Track Mobile Learning with Secure Access Using xAPI and CAC

Paul Miller
LSI, Inc.
Jacksonville, FL
pmiller@lsijax.com

Ilya Voloshin
LSI, Inc.
Jacksonville, FL
ivoloshin@lsijax.com

ABSTRACT

Delivering learning content to mobile devices has presented unique challenges not experienced in a traditional Learning Management System (LMS) environment. Identifying the student, securing student progress, presenting content offline, and reporting progress in a secure way are just some of the obstacles preventing mobile content delivery standardization.

Two technologies, when used in conjunction, can potentially solve these problems. They are Common Access Card (CAC) and Experience Application Programming Interface (xAPI). The Navy conducted an experimental project intended to securely record mobile learning events to a Learning Record Store (LRS) via xAPI statements using the CAC provided to United States Defense personnel and DoD learners. This process presented three challenges, with the first being to integrate a compatible CAC reader with Microsoft Surface, iOS (iPad), and Android native applications. CACs securely identify a student and provide encryption tools to encrypt data on mobile devices so only the user with the encrypting CAC can decrypt/access the data.

The next challenge was to securely transfer verified data to the LRS. The xAPI standard provides a way to sign each statement with a cryptographic signature, allowing the LRS to independently verify the integrity of each statement at any time. We used the same cryptographic certificates from the student's CAC to securely sign each statement. The final challenge was to provide security while sending the statements to the LRS. The LRS implemented SSL client certificate authentication to allow access to send the statements. We again used the student's CAC certificate to gain access to the LRS endpoint to securely transmit the data.

This paper details lessons learned from each aspect of this project, from identifying the student to securely transmitting the data. We successfully brought the secure CAC infrastructure to xAPI solving the problem of secure mobile content tracking and delivery.

ABOUT THE AUTHORS

Paul Miller is the Director of Application Development at LSI, Inc in Jacksonville, FL and is responsible for leading the Application Development Team (ADT) to develop and support the tools used by LSI training product development teams. He received his B.S. in Management of Information Technology from Grace College. His current research focuses on the development of desktop, web, and mobile software related to development, review, delivery, and tracking of eLearning content. Throughout his years at LSI, he has gained expertise in the field of tracking student data, which began with in-depth implementation of the SCORM standard and has evolved to include extensive experience with the Experience API (xAPI). Past projects include developing a mobile platform for delivering web-based content via packaged native applications for both Apple iOS (iPad) and Android tablets used by LSI's US Army and US Navy customers which allow for a SCORM package to easily transfer to a native tablet app with built-in xAPI reporting capabilities.

Ilya Voloshin is the Software Architect of Application Development at LSI, Inc in Jacksonville, FL and is responsible for software architecture, design, and implementation of n-tier applications and service-oriented architectures, as well as identifying and implementing technical standards, including coding standards, tools, and platforms. He received his B.S. in Computer Information Science & Business Management from the University of North Florida and is a Microsoft Certified Solution Developer. His primary research is concentrated on learning technologies, designing and developing distributed courseware development tools such as Kreus, SCO Workbench and Venus 2 Enterprise. These tools included SCORM output versions 1.2 and 2004 editions with complex sequencing and navigation rules as well as the new xAPI standard. Additionally, Ilya implemented the xAPI specification for an LSI developed LRS, called MiLRS. Ilya was the software architect for the mobile CAC/xAPI project described in this paper, concentrating on CAC architecture, encryption and use in xAPI statements.

Track Mobile Learning with Secure Access Using xAPI and CAC

Paul Miller
LSI, Inc.
Jacksonville, FL
pmiller@lsijax.com

Ilya Voloshin
LSI, Inc.
Jacksonville, FL
ivoloshin@lsijax.com

INTRODUCTION

Modern technologies in the training industry are driving students away from desktop computers and laptops and pushing them to smaller factor devices such as smartphones and tablets. The Department of Defense (DoD) relies heavily on Learning Management Systems (LMSs) to provide eLearning content to students while tracking student performance through lesson completion and assessments. An essential component of this process involves student identity and integrity of the data collected. Delivering learning content to mobile devices has presented unique challenges not experienced in a traditional LMS environment. Identifying the student, securing student progress data, and reporting progress in a secure way are just some of these challenges preventing DoD standardization of mobile content delivery. This paper will describe a project which combined two technologies to potentially solve these challenges. The technologies are Common Access Card (CAC) and Experience Application Programing Interface (xAPI).

We will begin by outlining the various technologies playing a part in the field of DoD eLearning and obstacles related to these technologies affecting mobile content delivery.

