2014 IEEE 14th International Conference on Bioinformatics and Bioengineering

Universal Physical Access Control System

Bassem Alhalabi, Clyde Carryl
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida, USA
{ alhalabi, ccarryl} @fau.edu

***Abstract*—With the recent rapid increase in the number of physical facilities and structures that need to be protected by restricting physical access to them, there has been an explosion in the number and type of physical access control systems being deployed to protect them. However, these systems are quite different from each other and there is no common standard that provides for interoperability between the various systems. The number and types of access devices being employed has grown steadily, but the systems in which they are being used are physically and technologically incompatible with each other. Consequently, there is renewed interest within the research community in developing a common universal system providing physical resource access protection regardless of the type of physical resource and where it is located. In this article we propose the Universal Physical Access Control System (UPACS) which provides a universal framework for controlling access to physical resources. It provides for the use of a wide variety of access devices and allows for both onsite and remote access. We show how it can be used to control access to any type of resource, including homes, vehicles and public infrastructure such as street lights and traffic lights and industrial infrastructure such as power plants. We also show how it can be implemented regardless of the location of the owner of the physical resource and the location of the resource relative to its users.**

Keywords—Physical Access Control Systems, Remote Access, UPACS, Asset Security.

I. INTRODUCTION

The many physical access control systems in use today are incompatible with each other. Each system controls access to the physical resources it is deployed to protect by providing a proprietary interface between an access device and a proprietary protecting device deployed at the physical resource to be protected. And due to these limitations and the recent proliferation of access control systems, there has been renewed interest within the research community in developing a universal access control system that functions across all manufacturer and device type platforms and provides for the seamless interchange of access control devices between access control service providers while still ensuring secure access to the protected physical resources.

To solve these problems we propose the Universal Physical Access Control System (UPACS), an encryption-enabled protocol which provides secure access to physical assets, each protected by a single customizable access control device. Access may be onsite or remote, and may utilize several types of access devices, including mobile phones, tablets, computers or any future devices that are capable of communicating over a public network. Furthermore, the number and types of physical resources it is capable of protecting is boundless. Due to its open architecture, it will be able to protect access to future physical assets with yet unthought-of functionality. The protocol is applicable anywhere access to physical facilities needs to be restricted, for example power plants, chemical plants, public infrastructure, as well as private property, including homes. Critical switches, doors, windows, and lights can all be protected, and sensor readings may also be received from remote sensors.

Users determine the number and configuration of the physical assets they may control, being able to easily add to and delete devices from their portfolio. Users may also assign or transfer access rights to other users, facilitating delegation of management of access control activity. Users manage access control by issuing commands to network nodes, one of which is physically positioned at each device under management. Depending on their requirements, users may delegate all or a subset of their own access permissions required to send commands to a node, and they can do so without the need for supervisor interaction. Users with appropriate permissions may easily commission and decommission network nodes, and remove users with only delegate rights as they require.

The remainder of this paper is structured as follows: section II describes the evolution of physical access control systems, highlighting the most important and relevant ones inspiring the development of UPACS; section III describes the UPACS framework and protocol suite in detail; and in section IV is the conclusion and plans for future work.

II. CRITICAL REVIEW OF CURRENT PHYSICAL ACCESS CONTROL SYSTEMS

Access control systems have been deployed in many applications in everyday life. They are used to restrict access to critical industrial plant facilities, for example an electrical power grid. They are also used to control access to government and military facilities, and are increasingly employed to control access to residential appliances.

There are two broad categories of access control systems [1]. Mechanical access control systems allow access only

978-1-4799-7502-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BIBE.2014.64

when there is a mechanical match between an access device and a blocking device. One simple example of such a system is a lock and key or unlocking combination, with the lock blocking access to protected resources until a key or unlocking combination is used to gain access. Unfortunately, this type of system is useful only when the user is physically present at the blocking device along with the access device and cannot be used remotely. Another limitation of mechanical access control systems is that any required security enhancements to the system may require costly replacement, reconfiguration or re-issuance of all of the access devices, and as such may be difficult to implement [1].

Unlike their mechanical counterparts, electronic access control systems rely upon an electronic match between an access device and the access control system. User authentication is performed by means of an electronic authentication protocol, and can be done either locally or remotely. Compared with mechanical systems, such systems are more suited to rapid adaptation to meet changing deployment requirements. As a consequence, electronic access control systems have evolved much more rapidly than their mechanical counterparts, and have benefitted from the development of several electronic authentication protocols.

A. Electronic Authentication Protocols

Currently the four most technologically important categories of electronic authentication protocols are those based on cryptography, biometrics, Physical Unclonable Functions (PUFs), and access cards. Each class of protocol makes use of a secret key or access process by which a user can gain access to protected resources. However, as with mechanical access control systems, there are strengths and weaknesses inherent in each type of approach. In particular, it is easy to understand how compromise of the access key could jeopardize the effectiveness of the protocol, though it must be noted that this is less of a risk factor with regard to biometric protocols.

