to the user device. The same procedure is followed for the certificate exchange between user device, microcontroller, and coprocessor. The flow of requests and responses is shown in Figures 2(a) and 2(b). The entire protocol procedure is repeated for each user device. The user device applications (Android/iOS) initiate the request while microcontroller responds to the request. The user application executes all the protocol steps as listed in Figure 2(b) under user device.

Upon correct acceptance of a certificate and a response to the challenge, the procedure initiates data exchange mode. In this mode, polling of data between devices and peripherals is performed, along with inquiry of new incoming connections or disconnections. The data from the peripheral is encapsulated in the protocol payload and forwarded to the devices, while the devices either acknowledge each or multiple packets, or send back data along with the acknowledgement. The data from devices is in protocol-derived packets, which are parsed and the relevant data is forwarded to peripherals.

The data sent over wireless link needs a suitable User Interface (UI) to display the data and a smartphone application based on the protocol running on the security system. The user devices look for a Bluetooth Universally Unique Identifier (UUID) for the very sole purpose of connecting with the system. The user applications follow the same procedure as the system does and counter-reply the protocol packets from the system. The application allows the

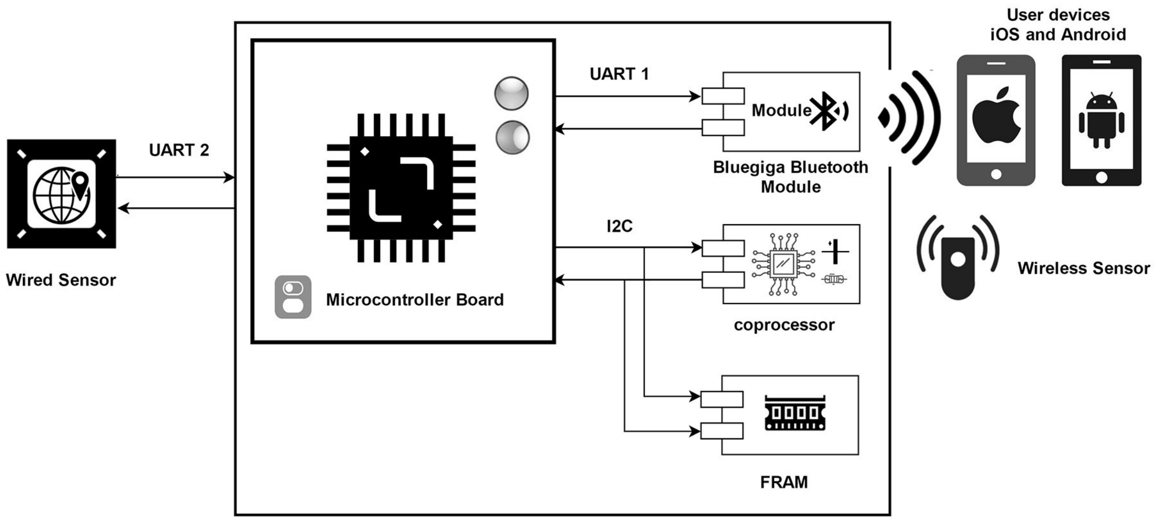
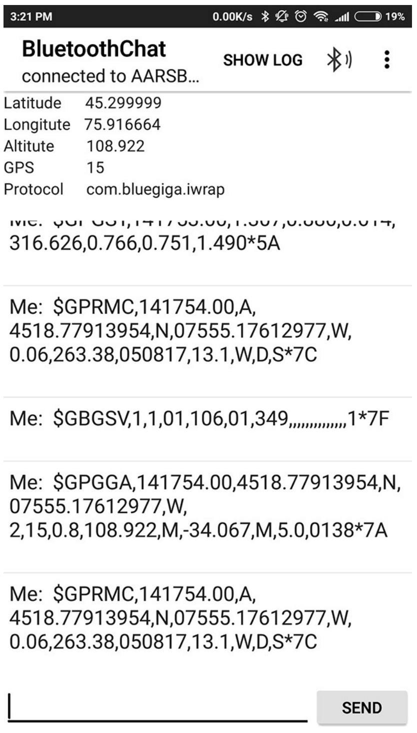


Figure 4. Schematic of security system.

users to send the data to the system which can be used to configure the sensor or peripheral settings such as power mode, intervals etc. In the background, the application follows authentication steps at first, and then initializes the encrypted data channel. The incoming protocol-based packets are acknowledged depending on the configurations. The UI of the devices displays the parsed data and also displays the source of the incoming data, as shown in Figure 5. The security system is programmed to connect with both Android and iOS-based devices using the same generic protocol; hence, it is a multiplatform-compatible system that connects to both platforms using a virtual serial port and the Serial Port Profile (SPP) of Bluetooth [30]. Furthermore, it supports connections with wireless sensors, which could possibly be near the range devices that have a short communication range, whose raw data is required to be secured.



connected to AARSB...

