An Automated, Disruption-Tolerant Device Authentication and Key Management Framework for Critical Systems

TW Edgar[1], A Ashok[1], GE Seppala[1], EY Choi[1], KM Arthur-Durett[1], M Engels[1], R Gentz[2], S Peisert[2]

[1]Pacific Northwest National Laboratory
Richland, Washington, United States of America

E-mail: thomas.edgar@pnnl.gov; aditya.ashok@pnnl.gov; garret.seppala@pnnl.gov; eric.choi@pnnl.gov; kristine.arthur-durett@pnnl.gov; matt.engels@pnnl.gov

[2]Lawrence Berkeley National Laboratory
Berkeley, California, United States of America

E-mail: rgentz@lbl.gov; sppeisert@lbl.gov

Abstract: *Key management is critical to secure operation. Distributed control systems, such as Supervisory Control and Data Acquisition (SCADA) systems, have unique operational requirements that make conventional key management solutions less effective and burdensome. This paper presents a novel Kerberos-based framework for automated, disruption-tolerant key management for control system environments. Experimental tests and their results are presented to quantify the expected performance overhead of this approach. Additionally, Zeek sensor analytics are presented to aid in monitoring the health and security of the key management framework operation.*

Keywords: *Key Management, ICS, SCADA, Authentication, Disruption Tolerant, Kerberos*

Introduction

Key management and access control infrastructure are fundamental to building secure systems; however, current key management and trust models were designed for enterprise Information Technology (IT) environments and do not suit the requirements of process and distributed control system environments (Baumeister 2011). These environments are geographically distributed with high-availability requirements that limit the ability of traditional centralised authentication and authorisation mechanisms. Due to operational management difficulties, the lack of a scalable technology to manage cryptographic keys for distributed Energy Delivery Systems (EDSs) hinders asset owners' deployment of products to secure communications. This problem is amplified as more renewable and distributed energy resources emerge and are integrated into the grid, increasing the number and complexity of EDS resources. Without an industry-accepted, scalable, secure, and robust key management, authentication, and authorisation service meeting operational requirements, development of secure cyber-physical applications will be difficult. Industry requires a cryptographic key and access control management solution to further the deployment of technical

solutions and to limit the risk associated with increased communication and functionality of smart grid applications.

Current key management and authorisation frameworks have been built around Internet operations and an always connected state. However, in some environments, the ability to query an online service for every authentication cannot be guaranteed, and the burden of updating and distributing revocation lists is too great. The electric utility industry, among others, needs a solution that provides the ability for distributed, intermittently connected systems to authenticate, while still providing robust centralised policy control and auditing to meet regulatory and best practice guidance. Also, most key management and authentication systems are designed for users and expect human interfaces and interaction. Control systems are designed to operate independently with limited human interaction. Providing automated services that enable devices to receive key material and authenticate each other is necessary for control systems. A new approach is needed that is tailored to the unique aspects of distributed control systems.

While a new protocol could be developed to address these problems, leveraging existing standardised and accepted protocols enables deployment and integration at a much more rapid pace. The Kerberos protocol is a well-established, widely accepted authentication and key management protocol that is already deployed and utilised in most enterprise environments. Through use of a novel architecture and deployment, Kerberos can be leveraged to provide the needed feature set for SCADA environments while providing a wealth of knowledge, experience, and software to support a usable and manageable rollout.

This paper describes an Automated Disruption-Tolerant Key Management (ADTKM) system built upon the Kerberos protocol for distributed automation and other control systems. The ADTKM leverages the unique characteristics of Kerberos for multiple domains of trust to enable centrally controlled authentication and remotely managed authorisation of devices to distribute key material for utilisation in secure applications. The Kerberos ticketing system provides the ability to operate in a disconnected state for a period of time. With some creative utilisation and operation, Kerberos can be the solution needed for this industry. Key management itself is often targeted in attacks; and, as such, developments for monitoring the health and security of the ADTKM approach are also presented. Experimental tests were performed to quantify the cost of this approach and to validate self-monitoring. The experiments, their results, and lessons learned are documented at the end of this paper.

Related Works

Key management and authentication are foundational to security operations. As such, there are various approaches, some well-established and used extensively, for distributed key material and authenticating access. This section provides an overview of the relevant work that has been done for applying key management frameworks to control system environments. The issues with disruption tolerance of common key management techniques is also detailed.

Theoretical models of trust and key management have been developed for varying conditions. When sharing keys, it is crucial to validate the identity of the parties involved in case one party is deceived into sending secure data to the wrong destination. As such, there a variety of ways

identities can be authenticated and trust distributed. The most basic is symmetric key management where trust is evaluated and approved by the communicator on a case-by-case basis and each pair of communicating partners shares the cryptographic material through some mechanism such as manual or key agreement protocols (Piètre-Cambacédès & Sitbon 2008). Secure Shell is an example of symmetric trust where each partner must negotiate accounts and each new server fingerprint must be approved as trusted.

The most common form of trust used is brokered trust, where some chosen authority is selected to bestow and validate identities. Public Key Infrastructure (PKI) is the most common implementation of brokered trust for key distribution. Certificate authorities are the selected central authorities around which PKI works and certificates are bestowed to users with various levels of validated identity. The certified authority then provides authentication of identity for others (Piètre-Cambacédès & Sitbon 2008). Second after PKI is distributed trust, or web of trust, where trust is organically organised through peers validating and authenticating identities (Zimmerman 1994). For example, if Alice trusts Bob, who in turn validates Charlie's identity, Alice can extend that trust to Charlie. Pretty Good Privacy (PGP) is the de facto implementation of web of trust.

Finally, there is the trust-free model where everyone can validate the authenticity of data without validating identities of peers. Blockchain ledgers are an example of distributed trust (Sun, Yan & Zhang 2016). Each model has strengths and weaknesses. Brokered trust enables strong control and enforcement of policy. Distributed trust is flexible and dynamic and obviates the need for identities. These features are valuable or counterproductive depending on each specific use case.

Currently, no key management framework has been accepted or deployed in great numbers across process control environments. International Electrotechnical Commission (IEC) 62351 Part 9 (IEC 2017) is a standard for implementing key management for the IEC 61850 protocol suite (IEC TC57 2019) and is the most formalised approach to key management in industry. Otherwise, there is a lack of deployment of general key management frameworks within SCADA systems. There have been multiple key management frameworks and protocols developed to address various issues within process control. Some address the complexity and performance issues of deploying complex PKI systems (Beaver et al. 2002; Tawde, Nivangune & Sankhe 2015; Ebrahimi, Koropi & Naji 2014; Rezai, Keshavarzi & Moravej 2017). Others provide improvements to create a consistent process across the hierarchy of SCADA communicating devices (Dawson *et al.* 2006) or group key management facilities for specific communication requirements of some protocols (Choi *et al*. 2009; Choi *et al*. 2010; Mittra 1997; Jiang *et al*. 2013). The SSP-21 secure communication protocol (Crain 2017) supports multiple key management approaches but has devised its own modifications to the X.509 certificate format to address some shortcomings with PKI. However, this makes modified certificate non-compliant with the entrenched PKI space and large number of tools provided which add further difficulty to the deployment.

The framework discussed in this paper is focused on solving the challenge of central policy control while enabling remote disconnected operation. Previous designs were developed using new hybrid protocol to achieve the desired feature set (Manz, Edgar & Fink 2010). While the previous work met the functionality requirements design, it required a new, untested protocol with a lack of tool and technology support. Leveraging accepted standard protocols is necessary to increase

operational viability. The objectives of the work documented in this paper were to utilise standards-based solutions in a novel architecture to solve the problem while having readily available tools, expertise, and infrastructure to support deployments.

Process Control Authentication and Key Management Requirements

Process control systems have unique operational characteristics that require additional functionality for a key management solution. EDSs have a hierarchy of communication where many distributed substations must operate independently and coordinate with a master station (Wang & Lu 2013). The distributed systems need to operate even though the communication channels between the master control station and remote substations cannot be assumed to be reliable (Rezai, Keshavarzi & Moravej 2017). There are existing and continuing efforts to instil security into applications in legacy and future distributed field environments such IEC 62351, IEEE 1711, and SPP-21 (Crain 2017). For resiliency, these secure applications operate between devices within field environments and must continue to operate with loss of connections to centralised control environments. Therefore, authentication and key management processes must handle periods of disconnected operation without significantly increasing risk to the system. A second requirement is to reduce management cost and burden. Traditional key management systems, such as PKI, can quickly become difficult to manage at scale (National Institute of Standards and Technology 2010); accepting operational risk is often the path chosen to overcome these challenges. The ADTKM system is tailored to address the unique properties of both the cyber and physical attributes of EDSs to ensure a strong foundation for implementing secure applications. Four feature sets drove the design and execution of the ADTKM system.

First, automating device key management relieves operational burden and increases the appeal of security applications. In enterprise networks, the authentication and key management process generally involves users accessing networked services; however, in control systems, the communication occurs mostly device-to-device. An authentication process for these environments must enable the automated communication establishment and maintenance between device-to-device communications.