Cryptography-based Protocols: Cryptography uses keys to provide secure communication between two entities. There are two categories of cryptographic protocols: symmetric protocols and asymmetric protocols.

Symmetric cryptographic protocols use a single key for both encryption and decryption. Two well-known examples of symmetric cryptographic protocols are the Data Encryption Standard (DES) and Advanced Encryption Standard (AES) protocols. Once adopted as the United States federal government standard for exchange of sensitive documents, DES is now considered to be no longer secure [2] due to its 56-bit key size limitation. It has since been replaced by AES, which allows key sizes up to 256 bits. However, even AES is not as secure as it once was as available computing power that can be used to attack systems protected by AES grows sharply [2].

Asymmetric cryptographic protocols use a public key and a private key to provide secure communication between two entities. The public key is used by a sender to encrypt messages intended for a chosen recipient, and the private key is known only by the intended recipient, who uses it to decrypt messages it receives that were encrypted using its published public key. Examples of asymmetric cryptographic protocols are RSA (named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman), Diffie-Hellman with the Digital Signature Algorithm (DH/DSA), and Elliptic Curve Cryptography (ECC).

RSA encrypts messages using a public key algorithm based on the product of two very large prime numbers, and decrypts messages using a private key algorithm based on the two prime numbers used to create the associated public key [2]. Due to the difficulty of factoring the product of two very large prime numbers to recover the factors given only the product, RSA has proven to be a very secure encryption protocol. However, increases in computer processing power used by attackers require explosive growth in RSA key sizes in order to provide the same level of security. Diffie-Hellman with the Digital Signature Algorithm (DH/DSA) is an improvement on RSA and achieves its strength from the difficulty of solving the discrete logarithm problem. DH/DSA computes a function $y = (gx) \bmod p$, where p is a very large prime number and g and x are smaller than p. Encryption using public key y and decryption using private key x are done by solving the (tractable) exponentiation problem. However, unauthorized decryption requires solving the discrete logarithm problem which is intractable for very large p [2], since although y is known, x is not. Unfortunately, as with RSA, increases in compute power used by attackers requires explosive growth in key sizes in order to maintain the same level of security.

ECC is an improvement upon DH/DSA that also requires growth in key sizes in order to keep up with increased attacker computing power, but more modest growth than required by DH/DSA [2]. It is similar to DH/DSA in that successful attacks require solving an intractable discrete logarithm problem. However, the prime field upon which p is defined is a set of points on an elliptical curve, with specific rules as to how to get from any point on the curve to the next point in sequence. ECC defines points P and Q on an elliptical curve $y^2 = x^3 + ax + b$ such that $Q = kP$, where Q is a public key and k is its corresponding private key. Encryption using public key Q and decryption using private key k require solving a (tractable) elliptic curve multiplication problem. However, unauthorized decryption requires solving the elliptic curve discrete logarithm problem, which is intractable for very large p [2] since although Q is known, k is not. However, although the required growth in key sizes to keep pace with the growth in available attacker computing power is not as steep as for DH/DSA, it is still substantial and represents a limitation of even this class of cryptographic protocols.

Biometrics-based Protocols: Biometric authentication uses unique characteristics of individual users to identify and authenticate them to the system. Among the more commonly used identification features are the user's iris, retina, fingerprint, palm print, voice, face or signature, as these are considered to be unique for each person [3]. Other features studied include peculiarities in eye movements [16] and electroencephalogram (EEG) signals [15]. However, to avoid security breaches due to the unauthorized imitation of one of these features, it is common for biometrics-based protocols to

employ multi-factor authentication, basing classification on more than one identification feature [3], for example voice and fingerprint.

However, biometrics-based protocols suffer from the limitation that access privileges cannot be transferred [1] among users, and in addition they are exposed to security breaches if the reference database is compromised.

Chip-level Authentication Protocols: Often access devices are identified and authenticated using protocols based on an encrypted key digitally stored in non-volatile memory. Such protocols are able to authenticate a device to the system if its stored key is present in the system's reference database. However, digitally stored keys are vulnerable to security breaches, and protocols based on encryption are impractical for power-constrained devices [4]. To solve these problems, devices can be uniquely identified by employing a Physical Unclonable Function (PUF) [4] [11] [12] [14]. A PUF is a function determined from the random peculiarities of the fabrication characteristics of individual integrated circuits, for example transistor delay signatures along specific paths [4]. It is known that when integrated circuit chips are fabricated, the timing delays along wiring and transistor paths through the integrated circuit are unique for each IC, and as such represent a unique identifier for each IC. A PUF creates a set of challenges and responses that comprise a lock to which there is only one key: the specific IC for which the lock was created, thus allowing authentication of only the correct device.