Latitude	45.299999
Longitute	75.916664
Altitute	108.922
GPS	15
Protocol	com.bluegiga.iwrap

```
Me:  $GPGST,141755.00,1.567,0.880,0.614,
316.626,0.766,0.751,1.490*5A

Me:  $GPRMC,141754.00,A,
4518.77913954,N,07555.17612977,W,
0.06,263.38,050817,13.1,W,D,S*7C

Me:  $GBGSV,1,1,01,106,01,349,,,,,,,,,,,,,1*7F

Me:  $GPGGA,141754.00,4518.77913954,N,
07555.17612977,W,
2,15,0.8,108.922,M,-34.067,M,5.0,0138*7A

Me:  $GPRMC,141754.00,A,
4518.77913954,N,07555.17612977,W,
0.06,263.38,050817,13.1,W,D,S*7C
```

Figure 5. Android user interface.

A multiplexing mode is required when there are multiple connections and there could be the possibility of passing commands to one device or configuring the Bluetooth module while the other device is exchanging a data stream. The multiplexing mode (MUX) can handle both commands and data in one mode. For the proper distinction between data and connection related commands to Bluegiga Bluetooth Chip and further to the device that is currently dealt with, a special syntax is followed. The syntax is the same for both sending data over UART by the host, and receiving data from the iWRAP firmware [31]. The mode provides a distinction between packets to and from multiple devices. The MUX frame encapsulates the protocol packet of the authentication system. After authentication, the procedure enters a polling loop and the loop runs infinitely, continuously forwarding data in both directions. Using multiplexing, multiple devices are secured by authenticating/encrypting separately and then sending data over the secured links.

6.2. Hardware

The circuit consists of sub-modules which include coprocessor, Bluegiga(BG) Bluetooth Module and user interface LEDs, buttons and Sensor interface via UART. The BG WT41u, which is a Bluetooth module, is connected to an STM32L432 microprocessor via UART.

The UART to USB debugger terminal RXs are connected to both TX and RX wires of the microcontroller for performance computation. The power pins of the BG module are connected to the power and ground as stated by the Bluegiga WT41u datasheet. The coprocessor whose architecture is explained in Section 4, is connected to the microcontroller via I2C pins along with an FRAM. The board, designed in Cadence Allegro PCB Editor, is shown in Figure 6.

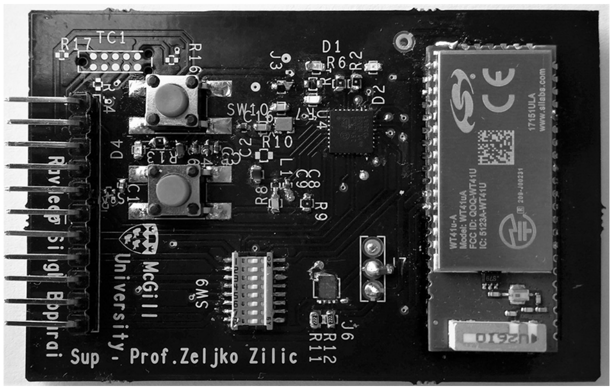


Figure 6. Assembled PCB.

6.3. Working Model and Experimental Results

The assembled project, with all the components and modules soldered on the PCB, were able to connect and authenticate the devices paired to the module. Two tests were conducted for performance evaluation. In Test 1, one device and one wired sensor were used, while Test 2 was conducted using 2 Android and 2 iOS devices with one wireless SensorTag (in this case, it was implemented as an Android device that sends the unsecured data) and one wired sensor. Test 1 uses normal mode, while Test 2 uses MUX enhanced algorithm which lets the user handle data and control simultaneously. All user devices continuously accept and acknowledge incoming data. In the test, the complete states of the protocol were explored and the system was shown to work for days without failure.