Shareable Content Object Reference Model (SCORM)

In 1999, Executive Order 13111 required the U.S. DoD to create a common set of standards for eLearning, which resulted in the first version of SCORM in January 2000 (Lundy, 2003). SCORM provides XML schemas defining an eLearning lesson's structure and metadata, it also provides a JavaScript Application Programming Interface (API) to allow communication between an LMS and the training content.

The SCORM standard is still used widely across DoD eLearning. SCORM package conformance is maintained by the Advanced Distributed Learning (ADL) Initiative, a US Government program that conducts research and development on distributed learning. The ADL Co-Lab provides SCORM conformance tools targeting content developers and LMS providers to ensure eLearning will display and communicate properly on a SCORM LMS.

Learning Management System (LMS)

An LMS is a web-based application for administration, tracking, reporting, and delivery of eLearning courses. Different LMSs provide a variance of features for user roles, communication, enrollment, and a vast number of other components. For the purposes of this paper, the features of note are common across all SCORM conformant LMSs, which are user identification, hosting of eLearning content, and SCORM based student performance tracking.

Because SCORM uses a JavaScript based API, SCORM always requires a web browser as the medium for consumption. This inherently limits training to HTML web content. There are benefits to this limitation. For instance, an LMS can easily control access to content via user accounts either through username/password or CAC authentication, which all major web browsers support out of the box. Most LMSs also utilize HTTPS, which is a secure extension of the Hypertext Transfer Protocol (HTTP). These well-established practices make SCORM and LMS the standard for delivering eLearning content to the DoD. Other training types, such as native mobile applications, training devices, and desktop simulators are not able to utilize SCORM and therefore do not gain the benefit of the LMS features.

Experience API (xAPI)

The development of the xAPI specification began with Rustici Software's Tin Can Project in 2011. The terms *xAPI* and *Tin Can API* are often used interchangeably. To combat the limitations previously mentioned of SCORM web-based content, the xAPI specification was created to provide user tracking capabilities to all types of training devices and media. It was, and is considered the successor to SCORM; however, primarily due to challenges addressed in this paper, xAPI is not as prevalent as we would like it to be in DoD training.

Rather than using web-based JavaScript technology, the xAPI utilizes Representational State Transfer (REST) and HTTP, or RESTful HTTP requests. This method allows for any device or application with a networked connection to send an HTTP request to a Learning Record Store (LRS) for tracking. The two primary items sent via this method are "xAPI statements" and "xAPI documents." While the SCORM specification contains a strict schema for trackable items, an xAPI statement simplifies a trackable event in the form of "actor verb object." This open schema of xAPI allows for a nearly limitless range of trackable data.

Learning Record Store (LRS)

An LRS can be considered the "server" and/or "database" for the xAPI specification. While SCORM requires a SCORM conformant LMS to track student data, xAPI uses an LRS. An LRS does not replace an LMS. The two are not mutually exclusive. An LRS is only for receiving and querying xAPI data (statements and documents). While it does provide user and application security for authenticating statements, it does not provide the features inherent to a typical full LMS, such as user management, curriculum management, and lesson assignment. Most major LMS providers now include an LRS within the application. The project described within this paper utilized an experimental LRS integrated with the Navy eLearning (NeL) LMS.

Common Access Card (CAC) Certificates

The US DoD provides all active duty personnel with an identification CAC. One of the functions of a CAC is to provide two-factor authentication for a user. The two factors are the physical card itself and the PIN to unlock the encrypted digital certificate on the card. CACs are a form of Smart Card technology. They use a public key infrastructure (PKI). The DoD provides a PKI to maintain the certificate lifecycle, including issuance, suspension, and revocation (Department of Defense [DoD], 2011). Any application or website intending to use the PKI for authentication, digital signatures, or encryption must validate the Public Key (PK) against the DoD's authority providers.

Two Factor Authentication

Two-factor authentication is a subset of multi-factor authentication and is deemed more secure than simple username and password authentication of a user. Two factor authentication consists of a method of confirming users' claimed identities by using a combination of two different factors—something that they have (such as a CAC card) and something that they know (such as a PIN to access the CAC).

THE EXPERIENCE API AND CAC PROJECT

In 2016, the United States Navy (USN) contracted LSI to develop Web Based Training (WBT) content and deliver it to four platforms: desktop browsers, Microsoft Surface 2 tablets, Android tablets, and iOS tablets (iPads). The objective of the contract was experimental. The USN chose a basic military training course to be delivered via multiple tablet options to see if any of the three tablets stood out over the others as more efficient or easier to use. The requirement was for the three tablet versions to be native mobile apps rather than simply making the WBT accessible via the built-in mobile web browsers. With this requirement, the WBT was to be available in an offline environment. The results of the tablet evaluation were not disclosed to LSI and is not the focus of this paper.