Second, while distributed operation is crucial to maintaining stable and secure EDSs, distributing control of the system is labour intensive and resource prohibitive. A fully distributed system would limit the observability and management of operations necessary to fulfil some of the regulatory collection and reporting guidelines of these environments (Critical Infrastructure Protection Committee 2009). Central policy management of authentication and authorisation is necessary, while still providing distributed operation.

Third, process control networks often require third-party access to equipment during emergencies and for regular maintenance by integrators and vendors. These external parties should be authenticated, and access should be controlled via a key management framework. While this is an ancillary requirement, it is an added benefit of the ADTKM system.

Finally, a common challenge within EDSs is the lack of ability for field devices to support necessary levels of cryptographic security. EDS equipment is often designed to last decades, and maintaining up-to-date security postures can be difficult. Two areas in particular lack the ability

to update for future security: having the resource capacity to perform advanced cryptography and having the necessary amount of high-entropy data available for performing the number of cryptographic actions. One of the most resource-consuming tasks that is fraught with risk is the generation of high-entropy key materials. Offloading key generation removes this risk by utilising an updatable platform design to generate large quantities of entropy data.

For this paper, a Dolev-Yao communication threat model (Dolev & Yao 1983) was utilised to drive the design of this key management system where the adversary can overhear, intercept, and synthesise any message and is only limited by the constraints of the cryptographic methods used. As a restriction to this model, any traffic, synthesised or not, is mirrored to a self-monitoring capability. Physical access to devices is provided to the adversary, under the condition that the defender is aware of such a compromise. In turn, the defender can put the device on a blacklist. The key material on the compromised devices, however, stays with the compromised device. A detailed list of scenarios is provided in **Appendix 2** of this paper.

ADTKM Architecture

The ADTKM system architecture is designed to accommodate the challenges and unique characteristics of process control environments. Process control networks place more emphasis on availability and reliability than do other more generic IT networks. Therefore, the ADTKM system has been designed with the assumption that the communication infrastructure between non-physically connected sites is unreliable. In addition to providing the functionality prescribed for the system, the ADTKM architecture is designed to reduce impacts on availability and operations as much as possible.

Figure 1, below, depicts the ADTKM system high-level architecture. The diagram shows the interaction between facilities within a utility's process control network as well as with a third-party entity that must interface with the process control equipment (an integrator or vendor). The various high-level communication interfaces are captured to showcase how the architecture fits together and integrates into current process control system networks. The architecture depicted leans heavily towards SCADA-type infrastructure, but the ADTKM system architecture is designed such that it accommodates other process control networks.

The control centre facility houses the main functional components of the ADTKM system. The control centre in a process control network is architecturally designed to control assets that are physically dispersed, either geographically or across disparate networks. The ADTKM system mimics this characteristic and was designed to centralise the trust management functionality of remote cryptographic assets. Therefore, the majority of the ADTKM system components are housed within the control centre.

The remote station represents a dispersed collection of assets that are physically separated from but monitored and controlled by the control centre. As previously mentioned, the design of the ADTKM system was created under the assumption that communication infrastructure between remote stations is unreliable. **Figure 1,** below, depicts the remote station to assist in describing how the interface operates in the face of unreliable communication infrastructure and a high-availability requirement.

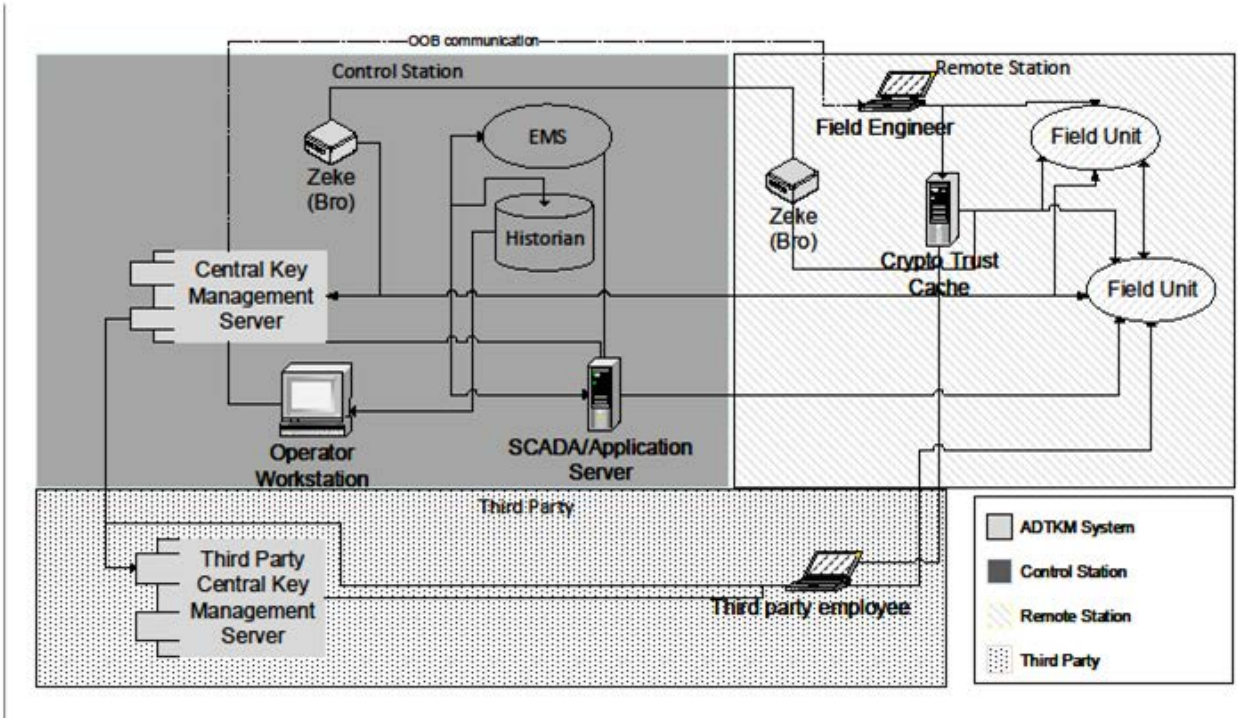

Figure 1: ADTKM logical architecture diagram

In **Figure 1**, all items shaded light grey are components of the ADTKM system. The remaining items are included as representative applications and entities within the process control environment. These applications and entities show the integration of the ADTKM system into the process control network. The interfaces and functionality that the ADTKM system provides to the applications are described using these representative applications.

Central Key Management server

The Central Key Management server (CKM) is the centralised control mechanism for the ADTKM system. The CKM provides the location and interface for users to define policies and configuration settings for the rest of the system. The CKM component is responsible for maintaining the process control system device identity enrolment data necessary to perform key management functionality. All cryptographic policies are maintained by this component. The information maintained by the CKM is the basis for the audit and forensic reporting capability of the ADTKM system.

The CKM also provides centralised authentication and authorisation services. All devices requesting access to cryptographic material or entities requesting access to applications or resources must first be authenticated to the CKM. Authorisation roles are defined and stored within the CKM component. These roles are utilised by the CKM to provide authorisation information to the Cryptographic Remote Trust Cache (CRTC) and end devices for allowing access to applications and resources. All authorisation and authentication actions are logged by the authentication, authorisation, and accounting service to support auditing and forensic activities.

The CKM provides the interface with peer CKM services to enable cross realm/domain authentication and authorisation. The CKM is designed to accommodate the requirements of third parties needing access to process control equipment for configuration or maintenance. To reduce the operational burden of managing third-party entities, the CKM component is designed to support roles that extend to other organisations.

Cryptographic Remote Trust Cache

The CRTC provides remote, distributed operation capability while still enabling centralised control. The CRTC leverages a ticket-based authentication and authorisation capability to enable the remote stations' cryptographic services to continue functioning for a time in the event of failed communication with the control centre. When a device requests a new key or an entity attempts to log into a remote station device, the device must first authenticate with the CKM. The CKM then provides a ticket with a configurable lifetime that enables the CRTC to authenticate and authorise the device remotely. In the event of communication failure, the remote station can continue operation because the needed Kerberos Ticket Granting Ticket (TGT) is stored locally.

The CRTC provides a reliable, cryptographically-entropic, random source to generate cryptographic material. It offers the ability to generate all of the commonly accepted and used cryptographic material such as different forms of symmetric and asymmetric keys and certificates. All cryptographic material in the system is generated by the CRTC component. The CRTC generates required key material for devices integrated into the ADTKM system.

Field unit

The field unit is the user of the ADTKM system. Devices that must securely communicate will implement the ADTKM client libraries in order to communicate with and collect key material from the other ADTKM components. The field units could include any type of embedded field controller or sensor such as Remote Terminal Units (RTUs), programmable logic controllers, Intelligent Electronic Devices (IEDs), and the control room software services that communicate with them such as Object linking and embedding for Process Control (OPC) servers, SCADA servers, and communication processors.

The general Kerberos standard defined in RFC 4120 is for user authentication with a password. However, the ADTKM system is targeted for machine-to-machine use, which does not utilise user accounts and passwords. Therefore, the ADTKM system leverages the Kerberos protocol extension by RFC 4556, Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) (Zhu & Tung 2006) to provide an authentication method using public key certificates for identity and authentication. The ADTKM system utilises PKINIT, along with a combination of Trusted Platform Module (TPM) (Trusted Computing Group 2016) and Institute of Electrical and Electronics Engineers (IEEE) 802.1AR (IEEE 802.1 Working Group 2018) for device identity.