The working model in Figure 7 demonstrates the transfer of the location data via the standard NMEA sequences. As one built-in feature of iOS is that the location data cannot be provided to the mobile device without the proper authentication, as this demonstrated interoperability with iOS is readily included in our system. In contrast, the YubiKey device does not support the iOS operation due to the stated "incompatibility" by YubiKey.

In IoT applications, the sensor-based systems feature two characteristics; low power and latency. The low power operation is achieved by putting the system to sleep when no data is being transmitted. The typical sensor update is received after 1 second from the location-based SensorTags, whereas high throughput is necessary for the systems which require continuous monitoring in real-time, while low power consumption is not a priority.

The project was tested for its performance in which the time taken was computed between every sensor data transmission packet by using a very precise Systick timer of the microcontroller, and timer difference was sent to the Debug window via debug UART. The project is BR/EDR based, therefore the throughput is required to be very high. In Test 1, with MUX feature disabled, it was found that the time difference between each packet transmitted to BG was around 10 milliseconds, where packet sizes were from 40 to 90 bytes. For 3562 data bytes (28496 bits) it took 500 ms, bringing the throughput to 57 kbps. With 18 bytes per header,

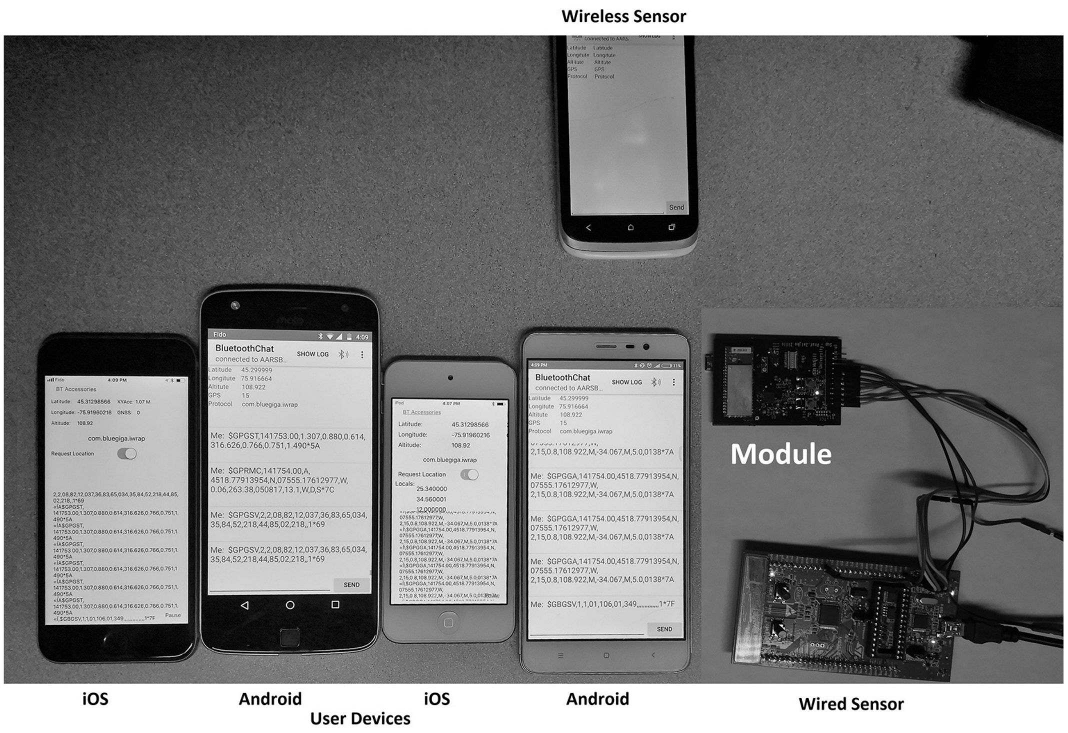


Figure 7. Working model.

per packet, there are a total of 918 header bytes making a total packet of 4480 bytes (35840 bits) giving 71 kbps throughput with a baud rate of 115200 bps, which is proven to be the high throughput of about 61.7% over the wireless link as compared to the wired link baud rate between microcontroller and Bluegiga Bluetooth chip. The throughput for certain time duration is encapsulated in Table 1. Test 2 computed the performance of the system with the multiple devices, the MUX was enabled which required different algorithm to handle UART communication. From the experiment, it was found that the time taken in updating each device in multiple device communication is increased by *n* times the time required for communication between the security system and one device while throughput decreases by *n* times for each device where *n* is the number of devices.