In all four environments, the content must track two metrics of student performance: usage time by section and answers to assessment questions. For the desktop browsers, these metrics are easily tracked through the well-established Navy eLearning (NeL) LMS via SCORM. The native apps however, must utilize xAPI statements sent to the newly developed NeL LRS. Since the content must be available in an offline environment, the mobile apps were to store the metrics until internet connectivity could be established. At that time, the student would manually choose to

synchronize the metrics to the LRS. The specific metrics to be tracked are not significant to this paper. The methods described will work for any metric collected via the mobile applications.

The NeL LMS requires a student to authenticate through a web browser using a DoD furnished CAC for identification. While not initially stated, USN and LSI worked together to establish the inherent security of CAC would also be required for authentication on the mobile apps and be passed through to the LRS with the xAPI statements.

Identifying the Student on Mobile Applications

It is always challenging to securely identify a user in a disconnected environment where the application has no way to externally verify an identity. Local account storage is limited to the device and cannot enforce things like limiting users to single accounts or prevent password sharing. CAC infrastructure, with its built in two-factor authentication process and central authority issuance provides a secure and trusted way to identify a user to a disconnected system. CAC authentication is secured by preloading the mobile device with a Root and Intermediate Certificate Authority (CA) certificates which validate Client Certificate signatures provided by the user. Since the user is required to enter a PIN to access the Client Certificate on the card, this two-factor authentication method prevents credential sharing. This project did not require user access level control. Any user with a valid CAC could access the content. Future projects may provide further authentication via communication to the LRS or a built-in offline database of authorized users.

To provide CAC access to the mobile device, a CAC reader and software are required.

CAC Reader Hardware

One of the first challenges we encountered during this project was to determine which mobile Smart Card reader was best suited for the iOS and Android devices. We learned quickly that unlike Microsoft Windows PCs, each Smart Card reader required proprietary program code to communicate with it. Windows provides a built-in smart card architecture which is used by all CAC reader manufacturers, so the Windows based mobile tablets were able to utilize this framework. This allowed for any Windows compatible USB smart card reader to work with the Microsoft Surface 2 tablets.

In our search for the best mobile CAC reader, we wanted to find a manufacturer that provides both iOS and Android readers. Our first purchase was the Precise Biometrics Tactivo Smart Card reader (the Tactivo brand is now owned by Identos GmbH). Precise Biometrics provided a library file to include into a mobile app. In our case, we included this library in the iOS and Android apps we were building for this project. This would allow our apps to communicate with the Tactivo CAC reader hardware device directly. While this was a possible solution, it was not viable due to the scope and period of performance of the project. We needed to find a solution with an available Software Development Kit (SDK) for easier integration. We next purchased Thursby Software Smart Card Readers for iOS and Android devices. Thursby provides the Sub Rosa Pro authentication app in the Apple App Store and Android Play Store. The Thursby SDK allowed us to delegate CAC PIN authentication and Client Certificate retrieval from the card to Thursby's Sub Rosa Pro authentication client. Our apps only need communicate with Sub Rosa Pro via the Thursby provided SDK. This significantly reduced the scope of the CAC reader integration effort. Additionally, Thursby supports industry and U.S. government security standards including HSPD-12, OMB-11-11, and FIPS 140-2.

CAC Reader Software Architecture

Every DoD issued CAC contains a unique 10-digit number assigned to personnel registered in the Defense Enrollment and Eligibility Reporting System (DEERS). This number is referred to as the Department of Defense Identification (DoD ID) number. NeL was already utilizing the DoD ID for its LMS users. For this project, we determined the DoD ID should be retrieved from the mobile user's CAC for identification and be sent along with the xAPI statements associated with that user to the NeL LRS. Now that the CAC hardware was selected, we had to modify our iOS and Android mobile applications to utilize the Thursby SDK via the Sub Rosa Pro authentication client to get this identifying information. In addition to the DoD ID, we also determined we should retrieve the user's name for display within the mobile apps.