Zeek (Bro) monitor

Security monitoring has long been seen as an essential analogue to enforcement (Anderson 1980). This is because enforcement is typically incomplete in order to make security computationally tractable (Schneider 2000) and/or usable, and because well-defined monitoring can cover unfore-

seen situations that enforcement might not know to cover. This is true for a variety of reasons (even for security protocols such as Kerberos which have been formally verified), not the least of which is because there can be gaps between protocol specification and implementation. The ADTKM system is no exception. Although ADTKM is itself a well-defined protocol based on well-defined components, including Kerberos, the system can still be attacked and/or fail in unexpected ways. Further, even in case of proper operation, it is desirable to simply have an independent record of events to provide justification that the system is operating correctly. However, by defining known good states and alerting on deviations from them, akin to specification-based intrusion detection (Ko, Ruschitzka & Levitt 1997), at least those attacks can be detected if not defended against in real-time (Peisert *et al*. 2007). To monitor ADTKM operation, the open-source Zeek (née Bro) Network Monitoring System (Paxson 1999) is used to capture and report this information. Zeek is designed with the understanding of the communication protocols used in substations and of the behaviour of the rest of the ADTKM system with which it can determine anomalous behaviour of the protocol. These deviations often represent a system failure or threat action. The authors' use of Zeek is distinctive from the way that Zeek and other intrusion detection systems are typically used in that this study performs specification-based intrusion detection (that is, alerting on actions that differ from a "known good" set of events) rather than the more traditional misuse-based intrusion detection (that is, alerting on actions that match a 'known bad' set of events). Thus, in the case described in this paper, Zeek plays an integral role in assuring that the protocol is operating correctly, rather than as a general security monitor.

Foundational Kerberos Feature Operation For ADTKM

The Kerberos protocol (Neuman *et al*. 2005) provides a perfect standards-based foundation to address the described challenges. Kerberos fulfils the requirements for providing short-term distributed operation in times of lost communication while still enabling centralised management of authentication and authorisation. Also, Kerberos has capabilities to support enabling third-party communication with devices when needed. Some infrequently used capabilities, described in the following sections, of the protocol are leveraged to support the desired functionality of the ADTKM system.

Cross-realm trust for separating authentication and authorisation and third-party access

The Kerberos protocol was designed to support cross-organisational and cross-domain authentication. Realms are defined by Kerberos as the authentication control boundary for an identity. Kerberos typically leverages a direct trust model. In such a case, a principle or inter-realm key is shared that enables the foreign realm to authenticate its users and generate TGTs for the other realm (**Figure 2**, below) This enables entities in one security domain to use the services in another security domain, or in the case of EDS integrators' or vendors' access to the devices they are contracted to maintain.

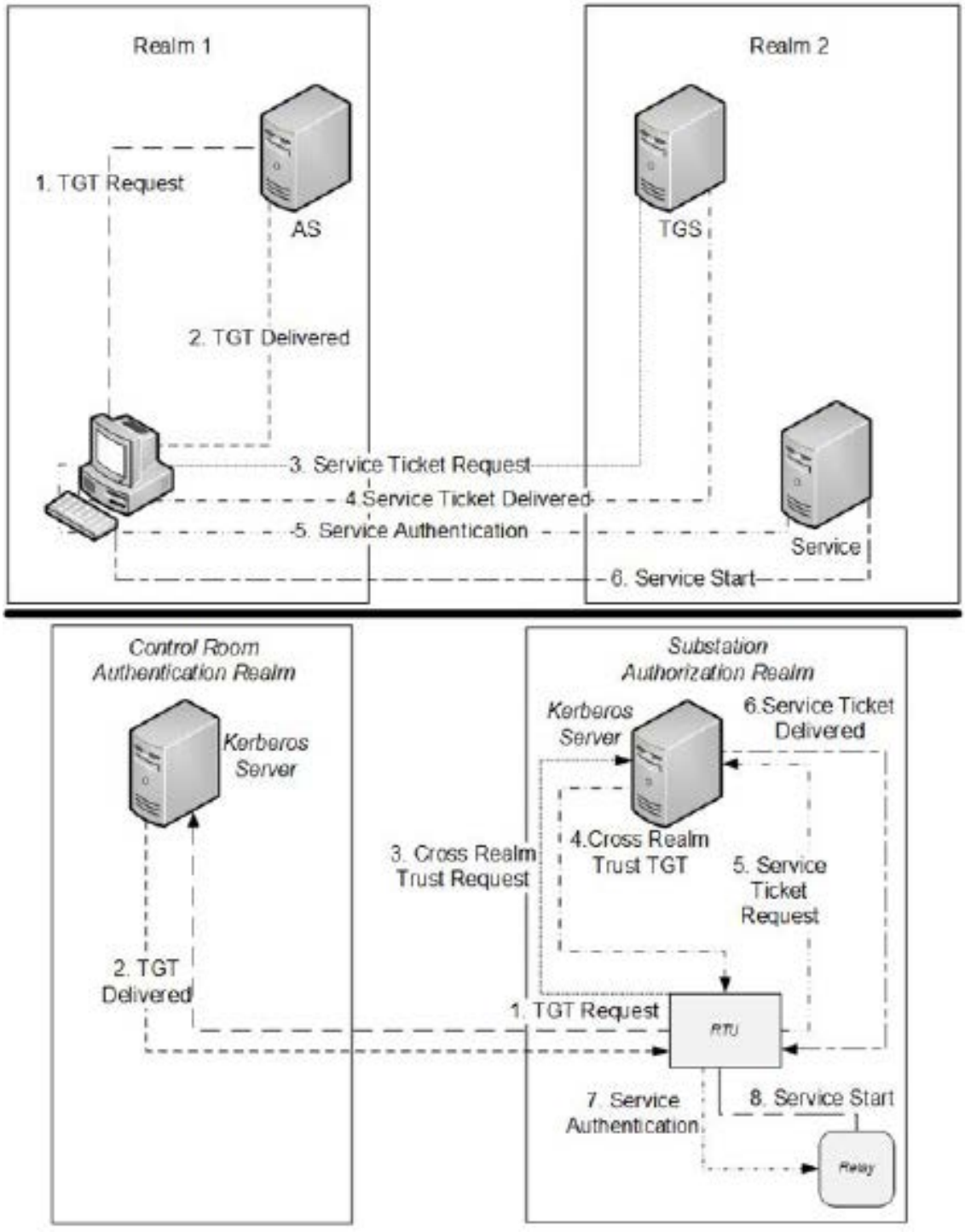

Figure 2: Direct trust, cross-domain Kerberos authentication (left) as compared to ADTKM Kerberos service architecture (right)

Additionally, cross-realm trust can enable disconnected operation. A normal Kerberos system is configured as in **Figure 3**, below. To enable distributed authorisation and key generation, cross-realm trust can be used for clients and services (again, see **Figure 2**, above) so that initial authentication is done at the control centre and the key distribution is handled in the field. Cross-realm trust allows domains of trust, generally entities of control such as different companies or different major units of companies, to enable accounts or users in one realm access to some set of services in another realm. The behaviour is enabled by sharing a trust between the two realms, allowing a TGT from one realm to authenticate to a service in another realm.