However, PUFs are susceptible to bit-flipping [17] and device aging [13]. Timing differences between signals propagated through the IC along two paths of similar length sometimes change such that the slower path becomes faster than the previously faster path, resulting in a phenomenon known as bit-flipping. As a consequence, the correctness of operation of any PUFs built using the paths in question will be adversely affected [17]. In addition, as the IC gets older propagation timings along various paths may change, thereby affecting PUF accuracy [13]. Yet another limitation of PUF security is that it is vulnerable to compromise in the event of unauthorized access to the PUF challenge-response pair reference database, or to eavesdropping of the challenge-response pairs exchanged between legitimate users and the system [11].

Access Card Authentication Protocols: Access card authentication protocols use identification (ID) information stored on an access card and an authentication algorithm to determine if the access card with the given ID is allowed access to the requested resources. They may also determine the user's appropriate level of access to each resource and restrict access accordingly. An acknowledged best practice is to avoid storing sensitive access control information (such as user passwords) on the server but to store any required information on the card itself, and several protocols have been designed around this principle [6] [7] [18]. And to further enhance security, some access card authentication protocols employ multi-factor authentication [8] [9] [19], combining card authentication with other authentication factors, such as a user's password. In general, access control protocols can be segmented into three phases: the Registration Phase, the Access Request Phase, and the Verification Phase [5].

Registration Phase: In the registration phase, the user submits his ID and password to the server, which then creates an access control list with appropriate permissions for each server resource and stores it along with a hash function on a new access card to be issued to the user.

Access Request Phase: In the access request phase, the user requests access by presenting his access card to the server along with his ID and password, and the access card sends a request to the server for the pre-assigned access levels.

Verification Phase: In the verification phase, the server verifies the validity of the user and request, verifies that the requested resource access levels are as pre-assigned, and approves/denies the request as appropriate. If the request is approved, the server sends the approval response to the card, which then attempts to verify the validity of the approval and the server.

A security analysis reveals the system's vulnerability to various forms of attack, and its ability to protect the privacy of communications between the server and users, determined by whether or not messages between the server and users are visible to others. There are several known types of attack.

Reflection Attack: A reflection attack occurs when an adversary intercepts a legitimate user's messages and impersonates the server.

Parallel Session Attack: A parallel session attack occurs when an adversary intercepts messages from the server to a legitimate user and uses the user's credentials to impersonate the user and begin a new session with the server.

Privilege Elevation Attack: A user with previously valid but now revoked resource access levels may attempt to access resources at the revoked access levels.

Replay Attack: An adversary may replay a message between the server and a legitimate user.

Man-in-the-Middle Attack: An adversary may intercept and modify messages between the server and a legitimate user.

TABLE I. UPACS PROTOCOL

Process	Purpose
Resource Registration	Register a new resource
Child Node Addition	Add a child node to control an individual device
Child Node Deletion	Delete a child node that is no longer needed
Access Rights Modification	Assign all or a subset of a user's access rights
User Deletion	Delete a user
Resource Actuation	Actuate a physical device

III. UPACS Framework and System Description

UPACS is a communication protocol providing secure access to a wide variety of physical devices and the ability to control the behavior of those devices over an unsecure network, such as the internet. As such it provides secure access to physical devices from anywhere there is an internet connection. It is made up of four main components: a user interface from which requests to interact with remote physical devices can be made; a resource controller known as a parent node which is accessible to the user over any public or private network; local device controllers known as child nodes, that are placed on location wherever a physical device is to be accessed and manipulated; and a trusted key server capable of administering cryptographic key management over the network. Protocol processes are designed to withstand network attacks enabled by the use of public untrusted networks in fulfilling the requirements for users to interact with remote physical resources, which may be any assortment of homes, physical plants, public infrastructure, or any other physical facilities for which secure remote access is required.

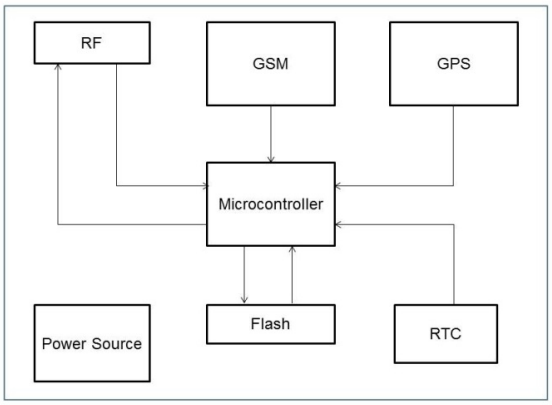

Figure 1 – Parent node

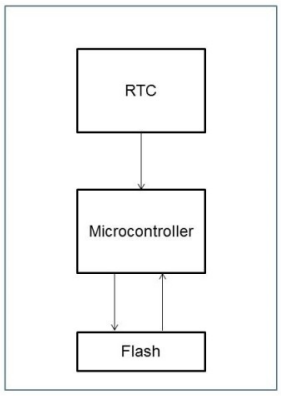

Figure 2 – Child node

Messages exchanged between protocol participants are sent over a public channel, thereby allowing an attacker to freely inspect and attempt to manipulate them. The protocol uses symmetric and asymmetric encryption techniques to protect the secrecy of these messages, with a trusted key server tasked with the responsibility of facilitating mutual authentication of the other protocol participants.