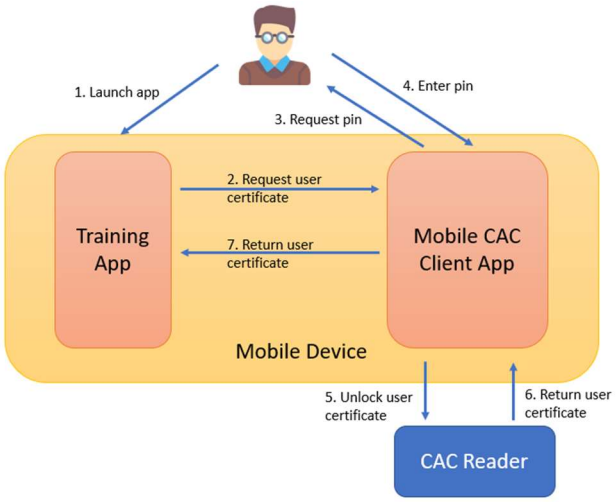

Figure 1. Identifying the Student with CAC Reader

Architecturally, there is a seven-step process to get from launching the mobile app and securely retrieving the user's DoD ID and name (see Figure 1). The user launches the training application (1) and the training application must identify the user by requesting the user's DoD ID and name. To retrieve this information from the user's CAC certificate, the training app needs to send a request (2) via Thursby SDK to Mobile CAC Client App (Sub Rosa Pro). The Mobile CAC Client App then prompts the user for their CAC PIN (3) and the user enters the PIN (4). The PIN grants the Mobile CAC Client App access to the secure storage of the certificates on the CAC (5) which are then returned to the Mobile CAC Client App (6). The Mobile CAC Client App then returns the certificate to the training app (7). The training application now knows who the user is and has the DoD ID information it needs for future tracking and encryption purposes.

Securing Student Progress Data

The next challenge to overcome was ensuring that any data collected from the user's session could not be compromised. Modern mobile operating systems provide isolated storage, limiting external application access to files created by the mobile app. These files, however, can still be accessed by users by plugging the devices into a PC and accessing files directly. To secure user data on the device, we employ a combination of industry standard symmetric encryption algorithms provided by the mobile operating system along with asymmetric encryption. We combine these two technologies using a technique similar to OpenPGP standard (RFC 4880) which defines secure message communication over the Internet.

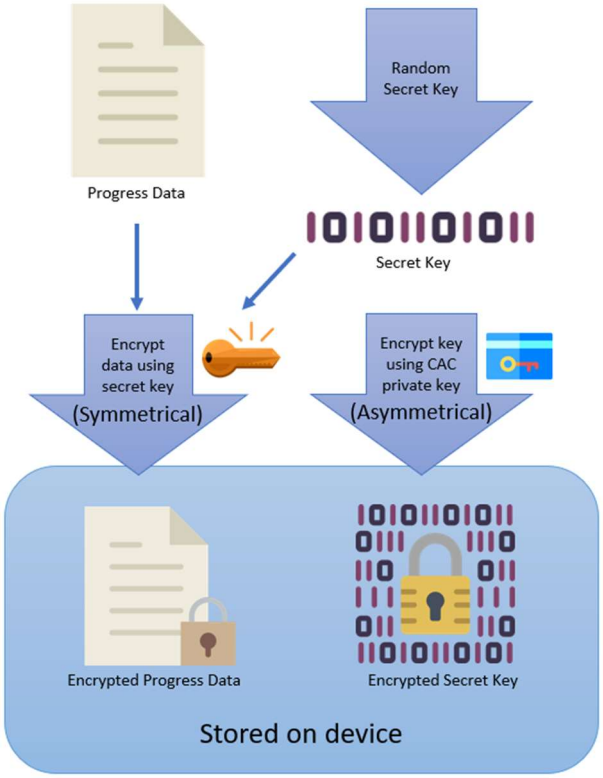

Figure 2. Securing Student Progress Data via Encryption

Symmetric cryptography uses one secret key to encrypt/decrypt the data. The secret key can either be a number, word, or string of random letters and must be owned by both the sender and recipient of the data. The secret key is transferred along with the data it encrypts, making symmetric cryptography less appealing for sending data over the internet. However, symmetric cryptography is more efficient than asymmetric for large amounts of data, due to its increased requirement for storage and processing. Therefore, we used symmetric cryptography to secure the progress data on the device. In order to do this, we must also store and encrypt, the lone secret key on the device as well. We have the progress data secured, but we must now secure the secret key.