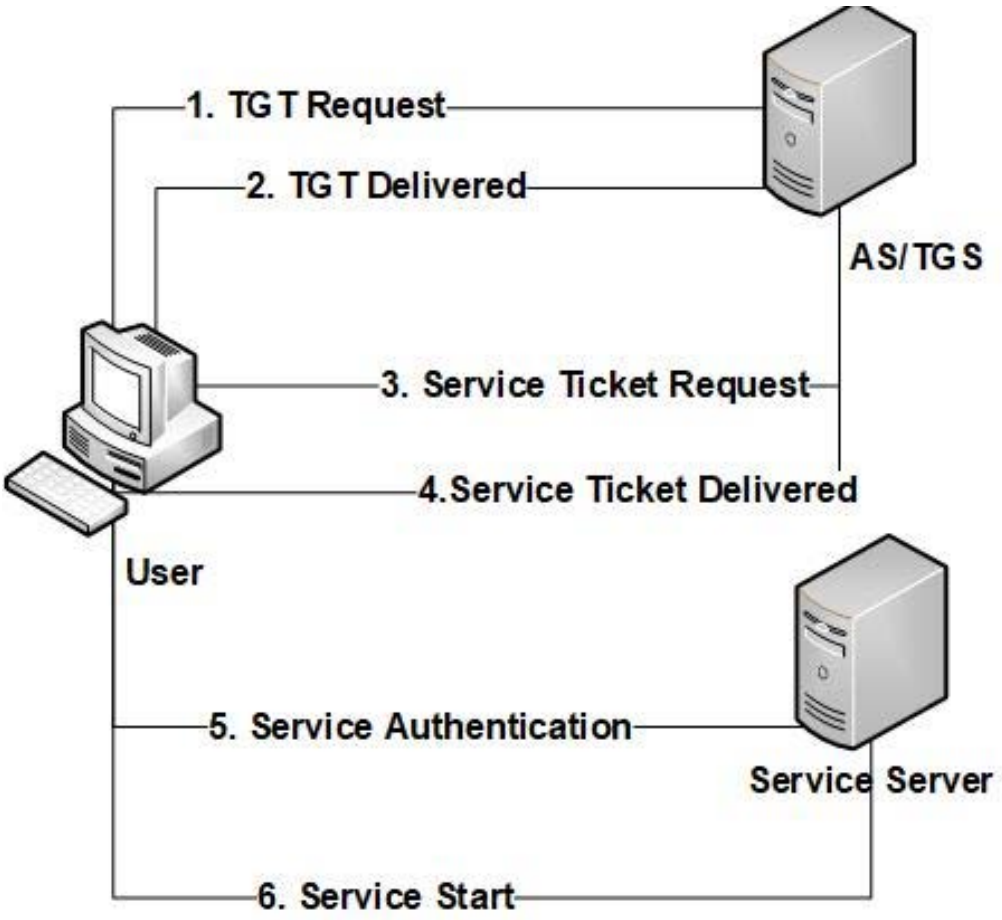

Figure 3: Traditional Kerberos architecture

Using cross-realm trust, it is possible to allow field environments to manage their own key distribution. Operating a single realm in the control room where all account principles exist provides central policy control and auditing of what services can be accessed by equipment and people in the field. Every field environment, or substation, operates its own additional realm where all service principles are configured. Through the separation of authentication in the control room and authorisation in the field, all devices, when trying to connect with a service, must first contact the control-room realm to authenticate and receive a TGT with a policy-defined lifetime. That TGT, through cross-realm trust, can then be used within the local substation realm to get a service ticket, which includes the necessary keys to establish secure applications and access secure services. If

communication with the control room is lost, the TGTs still function for their lifetime and enable the retrieval and use of additional key material to interact with services, thereby allowing secure applications and services to continue operation between field equipment. The separation of clients and services in this fashion enables the desired centralised control with temporary disconnected operation.

In **Figure 2** (above), the control room Kerberos service operates within the ADTKM CKM. Field units, such as the RTU and relay, must first authenticate with the CKM. With the TGT, they are then able to request access to a secure service (such as communication with peer IEDs for safety processes or higher-level controllers for automation control such as remedial action schemes) through the local field Kerberos service running within the CRTC. The service ticket from the CRTC includes key material that can be used for the requested secure service or application, in this case, the relay securely communicating with the RTU.

Authorisation extensions

The general application of Kerberos is as an authentication process for single sign-on. In this case, a user is authenticated by the Kerberos process, and then authorisation decisions are made by the end device based on the authenticated identity. However, Kerberos also provides the ability to extend and embed authorisation information to limit the applicability of service tickets (Microsoft Corporation 2018). One of the major users of Kerberos is Microsoft's Active Directory service for domain control, which embeds domain authorisation information into tickets to control service access. The ADTKM system similarly leverages this ability to provide authorisation to device service communications. Kerberos uses Service Principle Names (SPNs) to authenticate to the appropriate service. In the Active Directory use case, the SPN is generally a combination of the service name, server domain name, and the application instance. For control system environments, these concepts are not suitable. Some process control and SCADA protocols are object-oriented in design, where functions and objects are well-defined. These protocols lend themselves to attribute authorisation. For instance, the IEC 61850 (IEC TC57 2019) standard provides descriptions of services and devices to support defining a Kerberos SPN for these environments. IEC 61850 defines logical device names and logical node names that represent services a device can provide. By generating an SPN from these two pieces of information, service access policies can be defined. While IEC 61850 provides a use case with a strong capability to support the Kerberos protocol, there are other systems and applications that do not provide the same ease of mapping. For these use cases, a mapping for SPN is necessary.

Experimental Evaluation

All security additions to a system have the potential to impact performance. Each use case has different performance requirements. Performance impacts could be significant for some applications and inconsequential for others. It is important to understand the performance requirements of security solutions before using them for an application. As such, some experimental tests were executed to bound the performance impacts expected from the ADTKM system under various conditions.

The overall architecture used to evaluate the ADTKM was designed around a simple model of a SCADA system. A SCADA server in a control room is connected to an RTU substation. The RTU is connected to a field device (relay). The IEC 61850 Manufacturing Message Specification

(MMS) was the protocol used for SCADA communication. An Opal-RT real-time digital simulator was configured with an IEEE 39 bus physics model for driving the system inputs to the relays. The test plan included three phases. Each test phase focused on a different test configuration and system under test: a baseline system configuration, a system with the ADTKM solution integrated, and a system with an IEC 62351 test setup. All three test system configurations are presented in **Figure 4**, below.

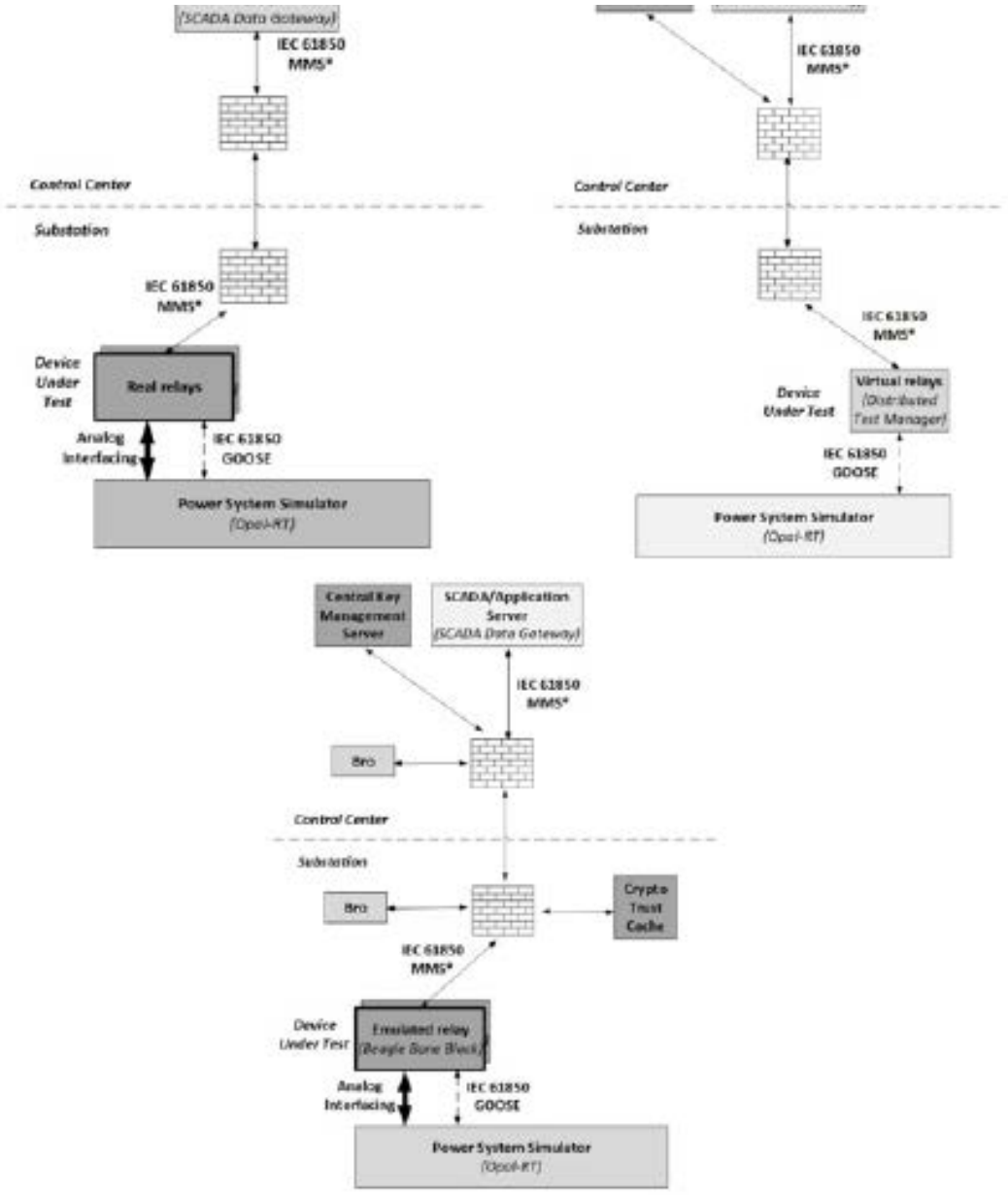

Figure 4: Baseline, IEC 62351, and ADTKM test system configuration

Real equipment was used in the baseline case to quantify the behaviour of the test setup. The goal of this test phase was to validate the model and configurations for realism and provide a baseline of performance to quantify the delta introduced by any key management and security actions. The use

of real equipment in the first phase helped calibrate the configuration and behaviour of simulated and proof-of-concept devices necessary in the second and third phases. In the baseline, the Triangle Microworks SCADA Data Gateway software and the Schweitzer Engineering Laboratories SEL 451 and 351 relays were leveraged.