The Bluegiga Bluetooth module supports serial communication with devices, therefore it does time division multiplexing. Table 2 shows the computation of time taken and throughput for each device based on total 3596 data bytes, where each packet has 18 header bytes and along with it, 5 MUX bytes, which are only sent between microcontroller and Bluegiga. Therefore, the MUX bytes are not sent over the wireless link. The complete header is of 1150 bytes, making the packets of 4496 bytes (35968 bits) excluding MUX bytes, make the throughput of the security system as high as 75.7 kbps that excludes MUX bytes. Table 2 is the actual performance between the system and each device when implemented using MUX algorithm.

Table 1. Throughput of the system.

Time	Data bits	Header bits	Total bits	Throughput
500 ms	28496	7344	35840	71 kbps

Table 2. Throughput and time taken between system and each device for 4496 bytes.

Number of devices	Time taken (ms)	Throughput of each device
1	475	75.7 kbps
2	931	38.6 kbps
3	1430	25.1 kbps
4	1806	19.9 kbps

The test results were perfectly close to the natural number multiplier, since the average time is stable either for one device or for many devices. Hence, there is no gain or loss of time between sending a packet to one device or to many devices by the microcontroller because each packet is independently sent by microcontroller, irrespective of the number of devices. The result of throughput for communication with one device in Test 2 is higher than in Test 1 because the algorithms were different for both cases. Test 2 is based on MUX that implements dual DMA for Sensor/Peripheral and communication module, which makes the system performance much faster and robust.

The multiplexing also reduces the power consumption by the factor of number of devices connected, since the same security system can secure multiple nodes using one microcontroller, one coprocessor and one wireless chip, whereas all other studied systems are point to point such as USB Type-C [19], where every device uses a separate system.

Further, the latency of the coprocessor sub modules was tested. The authentication module has a latency of 0.95 ms, that is the time it takes to retrieve the X.509v3 certificate and compute the challenge response with the HMAC-SHA256 procedure. The encryption module has a latency of 1.75 ms for the setup phase, that is the ECC key generation and AES key computation. Encryption by itself only takes 0.01 ms per packet, which is not very significant [21]. The coprocessor is active for only a tiny fraction of the total time, which can be in the hundreds of milliseconds. However, the communication between that and the microcontroller introduces overhead. As such, any time saved by outsourcing the security operations to the coprocessor is lost to that overhead. It can be safely estimated that if the microcontroller alone were to handle the security operations, the difference would not be noticeable and the throughput would be unaffected.

While the system is secure by design, vulnerabilities can exist. For instance, there can exist non-secure communication between a wireless, or even a wired sensor and the microcontroller. If the physical control of the interface or sensor is lost, the unsecured side of the communication can be then intercepted – it is assumed that physical control is guaranteed. The wired sensor is in general harder to break in, however it is still possible to gain access to the wired interfaces within the system. The sensor could be accessed by an impersonating microcontroller, which could retrieve all raw sensor data should that be the case. This can be mitigated by using programmable sensors where the platform specific protocol application can be implemented. Finally, the security coprocessor found within the system could be attacked. While the coprocessor by design is using inherently secure algorithms, and features tamper evidence, it could still be attacked on a hardware level, with information extracted out of the chip, or worse, the secure algorithms could be compromised (*i.e.* SHA collision). For all intents and purposes, the system can be considered secure but not virtually impenetrable.

7. Conclusion and Future Work

An authentication and encryption board was designed in such way that it extends and surpasses the security authentication scheme present for iOS devices to Android and other computing systems. The board can be used in a variety of scenarios to facilitate secure access to the common computing platforms and wired/wireless peripherals. The system is energy-efficient, because of the simultaneous securing ot the access between different devices. It is unique in that it provides multi-point functionality, in other words it can secure multiple communications without requiring a separate module for each communication channel, which also reduces the cost. The system has high throughput along with high-security level protection and can be implemented in the new or already existing system. Since iOS secures the access at OS level, which is proprietary, the proposed scheme can be used at an application level with all the devices, but due to the open-source nature of Android, the authentication hardware provisions, and even the encryption support could be easily incorporated at OS level.

The present-day industries still have a lot of wired sensors and peripherals (via the existing UART, SPI, or I2C interfaces) in which the system can be integrated to fetch the readings over the secured wireless link and even send commands back to them. An addition such as cloud upload via one of the connected Bluetooth devices can serve the further IoT purpose. The design can also be extended further to low-power systems that require high throughput, including the real-time systems.