In the execution of the protocol, *users* request services from *parent nodes*, each of which controls access to one or more *child nodes*. Each child node protects access to a single physical device, and authorized users may send commands to the child nodes to control the behavior of the devices under their control. A child node may be deployed anywhere on the network, since the protocol is not limited in terms of geographical placement of managed resources. Examples of physical devices that may be controlled by a child node include doors, windows, lights, medical devices, sensors, chemical devices, nuclear devices, and so on.

Parent nodes require a CPU, a real time clock (RTC) to provide time stamps, a GPS to provide location, flash memory to provide non-volatile storage of user and parent identities and nonces as well as device commands and user permissions, a near field communication (NFC) component to allow secure configuration of child nodes, and a GSM module for global network identity. A child node requires only a CPU, RTC and a small flash memory to store their commands and associated responses. Figure 1 depicts a parent node and Figure 2 depicts a child node. Figure 3 models a building being protected by a system of *n* UPACS child nodes.

The process of providing secure access to a physical resource using the Universal Physical Access Control System begins with the registration of the physical resource. We use the terms *resource* and *physical device* such that a resource is the collection of physical devices associated with a single parent node. *Resource registration* assigns a resource identity to the resource's parent node and is necessary before any of the other protocol processes take place. These processes are listed in Table I.

The owner of a resource registers its parent node P using the Resource Registration process, and subsequently may add any required child nodes $C_{i,\, i>0}$ using the *Child Node Addition* process. There is no limit to the number of child nodes that may be added to a resource. Child nodes no longer needed may be deleted using the *Child Node Deletion* process. Only resource owners may add or delete child nodes.

Users may delegate or assign resource access rights to other users by means of the *Access Rights Modification* process. A user does not have to be the resource owner to transfer or delegate child node access rights to another user. Any user may initiate the *Access Rights Modification* process and transfer all or a subset of his access permissions to another user without the need for supervisory interaction. The protocol prevents *Elevation of Privilege* attacks, preventing users from 'transferring' access permissions greater than their own.

Users no longer required or allowed to actuate a given device may be deleted by means of the *User Deletion* process. Note that deletion of a user applies only to the specific child node from which they have been deleted, and does not affect that user's ability to access any other child nodes that they have

been assigned to manage. This allows users to be re-assigned to manage new devices as needed. Only resource owners are authorized to delete users.

The physical devices that are controlled by individual child nodes may be actuated by authorized users using the *Child Node Actuation* process. A user may receive node actuation permissions on a per command basis either from the resource owner or from another user with approved access to the same command for the same child node.

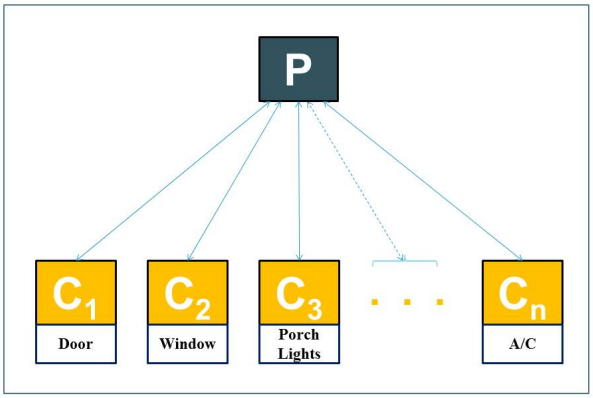

Figure 3 – Example of UPACS with n child nodes

The Universal Physical Access Control Protocol is extendible, allowing for an unlimited number and type of devices that could be controlled, and for the configuration of each child node to satisfy the unique requirements of the physical device that it is tasked to control, thereby ensuring sustained adequacy in meeting the requirements necessitated by future technological development. Prior to the addition of a child node, the node is a virgin embedded device capable of being custom configured to behave as required by the creating user. The child node addition process could then be used to download a custom state machine to the device, which allows the protocol to be used to accommodate future technological requirements.

A. *Mutual Authentication BetweenUser And Parent*

The resource owner U_0 (which could be a person or a process) has an asymmetric key pair: a secret key skU_0 and a public key $pk(sKU_0)$. Similarly the resource's parent node P has an asymmetric key pair: secret key skP, and public key $pk(skP)$ which is known to U_0.

Prior to executing any of the service aspects of the protocol, users and parents have to be mutually authenticated to each other, making use of the trusted key server S. To accomplish this, U_0 first sends a request for P's public key to key server S.

- $U_0 \rightarrow S$: $request(pk(skP))$

S returns P's identity and public key, signed with its secret key skS:

- $S \rightarrow U_0$: $\text{sign}((P, pk(skP)), skS)$

U_0 generates a fresh nonce N_{U0} and sends its identity and nonce to P, encrypted with P,'s public key.