Asymmetric cryptography (also known as public-key cryptography) is a system that uses a pair of keys – public key and private key. The public key may be disseminated freely, but the private key is only known to the owner of the data. With this system, any person with access to the public key can encrypt data using the public key, but that data can only be decrypted by the private key. This adds more security than symmetrical cryptography but is less efficient with large amounts of data. Since the only thing left to encrypt on the device was the secret key we used

for symmetrical encryption of the progress data, we used asymmetric cryptography to secure the secret key. Figure 2 illustrates how we achieved this dual encryption method. We generate a random secret key to be used for symmetric encryption of the progress data. Once the user's progress data has been encrypted, we then use an asymmetric cryptography algorithm to encrypt just the secret key and store it on the device alongside the already encrypted progress data. CAC technology uses Public Key Infrastructure (PKI); every user's CAC contains a unique private key only accessible to the owner of the CAC. This is the private key we use to encrypt the symmetric cryptography secret key.

When retrieving the progress data, the process is followed in the reverse. We first decrypt the secret key using asymmetric encryption from the private key on the user's CAC. The secret key is then used to decrypt user's data using symmetric encryption (see Figure 3).

By encrypting the secret key with the private key on the user's CAC, we ensure only the CAC user who owns the stored data can access it on the device. The data cannot be compromised, even if the device is connected to a PC and the file system accessed directly.

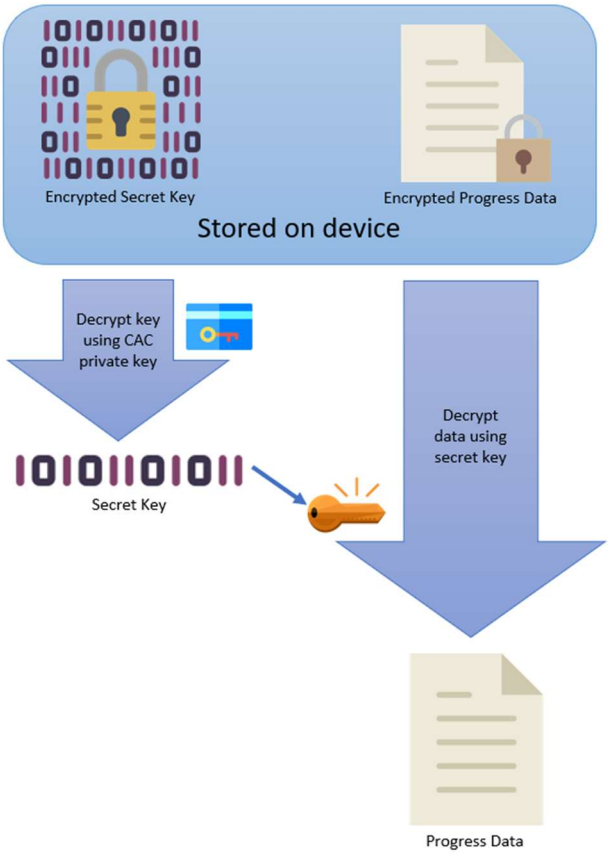

Figure 3. Decrypting Progress Data

Reporting Progress in a Secure Way

Once we had identified the student and secured student data, the final major challenge was to report the progress to the NeL LRS securely. When reporting progress to an LRS there are two things to consider. First, that data authenticity and integrity has not been compromised and second, the user has authorization to submit the data.

Data Integrity

Data integrity is covered by the xAPI specification Part Two Section 2.6, Signed Statements. "Signed Statements include a JSON Web Signature (JWS) as an Attachment. This allows the original serialization of the Statement to be included along with the signature" (Advanced Distributed Learning Co-Laboratories, n.d.).

As discussed in the Securing Student Progress Data section above, a CAC contains a private key unique to the card. The CAC stores this private key in a digital certificate. The digital certificate contains the private key along with other information such as the issuer of the certificate and other metadata. The issuer of all DoD CACs is a Certificate Authority (CA) owned by the DoD itself. During the issuing process, the DoD CA provides a digital certificate containing its own public key, which authenticates a CAC to a trusted DoD authority. This creates a certificate chain that can at any time be validated back to the DoD. This is an important part of achieving data integrity for our project with the Navy. By using the CAC, the NeL LRS can validate the data we send against a trusted authority by following the certificate chain.

A JWS must include a digital signature. We create this digital signature by signing the statement with the certificate from the user's CAC. This allows verification of data at any point, before or after transfer to the LRS, using nothing but the certificate chain included in the statement signature. Successful verification of the signature signifies that the original statement has not been modified in any way since its generation and signing by the original user.

Figure 4 visualizes the process of signature generation. We utilized *hashing* technology to ensure file integrity. Hashing is an algorithm that calculates a fixed-size bit string value from a file. At its most basic form, a file is simply blocks of data. Hashing transforms this data into a far shorter fixed-length value or key which represents the original string. The hash value is a unique mathematical signature of everything within that file. One main use of hashing is to compare two files for equality. Without opening two document files to compare them word-for-word, the calculated hash values of these files will allow the owner to immediately know if they are different (Chung, n.d.).