References

[1] V. Pureswaran and P. Brody, "Device Democracy: Saving the Future of the Internet of Things", IBM Corporation (2015). Available: http://public.dhe.ibm.com/common/ssi/ecm/gb/en/gbe03620usen/mc_asset_3314565__gbe03620usen-04_GBE03620USEN.pdf

[2] V. P. Kafle *et al.* "Internet of Things Standardization in ITU and Prospective Networking Technologies", *IEEE Communications Magazine*, vol. 54, no. 9, pp. 43–49, 2016. http://dx.doi.org/10.1109/mcom.2016.7565271

[3] M. Stevens *et al.* "The First Collision for Full SHA-1", Advances in Cryptology – CRYPTO 2017 Lecture Notes in Computer Science, 2017, pp. 570–596. http://dx.doi.org/10.1007/978-3-319-63688-7_19

[4] M. L. Das, "Two-Factor User Authentication in Wireless Sensor Networks", *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1086–1090, 2009. http://dx.doi.org/10.1109/twc.2008.080128

[5] R. Watro *et al.*, "TinyPK: Securing Sensor Networks with Public Key Technology", *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM, 2004. http://dx.doi.org/10.1145/1029102.1029113

[6] W. Shi *et al.*, "A New User Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography", *International Journal of Distributed Sensor Networks*, vol. 9, no. 4, pp. 730831, 2013. http://dx.doi.org/10.1155/2013/730831

[7] A. Salman *et al.*, "A Light-Weight Hardware/Software Co-Design for Pairing-Based Cryptography with Low Power and Energy Consumption", *IEEE International Conference on Field Programmable Technology (ICFPT)*, 2017. http://dx.doi.org/10.1109/fpt.2017.8280149

[8] M. Gao and Q. Gang, "A Novel Approximate Computing Based Security Primitive for the Internet of Things", *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017. http://dx.doi.org/10.1109/iscas.2017.8050360

[9] Ch. Lesjak *et al.*, "Hardware-secured and Transparent Multi-Stakeholder Data Exchange for Industrial IoT", *IEEE 14th International Confer-*

ence on Industrial Informatics (INDIN), 2016. http://dx.doi.org/10.1109/indin.2016.7819251

[10] S. Babar *et al.*, "Proposed Embedded Security Framework for Internet of Things (IoT)", *IEEE 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, 2011. http://dx.doi.org/10.1109/wirelessvitae.2011.5940923

[11] X. Zheng *et al.*, "Design and Implementation of a DPA Resistant AES Coprocessor", *IEEE 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM'08*, 2008. http://dx.doi.org/10.1109/wicom.2008.1087

[12] G. Varshney *et al.*, "A Security Framework for IOT Devices Against Wireless Threats", *IEEE 2nd International Conference on Telecommunication and Networks (TEL-NET)*, 2017. http://dx.doi.org/10.1109/tel-net.2017.8343548

[13] S. Srinivas, "Security Keys: Practical Cryptographic Second Factors for the Modern Web", *Financial Cryptography and Data Security: 20th International Conference, FC 2016*, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers. vol. 9603. Springer, 2017.

[14] D. Nilsson, "Yubico's Take On U2F Key Wrapping", 2014. https://www.yubico.com/2014/11/yubicos-u2f-key-wrapping/

[15] J. Ehrensvärd, "Secure Hardware vs. Open Source", yubico blog[online], May 16, 2016.

[16] P. Švenda *et al.*, "The Million-Key Question – Investigating the Origins of RSA Public Keys", *Proceedings of the 25th USENIX Security Symposium*, 2016.

[17] D. J. Bernstein *et al.*, "Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild", *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-42045-0_18

[18] M. Nemec *et al.*, "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli", *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017. https://doi.org/10.1145/3133956.3133969

[19] S. Wallick and Intel, "USB Type-C™ Authentication" USB Implementers Forum © 2017, Taipei, Taiwan, October 24 – 25, 2017. Available: http://www.usb.org/developers/presentations/USB_DevDays_Taipei_2017_USB-C_Authentication_USB_PD_Firmware_Update.pdf

[20] D. Balfanz *et al.* "Web Authentication: An API for accessing Public Key Credentials Level 1", in W3C Candidate Recommendation, 20 March 2018. https://www.w3.org/TR/2018/CR-webauthn-20180320/

[21] J. Wang *et al.*, "An ASIC Implementation of Security Scheme for Body Area Networks," *IEEE International Symposium on Circuits and Systems(ISCAS)*, Florence, 2018. http://dx.doi.org/10.1109/ISCAS.2018.8351098

[22] D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", 2008. http://dx.doi.org/10.17487/rfc5280

[23] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987. http://dx.doi.org/10.2307/2007884

[24] I. Blake *et al.* "Elliptic Curves in Cryptography", vol. 265, Cambridge University Press, 1999.