- $U_0 \rightarrow P$: $encrypt((U_0, N_{U0}), pk(skP))$

P decrypts the message to recover U_0 and N_{U0} and sends a request to key server S for U_0's public key:

- $P \rightarrow S$: $request(pk(sk\ U_0))$

S returns U_0's identity and public key, signed with its secret key skS:

- $S \rightarrow U_0$: $\text{sign}((U_0, pk(sk\ U_0)), skS)$

P generates a fresh nonce N_P and sends N_P, N_{U0} and its identity to U_0, encrypted with U_0's public key:

- $P \rightarrow U_0$: $encrypt((N_P, N_{U0}, P), pk(skU_0))$

U_0 decrypts the message and if the message contains its nonce it knows it is communicating with the right resource parent. It then sends back P's nonce N_P along with its request for additional protocol services, encrypted with its own nonce N_{U0}. When P receives and decrypts this message, if the message contains its nonce N_P then mutual authentication is complete and P will process the U_0's request:

- $U_0 \rightarrow P$: $sencrypt((RREQ, N_P), N_{U0})$

B. *Resource Registration*

From location L_0, U_0 sends parent node P's nonce N_P and a request for resource registration $RREQ$ to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment $RREQ\_Ack$ from P:

- $U_0 \rightarrow P$: $sencrypt((RREQ, N_P), N_{U0})$
- $P \rightarrow U_0$: $sencrypt((U_0, RREQ\_Ack), N_P)$

Upon receiving P's acknowledgment $RREQ\_Ack$ U_0 sends to P its location L_0 and the current timestamp T_0, encrypted with its nonce N_{U0}.

- $U_0 \rightarrow P$: $sencrypt((L_0, T_0), N_{U0})$

P decrypts the message with U_0's nonce to retrieve L_0 and T_0, which it then uses to compute its resource identity $ResourceID$ by encrypting L_0, T_0 and N_{U0} with its own nonce N_P.

P registers U_0 as the owner of resource $ResourceID$, stores its identity $ResourceID$ in persistent memory and sends $ResourceID$ and confirmation of successful registration $RREQ\_Confirm$ to U_0, encrypted with N_P.

- $P \rightarrow U_0$: $sencrypt((RREQ, RREQ\_Confirm, ResourceID), N_P)$

U_0 decrypts the message with N_P to recover $RREQ\_Confirm$ and $ResourceID$, which it records as the identity of the newly registered resource.

C. *Child Node Addition*

As many child nodes $C_{i,\ i>0}$ as are required may be added to a resource after its parent node P has been initialized with its secret identity $ResourceID$. Each child node C_i can be located anywhere a network connection can be established between itself and P. Any user U_x may attempt to add a child node C_i as follows:

U_x sends parent node P's nonce N_P and a request for child node addition *AREQ* to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment *AREQ_Ack* from P:

- $U_x \rightarrow P$: *sencrypt*((*AREQ*, N_P), N_{Ux})
- $P \rightarrow U_x$: *sencrypt*((U_x, *AREQ_Ack*), N_P)

Upon receiving P's acknowledgment *AREQ_Ack* U_x sends to P the resource's identity *ResourceID* encrypted with its nonce N_{Ux}. U_x also sends to P all the data necessary to initialize the new child node C_i.

- $U_x \rightarrow P$: *sencrypt*((*ResourceID*), N_{Ux})
- $U_x \rightarrow P$: *sencrypt*(($Perm_{Uxij}$=*True*, Cmd_{ij}, Act_{ij}), N_{Ux}), $j > 0$ for all commands j that child node C_i can execute, where $Perm_{Uxij}$ is the Boolean permission of user U_x to issue command Cmd_{ij} and Act_{ij} is the action that child node C_i will perform upon receiving command Cmd_{ij}.

P decrypts the messages with U_x's nonce to retrieve *ResourceID* and all node initialization data ($Perm_{Uxij}$, $Cmd_{ij,}$ Act_{ij}), j > 0 and if U_x is the registered owner of *ResourceID* then P generates a new random child node identity $cNode_i$ to be the identity of the new child node C_i and initializes C_i with all ($Perm_{Uxij}$, $Cmd_{ij,}$ Act_{ij}), j > 0.

P then sends *ResourceID*, $cNode_i$ and confirmation of successful child node addition *AREQ_Confirm* to U_x, encrypted with N_P.