Phase 2 focused on testing the performance and operation of a prototype implementation of the ADTKM. Since there are no commercially available operational devices that support the concepts in the ADTKM, it was necessary to use proof of concept relay software for testing device-to-device authentication and key management as described in the 'Field device prototypes' section, below. The rest of the SCADA setup is consistent with the baseline. The validated IEC 61850 configuration files from the baseline tests were used to configure the proof of concept relays to ensure consistency and limit the number of introduced artefacts. In addition to the SCADA test system, the additional ADTKM components were added to the communication network. This includes the authentication and authorisation services (Key Management Server and Crypto Trust Cache) as well as the system consisting of Zeek sensors monitoring the different networks.

The final setup for phase 3 was focused on evaluating the behaviour of an IEC 62351 key management process. Similar to testing the ADTKM solution, there is limited support for IEC 62351 in commercial relays; therefore, software relays, provided through the Distributed Test Manager from Triangle MicroWorks, were necessary to evaluate this test setup. The configurations of the relays were again transferred and used within these software relays to ensure consistency for comparative analysis. Finally, the PKI services necessary to operate the IEC 62351 protocols were included in the network.

Prototype Implementations

To evaluate the performance of the ADTKM concepts, it was necessary to develop a prototype implementation. Prototype code and hardware was developed to enable testing and demonstration of the ADTKM approach. The following sections provide an overview of implementations of the different architecture components.

Cryptographic Remote Trust Cache prototype

The CRTC was deployed on an SEL-3360 device running Ubuntu 16.04. It leverages its ticket-based authentication and authorisation capability through use of Heimdal (version 7.5.0), an implementation of Kerberos 5. The CRTC generates and registers all cryptographic material with the CKM for auditing and tracking purposes. The tools used to generate required key material for devices integrated into the ADTKM system include heimdal-clients 1.6, hxtools, and ktutil.

CKM prototype

The Central Key Management server is built on a virtual Ubuntu 16.04 system and utilises Samba 4.0 as the Active Directory Domain Controller. Heimdal is used by Samba for the underlying Kerberos implementation. All user accounts, groups, and authentication/authorisation policies are handled by the Samba utility samba-tool. For testing third-party trust scenarios, two or more instances of CKM virtual machines are executed, where each CKM belongs to different domains of control (or different companies). Different realms and domains are established within each CKM

to enable cross-realm authentication and authorisation. Kerberos keytabs are exported and shared to enable the cross-realm trust along with additional configuration through samba-tool.

Field device prototypes

In order to enable field devices to communicate securely, it is necessary to have them enabled with client tools to operate within and test the ADTKM approach. The prototype field devices were built on BeagleBone Black boards running Linux Debian 8 Jessie. The IEC 61850 software library from SystemCORP was used to provide the IEC 61850 standard operations. Client Kerberos utilities were necessary for field devices to get the key material needed to establish secure communication. Again, the Heimdal project is leveraged to provide the client software for interacting with the CKM and CRTC services. The PKINIT pre-authentication mechanism for Kerberos is used alongside X.509 certificates to provide device authentication to the CKM. A TPM is required to bind an identity of the prototype devices. A SparkFun CryptoCape was utilised for TPM services to bind an identity to each proof-of-concept relay.

Zeek Network Traffic Analyser

Zeek runs on computers with non-intrusive Ethernet tap access to communications between the key distribution server and the field devices within the remote station. It monitors the packets that communicate to and from control devices containing keys, as well as between local and central ticket granting servers, and reports appropriately on normal operation and error conditions. A list of known compromised, lost, or stolen field devices is made known to Zeek so it can properly identify revoked keys and field devices being misused. Multiple Zeek devices are strategically placed to monitor the communication between equipment and ADTKM components. If an anomalous event is detected, an alert is generated and logged.

Results

To contextualise results of the testing, both baseline 61850 operation and secured IEC 62351 system tests were performed for comparative analysis. The analysis of performance and behaviour are documented. **Table 1**, below, shows the high-level status of the tests run (as documented in Appendix 3). Following the success and performance result discussions for each phase, the comparison of results is reported, providing information on how differently the systems performed under each case. The dash mark under IEC 62351 represents that the test was not possible because of the way the standard was implemented by the application used in testing.

Test Description	Baseline	ADTKM	IEC 62351
Connection	✓	✓	✓
Device input/output (I/O)	✓	✓	✓
Loss of communication	✓	✓	×
Spoofing	×	✓	✓
Scanning	✓	✓	
Replay Attack	×	✓	✓
Key update		✓	✓
Exposed key		✓	✓
Failure/loss of key management server		✓	–
Security audit		✓	
Unauthorised access attempts		×	
Non-compliant device		✓	
Attack against self-monitoring systems		✓	

Table 1: Test results summary

Normal communication

Very similar performance was expected between the approaches for normal communication; however, the IEC 62351 performance was significantly slower (**Table 2**, below). While uncertain, it is not believed that IEC 62351 is the variable that causes the significant time difference. Limits in exactly reproduced test cases probably contributed to unexpected results. The differences in security mechanisms, the underlying IEC 61850 implementations, and underlying devices all probably had impact on these numbers.

ADTKM	IEC 62351
0.054 secs	0.2 secs

Table 2: Average round-trip time for SCADA communication

Three variables could be confounding the results: the encryption protocol, the version of IEC 61850, and the execution platforms. The ADTKM solution uses Kerberos security encryption mechanisms, which is slightly different from the Transport Layer Security used by the IEC 62351 protocol. The ADTKM approach is focused on getting key material to the end units to establish whatever security protocol they desire; it simply uses the Kerberos security mechanisms as an easy method to prototype. In addition, the IEC 61850 implementations between the ADTKM and IEC 62351 tests were different. The ADTKM solution utilised the SystemCORP library while the IEC 62351 test utilised the Triangle MicroWorks library. Differences in how the libraries implement the standard and perform functions could affect the timing. Finally, the platforms running these libraries were different. The ADTKM ran on embedded BeagleBone Black systems running Linux and few additional services. The IEC 62351 ran on a Windows 7 laptop. The platform variations could also affect results.

Ultimately, the key management approach should have little influence on the normal secure communication performance. The protocols, mechanisms, and software/hardware implementations all

are expected to have much more influence on performance. As expected, both tests added latency as compared to the baseline communication times. The additional processing for the cryptography and the additions of security data to packets will make the time to communicate slower but not significantly so for most use cases. Consequently, the test results were inconclusive.

With the addition of a custom handler to extract the state machine of ADTKM, Zeek handled all the outlined threats completely, with two exceptions. The first exception is legacy devices, which requires that the secondary connection already used for the legacy device also be used for communication between Zeek at both the central location and the field level. Thus, Zeek handles this partially. The other exception is addressing communication between two already compromised field devices, as encryption between the central key authority and the field devices prevents Zeek from checking whether issued tickets are valid.

Session establishment performance

As was expected, it was found that the ADTKM had a slower performance in the time to establish a new session (**Table 3**, below). There are several factors that influence this result. The first and most significant is that automated mechanisms to perform the authentication to the Kerberos authentication service were not developed. This step was manually performed. After completion, the ADTKM IEC 61850 emulated device applications were then started. Manual execution caused the large amount of time for session establishment of the ADTKM prototype. If automated, it is expected the time would be a higher sub-second. The ADTKM approach requires multiple session negotiations before the service ticket is finally delivered to the end device. First, the device must authenticate to the control room domain. With the control room domain TGT, the device can then authenticate to the appropriate substation domain of the device it wants to communicate. With the second TGT the device can request the service ticket with the communication keys. Multiple back and forth communications make initial session establishment slower than the IEC 62351 protocol.

ADTKM	IEC 62351
7.54 secs	0.11 secs

Table 3: Average time to establish first session

The IEC 62351 session establishment, on the other hand, is faster than would be expected in the default IEC 62351 behaviour. The Device Type Manager application only supports statically defined Certificate Revocation Lists (CRLs), which are configured when starting the application. As configured, session establishment requires only the time to negotiate a session between the two devices; no third-party communication is required. For installations that utilised CRL mechanisms, the session establishment performance will be the best. However, this approach has its detractions, discussed in the next section, and it is not the default option proposed by IEC 62351. The default mechanism recommended is to use an Online Certificate Status Protocol (OCSP) server, which provides real-time authentication of certificates per session establishment. The OCSP process only requires communication with one other third party instead of the three additional communications for the ADTKM process. As such, the OCSP approach for new session establishment, while not tested, is expected to be faster than the ADTKM approach.

Session renewal

New session establishment for the ADTKM approach is a less frequent event, designed to minimise the number of times it is required to communicate with the control room. As such, the general time for session establishment will be the time to renew a session. The session renewal process only requires the session negotiation between the local substation domain and the session establishment between the devices, thereby reducing the time to a session. Since this eliminated the manual step in the initial session establishment, the time difference is significant in the test results (**Table 4**).

ADTKM	IEC 62351
0.168 secs	0.11 secs

Table 4: Average time to re-establish session security

There is no difference in session establishment and re-establishment for IEC 62351, so the times are the same. Again, the CRL method is the fastest because it does not require communication with a third party. The OCSP method does require communication with a third party, so this performance is expected to be similar to the ADTKM session re-establishment performance.