[25] L. C. Washington, "Elliptic Curves: Number Theory and Cryptography", Chapman and Hall/CRC, 2003.

[26] R. Gallant *et al.*, "Improving the Parallelized Pollard Lambda Search on Anomalous Binary Curves", Mathematics of Computation of the American Mathematical Society, vol. 69, no. 232, pp. 1699–1705, 2000. http://dx.doi.org/10.1090/s0025-5718-99-01119-9

[27] E. B. Barker *et al.*, "Recommendation for Key Management Part 3: Application-Specific Key Management Guidance", No. Special Publication (NIST SP)-800-57 Pt3 Rev 1. 2015. http://dx.doi.org/10.6028/NIST.SP.800-57pt3r1

[28] E. Biham and L. Neumann, "Breaking the Bluetooth Pairing – Fixed Coordinate Invalid Curve Attack", Technion – Israel Institute of Technology. Available: https://www.cs.technion.ac.il/~biham/BT/bt-fixed-coordinate-invalid-curve-attack.pdf

[29] Silicon Laboratories Inc., "WT41u Data Sheet", 2017. Available: https://www.silabs.com/documents/login/data-sheets/wt41u-datasheet.pdf

[30] Silicon Laboratories Inc., "AN990: Bluetooth Serial Port Profile", 2012, datasheet. Available: https://www.silabs.com/documents/login/application-notes/AN990.pdf

[31] Silicon Laboratories Inc., "iWRAP 5.7.0, iWRAP 6.2.0 and iWRAP 6.1.1", 2017, datasheet. Available: https://www.silabs.com/documents/login/reference-manuals/iWRAP6-API-RM.pdf

Received: May 2018
Revised: January 2019
Accepted: February 2019

Contact addresses:

Ravdeep Singh Boparai
McGill University
Montreal
Canada
e-mail: ravdeep.boparai@mail.mcgill.ca

Anastasios Alexandridis
McGill University
Montreal
Canada
e-mail: anastasios.alexandridis@mail.mcgill.ca

Zeljko Zilic
McGill University
Montreal
Canada
e-mail: zeljko.zilic@mcgill.ca

RAVDEEP SINGH BOPARAI received his degree of Bachelor in Technology in Electronics and Communication Engineering from I.K. Gujral Punjab Technical University, Jalandhar, India in 2016 and Master in Electrical Engineering, with specialisation in Integrated Circuits and Systems in 2018 from McGill University, Montreal, Canada. He worked as Research Professional on VoIP based Gateways. He works on the Networking and Security Protocols at the Research and Development Department of Paradox Security Systems, Canada. His research interests are in the design, development and applications of embedded systems, Internet of Things (IoT), wireless communications, and security systems.

ANASTASIOS ALEXANDRIDIS was born in Volos, Greece, in 1990. He received the B.Sc. degree in Computer Engineering from Frederick University, Nicosia, Cyprus, in 2015, and the M.Sc. degree in Analog Electronics from the University of Edinburgh, Edinburgh, UK, in 2016. He is currently pursuing his Ph.D. degree working together with the Integrated Microsystems Laboratory at McGill University, Montreal, Canada. His current research interests include but are not limited to embedded systems, Internet of Things (IoT), blockchains, body area networks, mesh networks, and smart cities.

ZELJKO ZILIC received the B.Eng. degree from the University of Zagreb, Croatia, and the M.Sc. and Ph.D. degrees from the University of Toronto, Canada. From 1996 till 1997, he worked for Lucent Microelectronics on the design, test, and verification of Orca FPGAs. His current research interests include the design of deeply embedded systems that most notably deal with wellness and health. He has published more than 300 papers, for which he received a dozen of Best Paper or Honorary Mention Awards. He has supervised more than 80 M.Eng. and Ph.D. students, who have moved on to leading industrial and academic institutions. Prof. Zilic has been granted the Chercheur Strategique Research Chair from Quebec. He is a Senior Member of IEEE and ACM.