- $P \rightarrow U_x$: *sencrypt*((*AREQ*, *AREQ_Confirm*, *ResourceID*, $cNode_i$), N_P)

P also adds a row for each command $Cmd_{ij, j>0}$ to the resource's access control matrix acM_{Uij} = [$U_{x,x>0}$ $cNode_{i,i>0}$ $Perm_{Uxij,i>0,j>0}$ $Cmd_{ij,i>0,j>0}$]:

[U_x $cNode_i$ $Perm_{Uxij}$=*True* Cmd_{ij}]

For commands $Cmd_{ij,\ j=1,m}$ P computes the resource's new access control matrix acM'_{Uij} by vertically concatenating rows [U_x $cNode_i$ $Perm_{Uxij}$=*True* Cmd_{ij}] to acM_{Uij} as in Fig. 4.

$$acM'_{Uij} = acM_{Uij} \,//\, \begin{pmatrix} U_x & cNode_i & True & Cmd_{i1} \\ U_x & cNode_i & True & Cmd_{i2} \\ & & \vdots & \\ U_x & cNode_i & True & Cmd_{im} \end{pmatrix}$$

Fig. 4. Effect of child node addition on resource access control matrix

U_x decrypts the message with N_P to recover *AREQ_Confirm*, *ResourceID* and $cNode_i$, which it records as the identity of the new child node C_i.

D. *Child Node Deletion*

Any child node C_i may be deleted by the registered owner of the resource to which it belongs. It may not be deleted by any other user. Any user U_x may attempt to delete a child node as follows:

U_x sends parent node P's nonce N_P and a request for child node deletion *DREQ* to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment *DREQ_Ack* from P:

- $U_x \rightarrow P$: *sencrypt*((*DREQ*, N_P), N_{Ux})
- $P \rightarrow U_x$: *sencrypt*((U_x, *DREQ_Ack*), N_P)

Upon receiving P's acknowledgment *DREQ_Ack* U_x sends to P the resource's identity *ResourceID* and the identity of the child node to be deleted $cNode_i$, encrypted with its nonce N_{Ux}.

- $U_x \rightarrow P$: *sencrypt*((*ResourceID*, $cNode_i$), N_{Ux})

P decrypts the message with U_x's nonce to retrieve *ResourceID* and $cNode_i$ and deletes the node if $cNode_i$ belongs to resource *ResourceID* and U_x is the registered owner of resource *ResourceID*.

P removes all rows [X =$cNode_i$ X X] from the resource's access control matrix acM_{Uij}.

[X =$cNode_i$ X X] = [].

P then sends *ResourceID*, $cNode_i$ and confirmation of successful child node deletion *DREQ_Confirm* to U_x, encrypted with N_P.

- $P \rightarrow U_x$: *sencrypt*((*DREQ*, *DREQ_Confirm*, *ResourceID*, $cNode_i$), N_P)

U_x decrypts the message with N_P to recover *DREQ_Confirm*, *ResourceID* and $cNode_i$, which it records as being deleted.

E. *Access Rights Modification*

Any user U_x may transfer all or any subset of his resource access rights to another user U_z subject to the prior access permissions of both U_x and U_z.

Rights transfers done on any given child node C_i do not affect U_x's permissions $Perm_{Uxij}$ to issue any of the j commands that can be issued to child node C_i. However, U_z's permission $Perm_{Uzij}$ to issue any given command Cmd_{ij} to child node C_i will be updated by the request. Neither U_x nor U_z has to be the owner of the resource to which child node C_i belongs.

User U_x may attempt to transfer all or a subset of his access permissions $Perm_{Uxij}$ for child node C_i to user U_z as follows:

U_x sends parent node P's nonce N_P and a request for access rights modification *TREQ* to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment *TREQ_Ack* from P:

- $U_x \rightarrow P$: *sencrypt*((*TREQ*, N_P), N_{Ux})
- $P \rightarrow U_x$: *sencrypt*((U_x, *TREQ_Ack*), N_P)

Upon receiving P's acknowledgment *TREQ_Ack* U_x sends to P the resource's identity *ResourceID*, the identity of the child node to be affected by the transfer $cNode_i$ and the user to which the rights transfer is intended U_z, encrypted with its nonce N_{Ux}. U_x also sends to P the j child node commands Cmd_{ij} of child node C_i to be affected by the transfer and the requested permissions R_{ij} for U_z to issue commands Cmd_{ij}.

- $U_x \rightarrow P$: *sencrypt*((*ResourceID*, $cNode_i$, U_z), N_{Ux})

- $U_x \rightarrow P$: *sencrypt*((R_{ij}, Cmd_{ij}), N_{Ux}), $j > 0$ for all commands j that child node C_i can execute, where R_{ij} is the requested Boolean permission of user U_z to issue command Cmd_{ij}.

P decrypts the message with U_x's nonce to retrieve *ResourceID*, $cNode_i$, and U_z, retrieves from storage all of U_x's $cNode_i$ permissions Y_{ij} and any prior U_z permissions Z_{ij} for $cNode_i$, and computes the new permissions P'_{Uzij} for U_z to issue commands Cmd_{ij} to C_i as:

$$P'_{Uzij} = Y_{ij} R_{ij} + Z_{ij}$$

If U_z had prior C_i access permissions Z_{ij}, P updates the resource's access control matrix acM_{Uij} for all rows [$=U_z$ $=cNode_i$ Z_{ij} $= Cmd_{ij}$] to [$=U_z$ $=cNode_i$ P'_{Uzij} $= Cmd_{ij}$].