Discussion

Many things, beyond performance measurements, were learned from the comparative testing of the different key management approaches and implementations. In this section, the qualitative results of the testing are presented.

Replay/spoofing/masquerading defence

The replay, spoofing, and masquerading defences of both the ADTKM and IEC 62351 are expected to be similar. Both approaches are designed to authenticate devices and prevent malicious entities from manipulating data and acting like legitimate devices. Both approaches prevented these attacks from occurring. Devising good tests for these types of attacks is difficult, and no attack implementation testing can cover every possibility; therefore, the tests show that these protocols are secure against simple attack attempts. However, since both approaches are built upon well established and accepted security mechanisms, their robustness to these attacks has been verified through previous research. The ADTKM project also developed secondary security monitoring techniques to detect when these types of attacks are attempted. While the prototype ADTKM did not fully succeed as expected in all the test cases, the ability to detect attack attempts helps prevent further attacks against the system. Similar capabilities could be developed to support IEC 62351 as well.

Authorisation

Deauthorisation is the process of revoking access and credentials from a device. The authorisation process is one of the defining differences of the ADTKM approach. A device's identity is authorised to a set of substation/field domains in the control room on session establishment. The authorisation persists by a configurable policy such as an hour, day, week, etc. When authorised, an entity can retrieve session keys and communicate. Kerberos extensions also provide the mechanisms to

add function-specific authorisations, such as accessing configuration management but not SCADA functions. While this research did not delve deeply into function-specific authorisation, it could add an additional granularity of authorisation that can be centrally defined and managed.

IEC 62351, on the other hand, provides limited authorisation features. The utilisation of certificates is for authentication purposes but does not specify what an authenticated entity is authorised to do. This forces end units to handle authorisation processes, which requires distributed management and control. Efforts to embed authorisation information into the certificates have and are being developed, but nothing has been standardised or established for this use case. CRLs have a long history of problems in managing them. Distributing CRLs in a timely and efficient manner is difficult. This challenge is likely the reason IEC 62351 lists the OCSP method as the default and recommended mechanism. The major limitation of the OCSP approach is a required communication channel to the central service which limits disconnected operation. OSCP stapling (Rescorla 2018) is a newer technique developed to allow a requesting service the ability to prefetch the OCSP authenticity response to provide the peer device on connection. This technique approximates some of the benefits of the ADTKM approach but does not address the authorisation challenges.

Key material generation

In the ADTKM Kerberos approach, the session key material is generated by the substation/field domain servers. This alleviates limitations of end devices in generating enough entropy material for keys. The IEC 62351 certificate-based approach forces the end devices to adequately generate key material for session use.

Disconnected operation

One of the key functional goals of the ADTKM approach is to support limited disconnected secure operations. The authentication and authorisation time period of this approach allows organisations to set policy on how long devices can continue to securely operate in a disconnected state. This allows them to tailor security controls to their risk posture and regulatory requirements. The IEC 62351 certificate-based approach has less flexibility in disconnected operation. The CRL method allows devices to operate in a disconnected state indefinitely. This may not be appealing if the disconnected state was induced to attack the system and the devices should be disconnected for security and/or safety. The OCSP method forces an always connected situation, where any loss in communication with the control room leaves devices in an uncertain state of how to behave when a device cannot be authenticated.

Zeek network monitoring

Since key management provides the foundation of a system's security, it is a high-interest attack target and should be monitored. Zeek's attack detection provides the necessary functionality to monitor the behaviour of key management communication and is compliant with the ADTKM threat model. This includes detection of traffic from compromised or stolen devices that were previously authenticated but whose keys were subsequently revoked, without the need to keep memory-expensive CRLs for an indefinite duration. Zeek monitors for correct use of the key exchange syntax and semantics as well as denial-of-service attempts and raises alarms if necessary. This also includes too many requests for a new key to the CKM itself. Since Zeek tracks process state mod-

els, information collected by Zeek can also be used to support troubleshooting and error management. Zeek allows clients to connect to its database to enable such queries. Zeek does a good job of tracking the real-time status of the nodes in a decentralised fashion (good for load balancing).

Conclusions and Future Work

This paper presented a novel key management design, using Kerberos, that solves some of the unique requirements of remote-control systems; a working implementation was also demonstrated. Kerberos is already a well-established protocol with a large install base. It can be leveraged to provide a secure foundation for security in remote process control environments using some of its more peripheral features. This enables leveraging of existing IT expertise for securing operational technology environments using a thoroughly tested and widely used framework.

The source code implementations for the base features (Seppala 2019) and the monitoring system (Gentz and Peisert 2019) have been released into open source. However, there is still a need for additional work in the future. While the ability to embed authorisation information mapped to protocol functions and objects was designed during this project, the current open-source implementation lacks this feature. Future effort is necessary to develop the taxonomy or language of authorisation SPNs for each SCADA/process control protocol. The client libraries would need to read and honour this authorisation information.

Finally, for this approach to progress requires developing more applications around its use. Security should be built into applications, and key management is at the base of this functionality. The current prototype was developed as a wrapper around an existing application, which has some negatives such as that odd network behaviour and potential application instability issues. Future efforts are necessary to apply the ADKTM approach to a variety of applications to test performance and behaviour.

Acknowledgements

This research was supported in part by the Director, Cyber Security, Energy Security, and Emergency Response, Cyber Security for Energy Delivery Systems program, of the U.S. Department of Energy, under contracts DE-AC05-76RL01830 and DE-AC02-05CH11231. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors of this work.

References

Anderson, JP 1980, *Computer security threat monitoring and surveillance*, Technical report, James P. Anderson Company, Fort Washington, PA, US.

Baumeister, T 2011, 'Adapting PKI for the Smart Grid', *Proceedings of the 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 249–54.

Beaver, CL, Gallup, D, Neumann, W & Torgerson, M 2002, *Key management for SCADA, Technical report SAND2001-3252*, Sandia National Laboratories, viewed 12 November 2019, <https://energy.sandia.gov/wp-content/gallery/uploads/013252.pdf>.

Choi, D, Kim, H, Won, D & Kim, S 2009, 'Advanced key-management architecture for secure SCADA communications', *IEEE Transactions on Power Delivery*, vol. 24, no. 3, pp. 1154–63.

Choi, D, Lee, S, Won, D, & Kim, S 2010, 'Efficient secure group communications for SCADA', *IEEE Transactions on Power Delivery*, vol. 25, no. 2, pp. 714–22.

Crain, A, 2017, 'Secure SCADA protocol for the 21st century (SSP-21)', *Proceedings of the 2017 SANS ICS Security Summit and Training*, Orlando, FL, US.

Critical Infrastructure Protection Committee 2009, *Critical Infrastructure Standards 002-014*, North American Electric Reliability Corporation (NERC), Washington, DC, US.

Dawson, R, Boyd, C, Dawson, E & Gonzalez-Nieto, JM 2006, 'SKMA: A key management architecture for SCADA Systems', *Proceedings of the 2006 Australasian Workshops on Grid Computing and e-Research*, vol. 54, ACSW Frontiers '06, Hobart, Tasmania, AU, pp. 183-192, ISBN 1-920-68236-8, <http://dl.acm.org/citation.cfm?id=1151828.1151850>.

Dolev, D & Yao, A 1983, 'On the security of public key protocols', *IEEE Transactions on Information Theory*, vol. 29, no.2, pp. 198–208, ISSN: 0018-9448, doi:10.1109/TIT.1983.1056650.

Ebrahimi, A, Koropi, F & Naji, H 2014, 'Increasing the security of SCADA systems using key management and hyper elliptic curve cryptography', *Proceedings of the 9th Symposium on Advances in Science & Technology* (9th SASTech 2014), pp. 17–24.

Gentz, R & Peisert, S 2019, 'LBNL Disruption Tolerant Key Management Monitoring for Stream-Processing Architecture for Real-time Cyber-physical Security (DTKM-SPARCS)' Github repository, <https://github.com/lbnl-cybersecurity/dtkm-sparcs>.

IEEE 802.1 Working Group 2018, 'IEEE Standard for local and metropolitan area networks: Secure device identity', *IEEE Standard 802.1AR-2018*, pp. 1–73.

International Electrotechnical Commission (IEC) Standard 2017, 'Power systems management and associated information exchange–Data and communications security, Part 9: Cyber security key management for power system equipment', IEC 62351-9:2017.

——2019, TC57 - Power systems management and associated information exchange 2019, 'Communication networks and systems for power utility automation – ALL PARTS', IEC 61850:2019 SER.

Jiang, R, Lu, R, Lai, C, Luo, J & Shen, X 2013, 'Robust group key management with revocation and collusion resistance for SCADA in smart grid', *Proceedings of GLOBECOM - IEEE Global Telecommunications Conference*, pp. 802–7.

Ko, C, Ruschitzka, M & Levitt, K 1997, 'Execution monitoring of security-critical programs in distributed systems: A specification-based approach', *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, US, pp. 175–87.

Manz, D, Edgar, T & Fink, G 2010, 'A hybrid authentication and authorization process for control system networks', *2010 Sixth International Conference on Information Assurance and Security*, Atlanta, GA, US, pp. 36–9.