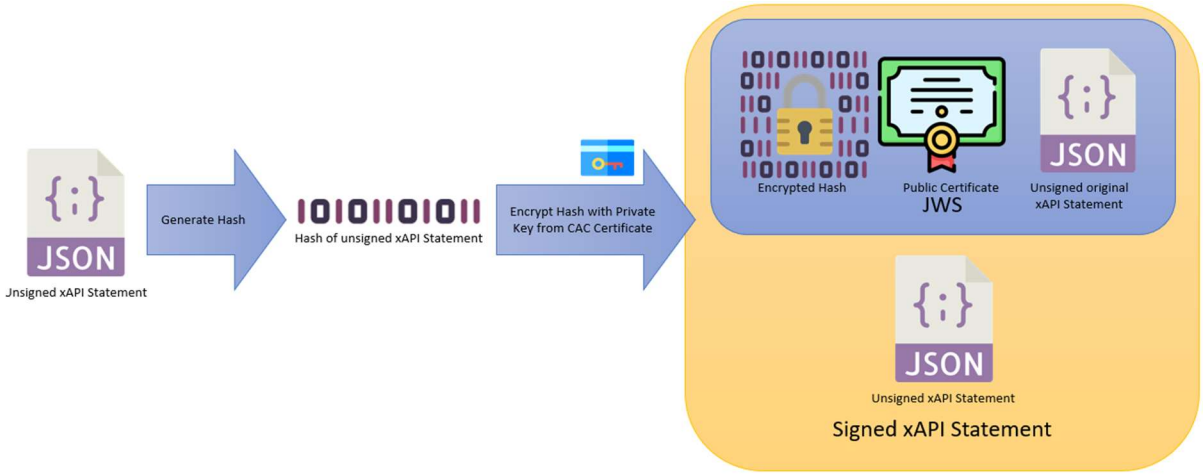

Figure 4. Signing of xAPI Statement

The unsigned xAPI statement in JSON format is passed to a standard hash function that produces a unique hash of the statement. This now provides us with an unencrypted fingerprint of the statement. The hash is then encrypted with the private key from the user's CAC certificate. The encrypted hash is then combined with the public key from the user's CAC and the original unsigned statement. The public key will be used later for decryption. The combination of the three form the JWS. These are then attached to the now signed xAPI statement as it is sent to the LRS. It is required to include the original unsigned statement, because once the statement is submitted to the LRS, the statement will likely get modified by the LRS for internal tracking purposes. This could be a date/time stamp or other metadata. Any change to the statement would negatively affect the hash validation done in later steps.

At this point in our example, the xAPI statement has been signed and sent to the LRS. The NeL LRS, as with most LRSs, does not perform validation prior to accepting the statement. The validation process is intended to be done by the consumer of this statement in the future. For example, the data in the NeL LRS might be retrieved from the NeL LMS to add to a student's training record. The LMS should not blindly accept every statement from the LRS. Instead, it should validate it using the process described in Figure 5. Validation is performed for non-repudiation and to verify the statement has not been modified.