If U_z had no prior C_i access permissions, for commands $Cmd_{ij,\ j=1,m}$ P computes the resource's new access control matrix acM'_{Uij} by vertically concatenating rows [U_z $cNode_i$ P'_{Uzij} Cmd_{ij}] to acM_{Uij} as in Fig.5.

$$acM'_{Uij} = acM_{Uij} \,//\, \begin{pmatrix} U_z & cNode_i & P'_{Uzij} & Cmd_{i1} \\ U_z & cNode_i & P'_{Uzij} & Cmd_{i2} \\ & & \vdots & \\ U_z & cNode_i & P'_{Uzij} & Cmd_{im} \end{pmatrix}$$

Fig. 5. Effect of access rights modification on resource access control matrix when U_z had no prior access permissions for node $cNode_i$

F. *User Deletion*

The permissions of any user U_z to issue commands to a child node C_i with identity $cNode_i$ may be revoked by the owner of the resource to which C_i belongs in a process called user deletion. Only the resource owner may delete a user. A resource owner U_x may delete a user U_z as follows:

U_x sends parent node P's nonce N_P and a request for user deletion *DUSR* to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment *DUSR_Ack* from P:

- $U_x \rightarrow P$: *sencrypt*((*DUSR*, N_P), N_{Ux})
- $P \rightarrow U_x$: *sencrypt*((U_x, *DUSR_Ack*), N_P)

Upon receiving P's acknowledgment *DUSR_Ack* U_x sends to P the resource's identity *ResourceID*, the identity $cNode_i$ of the child node to be affected by the deletion, and the user to be deleted U_z, encrypted with its nonce N_{Ux}.

- $U_x \rightarrow P$: *sencrypt*((*ResourceID*, $cNode_i$, U_z), N_{Ux})

P decrypts the message with U_x's nonce to retrieve *ResourceID*, $cNode_i$, and U_z, and deletes the user if $cNode_i$ belongs to resource *ResourceID* and U_x is the registered owner of resource *ResourceID*.

P removes all rows [$=U_z$ $=cNode_i$ X X] from the resource's access control matrix acM_{Uij}:

[$=U_z$ $=cNode_i$ X X] = [].

P then sends *ResourceID*, $cNode_i$, U_z and confirmation of successful user deletion *DUSR_Confirm* to U_x, encrypted with N_P.

- $P \rightarrow U_x$: *sencrypt*((*DUSR*, *DUSR_Confirm*, *ResourceID*, $cNode_i$, U_z), N_P)

U_x decrypts the message with N_P to recover *DUSR_Confirm*, *ResourceID*, $cNode_i$, and U_z and records U_z as being deleted with regard to $cNode_i$.

G. *Resource Actuation*

Any user U_x with may issue commands Cmd_{ij} to any child node C_i and C_i will process the command if U_x's permission to issue command Cmd_{ij} is set to *True*. The process for issuing commands to a child node is as follows:

U_x sends parent node P's nonce N_P and a request for child node actuation *ACTC* to parent node P, encrypted with its own nonce N_{U0} and waits for acknowledgment *ACTC_Ack* from P:

- $U_x \rightarrow P$: *sencrypt*((*ACTC*, N_P), N_{Ux})
- $P \rightarrow U_x$: *sencrypt*((U_x, *ACTC _Ack*), N_P)

Upon receiving P's acknowledgment *ACTC_Ack* U_x sends to P the resource's identity *ResourceID* and the identity $cNode_i$ of the child node to be actuated, encrypted with its nonce N_{Ux}. U_x also sends to P one or more commands $CmdReq_{ij}$ for child node C_i.

- $U_x \rightarrow P$: *sencrypt*((*ResourceID*, $cNode_i$), N_{Ux})
- $U_x \rightarrow P$: *sencrypt*(($CmdReq_{ij}$), N_{Ux}), $j > 0$

P decrypts the messages with U_x's nonce to retrieve *ResourceID*, $cNode_i$ and all commands to be executed $CmdReq_{ij}$, j > 0 and for each command $CmdReq_{ij}$ if $CmdReq_{ij}$ is a valid command for node $cNode_i$ and U_x has permission of *True* for command $CmdReq_{ij}$ then P sends command $CmdReq_{ij}$ to $cNode_i$ for execution.

P then sends *ResourceID*, $cNode_i$ and confirmation of successful command delivery *ACTC_Confirm* to U_x, encrypted with N_P.

- $P \rightarrow U_x$: *sencrypt*((*ACTC*, *ACTC_Confirm*, *ResourceID*, $cNode_i$), N_P)

U_x decrypts the message with N_P to recover *ACTC_Confirm*, *ResourceID* and $cNode_i$.

IV. CONCLUSION

The Universal Physical Access Control protocol (UPACS) enables secure access to physical resources that need to be protected with access restrictions.