Microsoft Corporation 2018, *MS-PAC: Privilege attribute certificate data structure*, viewed 12 November 2019, <https://msdn.microsoft.com/en-us/library/cc237917.aspx>.

Mittra, S 1997, 'Iolus: A framework for scalable secure multicasting', *SIGCOMM '97 Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer communication*, pp. 277-88.

National Institute of Standards and Technology (NIST) 2010, *NISTIR 7628 Guidelines for smart grid cyber security v1.0*, September, viewed on 12 November 2019, <https://www.nist.gov/sites/default/files/documents/smartgrid/nistir-7628_total.pdf>.

Neuman, C, Yu, T, Hartman & S, Raeburn, K 2005, 'The Kerberos network authentication service', *Network Working Group Request for Comment (RFC) 4120*, vol. 5, viewed on 12 November 2019, <https://rfc-editor.org/rfc/rfc4120.txt>.

Paxson, V 1999, 'Bro: A system for detecting network intruders in real-time', *Computer Networks*, vol. 31, no. 23, pp. 2435–63.

Peisert, S, Bishop, M, Karin, S & Marzullo, K 2007, 'Toward models for forensic analysis', *Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, Seattle, WA, US, pp. 3–15.

Piètre-Cambacédès, L & Sitbon P 2008, 'Cryptographic key management for SCADA systems-issues and perspectives', *2008 International Conference on Information Security and Assurance (ISA 2008)*, pp. 156–61.

Rescorla, E 2018, 'The transport layer security (TLS) protocol version 1.3'. *Network Working Group Request for Comments (RFC) 8446*, viewed on 12 November 2019, <https://rfc-editor.org/rfc/rfc4120.txt>.

Rezai, A, Keshavarzi, P & Moravej, Z 2017, 'Key management issue in SCADA networks: A review', *Engineering Science and Technology: An International Journal*, vol. 20, no. 1, viewed 12 November 2019, <http://www.sciencedirect.com/science/article/pii/S2215098616303482>.

Schneider, F 2000, 'Enforceable security policies', *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 1, pp. 30–50.

Seppala, G 2019, 'ADTKM Disruption Tolerant Key Management', Github Repository, <https://github.com/pnnl/ADTKM>.

Sun, J, Yan, J & Zhang, K 2016, 'Blockchain-based sharing services: What blockchain technology can contribute to smart cities', *Financial Innovation*, vol. 2, no. 1, p. 26.

Tawde, R, Nivangune, A & Sankhe, M 2015, 'Cyber security in smart grid SCADA automation systems', *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, IN, pp. 1–5.

Trusted Computing Group 2016, *Trusted platform module library: Part 2: Structures,* Family 2.0, Revision 01.38, 29 September, Trusted Computing Group.

Wang, W & Lu, Z 2013, 'Cyber security in the smart grid: survey and challenges', *Computer Networks*, vol. 57, no. 5, pp. 1344–71.

Zhu, L & Tung, B 2006, 'Public key cryptography for initial authentication in Kerberos (PKINIT)', *Network Working Group Request for Comment (RFC) 4556*, pp. 1–42.

Zimmerman, P 1994, *The official PGP user's guide*, DIANE Publishing Company, Collingdale, PA, US.

Appendix 1: Lessons Learned

Through the implementation of a prototype system and execution of experimental tests, some distinct and critical things were discovered to enable a successful ADTKM system.

PKINIT library support

With Kerberos, the choice is between two popular implementations—Massachusetts Institute of Technology (MIT) and Heimdal—both offering unique strengths and weaknesses. MIT Kerberos is more widely used and much older, hence its enterprise stability and strong development support. It has better documentation available and debugging capabilities. Issues that users come across are often discussed online, and one is more likely to find answers to problems when using the more popular tool, in general.

However, a Linux-based field device with Samba works as an Active Directory Domain Controller. Samba requires Heimdal to use the PKINIT feature. Unfortunately, Heimdal has weak debugging capabilities that make it difficult to troubleshoot the many issues encountered.

At the end of the day, both Heimdal and MIT Kerberos offer the same basic functionality; they are just handled in different ways. The packages that need to be installed and the way configuration files need to be set up are different. Getting the configuration files correct was one of the trickiest tasks in this project, as there was a lot of contradictory information online.

Critical dependency on DNS

The Domain Name Server (DNS) is critical to the operation of the Kerberos protocol. Similar to a phone book, a DNS provides a directory of domain names and translates them to Internet Protocol (IP) addresses. Without the DNS in place, none of the devices would know how to communicate with another, which makes it a crucial component of the project. Kerberos' functioning properly is highly dependent on devices being able to find each other and communicate fully.

Originally, having the DNS on the Active Directory machine was tried. This proved to be troublesome if the connection to the main Active Directory machine was ever severed. Since the objective is for devices to be able to communicate and authenticate within the field, an alternative spot for a secondary DNS was necessary. It was decided a standalone local machine would provide continued operation even when the main Active Directory machine was disconnected. This allowed all the devices in play to communicate with each other throughout the tests, even when other important machines were disabled or turned off.

Abnormal network behaviour from wrapped system call security

The IEC 61850 library utilised in the prototype implementations of the ADTKM end devices provided a black-box communication process. All of the session establishment and socket control is handled within the library code and not exposed to the user. In order to add additional security to the IEC 61850 protocol, it was necessary to wrap system calls (such as send and receive) with Simple Authentication and Security Layer (SASL). The result was to force Kerberos authentication to succeed before allowing the client and server devices to communicate.

This resulted in the masking of information passed between the applications on the two communicating devices. Due to the SASL wrapper, the communication was encrypted, and the data were unrecognisable when investigating .pcap files with analysis tools such as Wireshark. This helped confirm some of the test cases that involved using Zeek for analysis, while not interrupting Zeek or any other logging from accessing the information they needed.

The wrapped system calls can affect the ordering of steps in which the secure link is established, which in turn can cause differences in the ordering of network traffic, which then affects Zeek analytics. An example is a non-trusted device that tries to establish a connection to another device and to keep it idle, and then acquires the key material needed to exchange any data. This requires the Zeek monitoring system to listen for a 'first data exchanged' event and not a 'connection established' event.

ICD file non-interoperability

One of the difficulties encountered during development of the prototype virtual relay used for implementing ADTKM was creation of a custom IEC 61850 IED Configuration Description (ICD) file that combined tags from several individual ICD files from real-world IEDs such as those from SEL. The process to combine tags from individual devices was very tedious and involved manual labour in creating a merged IED file to support a 61850 enabled RTU.

As there are no commercial devices that currently support IEC 61850 with support for IEC 62351, a prototype virtual relay that leveraged an IEC 61850 software stack was used to validate the per-

formance of ADTKM; specifically, the IEC 61850 software stack provided by SystemCORP. The process for this software to map data points in the ICD file to the internal database is very cumbersome as it requires a mapping to be established for each tag that is used manually. This process has to be performed individually for all tags and currently does not provide any support for automation or run-time modification using an external file. This limitation impacted the number of tags created in the prototype virtual relay and the speed at which changes could be made to test virtual prototype relays with different ICD files as a part of testing and performance evaluation.

In addition to the manual changes in an ICD file, changing the datatype used requires extensive rewriting of the source code to update and read the tag. Additionally, the SystemCORP library has limitations on the number of servers and clients that can run on one device. Up to two clients were successfully deployed on one BeagleBone Black, but only a single server can be run. This limited the ability to perform some tests.

Test application interoperability

Over the course of the testing with Triangle MicroWorks' Distributed Test Manager, it was observed that the IEC 61850 software stack used did not work well to obtain the tag lists for the device in the Kepware KEPserverEX OPC server via self-description. This could be potentially attributed to IEC 61850 interoperability issues by comparing and testing the self-description features by connecting to a real-device that supports IEC 61850.

Unexpected errors were observed that crashed the software during the testing of the use cases for the IEC 62351 security comparisons. These errors occurred due to incompatibility in the X509 certificate versions used for the Transport Layer Security sessions.

Appendix 2: Mitigated Risk Scenarios

The following are six risk scenarios mitigated by the ADTKM system.

Scenario 1: Loss of Communication to Control Room

Emergency event

A fault condition is occurring, and communication to the control centre is lost. Mitigations include

- TGT allows field authorisation for a user-configurable time;
- CRTC provides the ability to enable limited distributed authorisation;
- CRTC provides distributed logging and caching of what is happening to report back to the control room when connection is re-established for auditing and central control.

Third-party assistance

A situation occurs in which communication is lost to the control room and third-party field engineers need to help restore service more quickly. Mitigations include

- Authentication to provide TGT can be performed through secondary communication (such as cellular, satellite), which then enables access to local devices;

- Through centralised service, temporary access to additional third-party field engineers could be provided;
- See 'Scenario 2' for general third-party access case.
-

Scenario 2: Integrator/Third-Party Access

Often integrators, third-party vendors, or other utilities (in shared environments) need to access or communicate with equipment. Mitigations include

- Cross-realm authentication (part of Kerberos);
- Additional trust checks (these will be defined as part of this project);
- Zeek communication monitoring (detect probing deviation of process from insider threat).