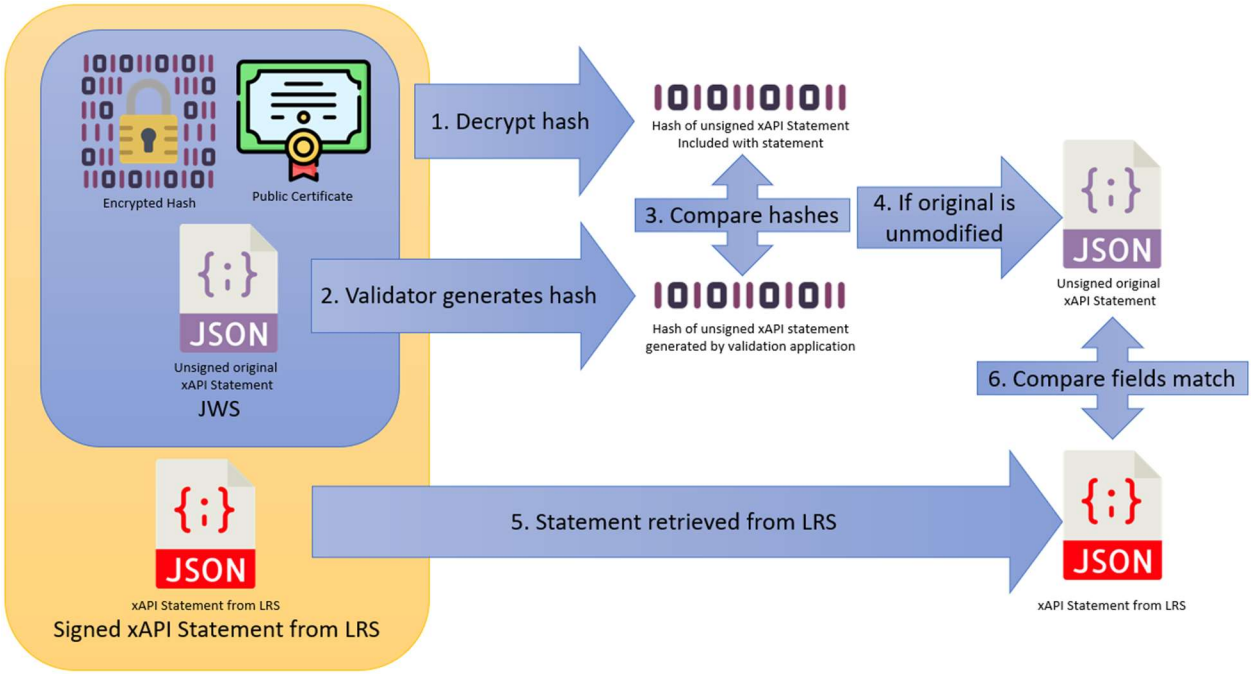

Figure 5. Validation of xAPI Statement

Figure 5 illustrates the process of statement validation done by a consumer of the data (such as the NeL LMS). The left side of the diagram represents the xAPI statement retrieved from the LRS with the JWS xAPI attachment. The JWS contains the encrypted hash, the public certificate (public key), and a copy of the unsigned original xAPI statement. The first step is to verify the integrity of this signature (JWS). The validator decrypts the encrypted hash using the public key provided in the JWS (1). Next, the validator must generate its own hash from unsigned original xAPI statement also included in the JWS (2). The validator then compares the two hashes (3). If they match, we have a valid signature (JWS) and can trust that the original statement has not been modified. The validator now takes the xAPI statement from the LRS (5) and compares the fields to the verified original statement (6). As mentioned above, the LRS may have added metadata to the statement. The xAPI specification, Part Two, Section 2.3.1 outlines which fields need to match to consider the statement identical (Advanced Distributed Learning Co-Laboratories, n.d.).

Mutual Secure Socket Layer (SSL) Authentication

With data integrity resolved for reporting progress, we must now ensure the secure transfer of data from the mobile device to the LRS server. For this process, we once again leveraged CAC provided security (Figure 6). The NeL LRS required us to use Mutual Secure Socket Layer Authentication. The mobile applications begin this process by requesting a Secure Socket Layer (SSL) communication connection (1). The server in turn, sends back a server certificate and a request for a client certificate in return (2). The mobile application client, which has already validated the user via CAC pin, retrieves the certificate from the card and sends it to the NeL LRS server (3). The server now validates the client certificate using the public certificate chain (4). Once authentication is successful, a secure SSL connection is established between the mobile application and the server (5).