Physical devices are actuated by UPACS child nodes, which themselves are protected from unauthorized access by UPACS parent nodes. Both parent nodes and child nodes may be located anywhere there is network access.

The protocol protects messages and secret information exchanged over an untrusted communication channel between users and parent nodes, both of which rely upon the services of a trusted key server for mutual authentication.

The protocol supports registration of new resources for access to users, addition and deletion of child nodes by resource owners, transfer of access rights between users, and revocation of users' rights to issue commands to selected child nodes by the owner of the resource to which the child node belongs. Users may issue commands to any child node of any resource to which they have access permissions.

Future work will be to develop a formal verification of the UPACS protocol, to verify the protocol's ability to withstand various security attacks, and to implement the protocol as a proof of concept and demonstrate its capabilities in dealing with concurrency issues and solving known security problems.

REFERENCES

[1] Y. Shu, Y. Gu and J. Chen, "Sensory-data-enhanced authentication for RFID-based access control systems," in *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, 2012.

[2] Certicom, "An Elliptic Curve Cryptography (ECC) Primer," June 2004. [Online]. Available: http://www.certicom.com/pdfs/WP-ECCprimer_login.pdf. [Accessed February 2014].

[3] A. Eng and L. Wahsheh, "Look into My Eyes: A Survey of Biometric Security," in *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, 2013.

[4] G. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007.

[5] H.-Y. Chien and J.-K. Jan, "An integrated user authentication and access control scheme without public key cryptography," in *Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on*, 2003.

[6] C. Yang, W. Ma, B. Huang and X. Wang, "Password-Based Access Control Scheme with Remote User Authentication Using Smart Cards," in *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, 2007.

[7] C. Yang, Z. Jiang and J. Yang, "Novel Access Control Scheme with User Authentication Using Smart Cards," in *Computational Science and Optimization (CSO), 2010 Third International Joint Conference on*, 2010.

[8] H. Tsague, F. Nelwamondo and N. Msimang, "An Advanced Mutual-authentication Algorithm Using 3DES for Smart Card Systems," in *Cloud and Green Computing (CGC), 2012 Second International Conference on*, 2012.

[9] Y. Shu, Y. Gu and J. Chen, "Dynamic Authentication with Sensory Information for the Access Control Systems," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 25, no. 2, pp. 427-436, Feb 2014.

[10] G. Shen and B. Liu, "Research on Embedding ECC into RFID Authentication Protocol," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, 2012.

[11] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach and S. Devadas, *Robust and Reverse-Engineering Resilient PUF Authentication and Key-Exchange by Substring Matching,* vol. PP, 2014, pp. 1-1.

[12] K. Rosenfeld, E. Gavas and R. Karri, "Sensor physical unclonable functions," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, 2010.

[13] A. Maiti, L. McDougall and P. Schaumont, "The Impact of Aging on an FPGA-Based Physical Unclonable Function," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, 2011.

[14] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, 2004.

[15] H. J. Lee, H. S. Kim and K. S. Park, "A study on the reproducibility of biometric authentication based on electroencephalogram (EEG)," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, 2013.

[16] O. V. Komogortsev and C. D. Holland, "Biometric authentication via complex oculomotor behavior," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, 2013.

[17] S. Eiroa, J. Castro, M. Martinez-Rodriguez, E. Tena, P. Brox and I. Baturone, "Reducing bit flipping problems in SRAM physical unclonable functions for chip identification," in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, 2012.

[18] Y.-C. Chen and L.-Y. Yeh, "An Efficient Authentication and Access Control Scheme Using Smart Cards," in *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, 2005.

[19] F. Bhutta, A. Ghafoor and S. Sultan, "Smart phone based authentication and authorization protocol for SPACS," in *High Capacity Optical Networks and Enabling Technologies (HONET), 2012 9th International Conference on*, 2012.

BIOGRAPHY:

Clyde Carryl received a B.S. in electrical engineering from Howard University and a M.S. in computer engineering from Florida Atlantic University. He is currently working towards the Ph.D. degree in computer engineering at Florida Atlantic University. His research interests include physical access control systems, embedded systems security, software defined networking security, and dedicated short-range communications. He is a member of the Tau Beta Pi Engineering Honor Society and IEEE.

Dr. Bassem Alhalabi's primary research is the development of pragmatic industrial, consumer, medical, and educational systems with emphasis on Embedded Systems, Web-based and Smart Controls, and Distance Education & Remote Labs. He Co-founded the CADET research center in 1999, and has been co-directing/ directing it since. Through his private consulting company, Dr. Alhalabi works with inventors on their feasibility study, design specification, system architecture and integration, prototypes and proof of concepts, and design for production. Dr. Alhalabi received a BS and an MS in electrical engineering from Ohio University and Purdue University, respectively, and an MS and a PhD in computer engineering from the University of Louisiana at Lafayette. He holds a US patent and others are pending. He is a member of IEEE and various other professional and honor organization, and a recipient of various academic awards.