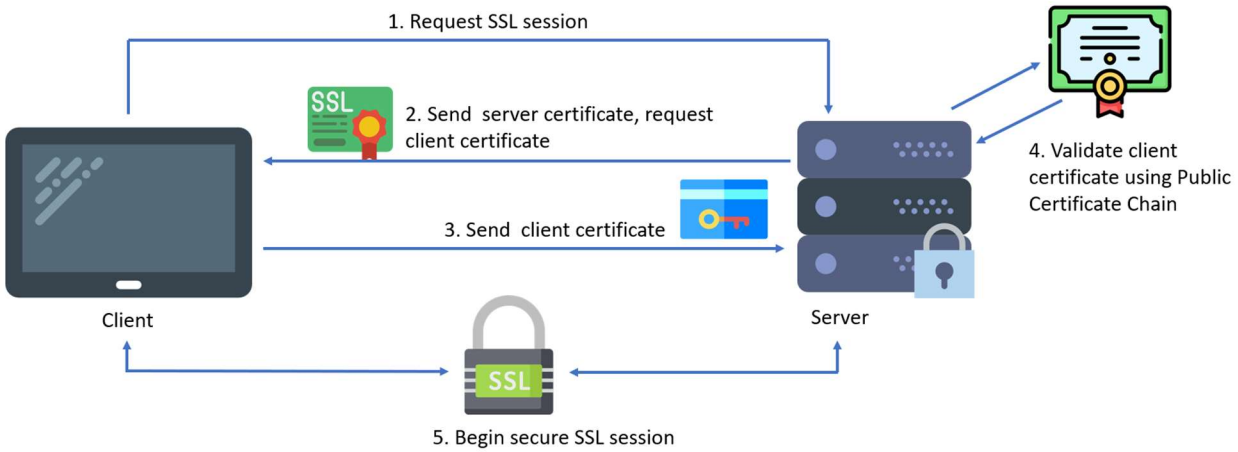

Figure 6. SSL Authentication

THE PATH FORWARD

The methods and procedures described in this paper were successful in securely identifying a student, securely retaining student progress, and securely reporting student tracking data to an LRS. With the technological map laid out for us, we must now look ahead to the next challenges.

One of these challenges is providing an infrastructure to support xAPI. As previously discussed, SCORM based LMSs have a long history in DoD training. The process is well established, and there are many thoroughly vetted LMSs used across the Services. While they are slowly becoming more prominent, LRSs do not share this familiarity.

Another challenge is the xAPI specification itself. Unlike SCORM, the xAPI specification does not define exactly how to track eLearning interactions. This is the job of xAPI profiles. By specification standards, an LRS will only enforce the format restrictions for xAPI statements; it will not regulate the content of the statements nor require certain statements to exist. To fully achieve the level of tracking SCORM provides, the xAPI eLearning profile must get industry approval, and an LRS must be able to enforce xAPI profile rules. Defining and enforcing xAPI profiles is a vast topic outside the scope of the project discussed in this paper but is equally as important.

Finally, a difficult challenge is the consumption of the xAPI data. An LRS is simply a database of user activity and does not perform the functions of a conventional LMS. For the solution described in this paper, the NeL LMS provided user management, data reporting, class management, and mapping of student records retrieved through the LRS to other systems. The ideal solution will be an existing LMS extended with an LRS. Ultimately, something needs to consume the data and display it in a human readable form.

In closing, we will discuss some possible alternatives to using CAC for identification, securing progress data, and data integrity. We will also discuss how our experimental project could affect future training.

Alternative Forms of Authentication

A common alternate form of authentication would be the OpenID Connect/OAuth 2.0 standard. There are multiple ways to achieve this, all of which require an authentication service available while the mobile device is connected to the internet. The benefit of this approach is it allows federated identity management across multiple third-party services like Microsoft Azure, Google, or Facebook. This is far less restrictive than CAC authentication. These standards can provide secure centralized authentication, but they require online access and they outsource security to a non-government owned third-party. Another alternate option is the combination of both CAC infrastructure and OpenID Connect/OAuth 2.0. One of the federated identity options for OpenID could be CAC by requesting a CAC client certificate at login time to validate users. This would provide better security but would also require a mobile CAC reader and online access. While commercial products that combine OpenID and CAC exist, they were determined to be outside the scope of this project. Table 1 below breaks down the differences between the method used in this project (Offline CAC Authentication) and the two discussed in this section.

Table 1. Alternate Forms of Authentication

	Provides PKI	Offline Authentication	Federated Identity
Offline CAC Authentication	YES	YES	
OpenID Connect/OAuth 2.0			YES
CAC + OpenID Connect/OAuth 2.0	(OPTIONAL)	(OPTIONAL)	YES

Effects on Training

There were no performance data sets collected for this project. There also were no metrics calculated to suggest better user engagements or improved performance. This was not an objective of this project. However, other papers and studies have been conducted on this topic, including an I/ITSEC 2011 paper by Jason Haag of ADL. Haag's research showed high levels of satisfaction and mobile preference for his targeted student group. He showed mandatory training could be made more accessible and feel less forced upon, if a mobile alternative is available (Haag, 2011).

REFERENCES

Lundy, J. (2003). SCORM Standard Unites E-Learning Software and Content. Retrieved from https://www.bus.umich.edu/KresgePublic/Journals/Gartner/research/114200/114233/114233.html

Department of Defense. (2011, May 24). Public Key Infrastructure (PKI) and Public Key (PK) Enabling (DoD Instruction 8520.02). Washington, DC: Takai, T.

Advanced Distributed Learning Co-Laboratories. (n.d.). Experience API. Retrieved from https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Data.md

Chung, C. (n.d.) Introduction to Hashing and Its Uses. Retrieved from https://www.2brightsparks.com/resources/articles/introduction-to-hashing-and-its-uses.html

Haag, J. (2011). From eLearning to mLearning: The Effectiveness of Mobile Course Delivery. Retrieved from https://adlnet.gov/adl-assets/uploads/2015/11/e_to_mLearning_paper.pdf