Scenario 3: Spoofing, Man-in-the-Middle, Masquerading

An attacker has gotten onto the network (for example, compromised a computer, insertion into communication path). Mitigations include

- Zeek communication sensing;
- Cryptographic protections of authentication/authorisation communication;
- Identities bound to devices with TPM.

Scenario 4: Stolen Device

A device is stolen or compromised to use as a method to attack/compromise the rest of system. Mitigations include

- TPM used for identity credentials protects from reuse of identity;
- Zeek communication monitoring;
- See 'Scenario 5' for further mitigations.

Scenario 5: Exposed Key (Employee Fired or Quit)

For some reason, a key or set of keys is no longer secure. In some instances, there are regulatory guidelines on how quickly cryptographic material and access control must be updated. Mitigations include

- Central key information storage provides quick audit trail of what keys are being used and their provenance;
- Token-based key system provides a short life span for authorised use of keys—users can set this time;
- Centralised authentication/authorisation process prohibits disabled accounts from obtaining new keys.

Scenario 6: Security Audit

Some utilities must comply with federal and regional cybersecurity regulations and be able to show they are meeting regulations. Mitigation includes

- Central key information storage provides quick audit trail of what keys are being used and their provenance.

Appendix 3: Detailed Test Description and Results

Test Phase 1

IEC 61850/MMS to OPC server connection

This test case establishes

- Device can successfully connect to an OPC (Object Linking and Embedding for Process Control/OLE for Process Control/Open Platform Communications) server via the IEC61850/MMS protocol;
- Device data structures can be accessed on OPC server via supplied ICD file;
- Device data structures can be accessed on OPC server via device self-description in compliance with IEC61850/MMS spec.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850/MMS and connect.
- Import ICD file into OPC server and verify all data blocks have been imported.
- Delete prior connection.
- Configure OPC server for connection again and connect.
- Select self-description and verify all data blocks have been imported.
- Capture 10-minute pcap for baseline time characteristics (latency, jitter, round trip time).

Test Results:

- Pass.
- OPC server was able to successfully connect with the RTU running the IEC 61850 server using the ICD file.
- OPC server was not able to obtain tags via self-description from the RTU software. This was observed to be an issue with Triangle MicroWorks' Distributed Test Manager.
- Based on the 10-minute pcap, here are the baseline timing characteristics: round-trip time ~200 ms between the RTU and OPC server; ~1-4ms between the relay and RTU.

Device input/output

This test case establishes

- Device properly displays static measured values applied to the board I/O for all possible analogue values the device can measure;
- Device properly displays dynamic measured values from the board I/O for all analogue values that can be dynamic;
- Device properly shows digital inputs and operates digital outputs;
- Controllable outputs can be properly and stably controlled.

Test Procedures:

- After connection with OPC server is established, verify all analogue input data blocks are visible in OPC server.
- Verify all digital input data blocks are visible in OPC server.
- Verify all digital output data blocks are visible in OPC server.
- Apply full signal to digital inputs and verify OPC server shows inputs ON.
- Apply static half-scale values to analogue inputs and verify OPC server shows approximately correct half-scale value.
- Apply static full-scale values to analogue inputs and verify OPC server shows approximately full-scale value.
- Use OPC server to command digital outputs to close, and verify they do.
- Use OPC server to command digital outputs to open, and verify they do.
- Apply 10% scale values to analogue inputs, and verify OPC server shows approximately 1/10th scale value.
- Slowly increase analogue input values to 100% while monitoring the values reported by the OPC server, and verify the response approximately matches the physical increase of the signal.
- Repeat previous step with a slowly decreasing signal.

Test Results:

- Pass.
- All analogue and digital data blocks were visible in the OPC server.
- Values were tracking and responding to changes made in the simulator to drive the analogue inputs throughout the range of 10-100% scale values.
- Values of digital input and output blocks were also visible in the OPC server and matched the values on the RTU and the relay.

Loss of communication

This test case establishes

- IEC 61850 substation operation continues to operate properly in the event of communications failure with outside devices and/or networks;
- This test disconnects the SCADA server from the network to verify device continues to work in the substation.

Test Procedures:

- Disconnect the key server from the network the device is connected to.
- Verify that values are still being shown in OPC server.
- Change analogue input values, and verify changes show up in OPC server.
- Exercise digital inputs and outputs, and verify proper operation.

Test Results:

- Pass.
- The RTU was able to continue operation even when the communication link to the OPC server was lost.
- Changes in analogue and digital inputs were tracking appropriately.
- Digital outputs issued from the RTU were also seen updating on the relay appropriately.

Spoofing

This test case

- Establishes proper operation in the event of spoofed traffic to the device;
- Subjects the unsecured device to a spoofing/man-in-the-middle attack to establish baseline (unsecured) behaviour.

Test Procedures:

- Perform spoofing/man-in-the-middle attack on device, and record behaviour.
- Setup secondary system with KEPserver.
- Set secondary system IP as the same as first KEPserver.
- Connect, and attempt to collect data and control I/O.

Test Results:

- Partial fail.
- When the first device was connected, any attempts to establish a connection to the relay from a spoofed RTU device were reset by the relay at the TCP layer by sending a reset.
- If the first RTU device connection was disconnected, then the victim device would accept the connection from the KEPserver.

Scanning

This test case

- Establishes that the device does not react adversely to network scans and reports back properly;
- Subjects the device to active network scanning.

Test Procedure:

- Perform an active scan against the device IP address, and verify,
- IP address gets reported correctly by scanning tool, and
- Device remains responsive during and after scan.

Test Results:

- Pass.
- The relay's IP address was corrected reported by the scanning tool (Nmap).
- The relay remained responsive during and after the scan.

Replay attack

This test case

- Establishes device behaviour when subjected to a replay attack where a previously recorded protocol interaction is played back at the device to induce unwanted actions;
- Subjects the device to a replay attack of network traffic, which is done to establish baseline, unsecured device behaviour.

Test Procedures:

- Send control action to a digital I/O.
- Capture pcap of command.
- Craft replay packets of control command.
- Perform a replay attack against the relay.
- Capture pcap of the behaviour.

Test Results:

- Fail.
- Crafting packets and injecting them using the Scapy python library successfully performs control on a digital I/O.
- Success of this simplistic of an attack is predicated on the configuration of the field device.
- Allowed master/controller IP addresses can be set in some equipment; and as such, replaying a command from another IP would not work.
- More sophisticated attacks such as session hijacking could be performed that would achieve similar results.
- Intent of this test is just to show that the default protocols lack authentication mechanisms that are solved when deploying the security protocols in the second and third phases.

Test Phase 2
IEC 61850/MMS to OPC server connection

This test case establishes

- Device can successfully connect to an OPC (Object Linking and Embedding for Process Control/ OLE for Process Control) server via the IEC61850/MMS protocol with IEC 62351 security;
- Device data structures can be accessed on OPC server via supplied ICD file;

- Device data structures can be accessed on OPC server via device self-description in compliance with IEC61850/MMS spec.;
- How much performance impact, if any, will occur during normal operation due to ADT-KM;
- Zeek successfully detects the connection, extracts the correct certificates, and does not cause a false alarm.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850 MMS and connect.
- Import ICD file into OPC server, and verify all data blocks have been imported.
- Delete prior connection.
- Configure OPC server for connection again and connect.
- Select self-description, and verify all data blocks have been imported.
- Capture 10-minute pcap for analysis of time characteristics (latency, jitter, round trip time).

Test Results:

- Pass.
- Device connection to OPC established.
- Data structures able to be accessed on OPC server via device.
- Zeek logs detected and logged Kerberos activity.
- Based on the 10-minute pcap, round trip time: ~200 ms between the RTU and OPC server, ~200 ms between the relay and RTU with IEC 62351 security implemented.

Key update/new session

This test case establishes how much latency will be incurred due to the key management processes to authenticate and establish secure communication.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850 MMS, and connect.
- Import ICD file into OPC server, and verify all data blocks have been imported.
- Delete prior connection.
- Configure OPC server for connection again, and connect.
- Select self-description, and verify all data blocks have been imported.
- Force new session establishment.
- Capture pcap of the session establishment.

Test Results:

- Pass.
- Pcap files obtained during Kerberos authentication process.
- Log files (Zeek logs, pcap files) show details behind timing, which can be used to deduce how much latency was incurred.

Device input/output

This test case establishes that the ADTKM system does not interfere with control of Device I/O.

Test Procedures:

- After connection with OPC server is established, verify all analogue input data blocks are visible in OPC server.
- Verify all digital input data blocks are visible in OPC server.
- Verify all digital output data blocks are visible in OPC server.
- Use OPC server to command digital outputs to close, and verify they do.
- Use OPC server to command digital outputs to open, and verify they do.

Test Results:

- Pass.
- Logs showed that Device I/O was intact while ADTKM systems active.

Loss of communication to control room

This test case establishes

- Device and encryption/decryption continue to operate properly in the event of communications failure with outside devices and/or networks;
- If Zeek successfully detects communication during the loss of communication to the control room, extracts the certificate successfully, and correctly identifies if a TGT previously granted can still be used or is invalid due to expiration.

Test Procedures:

- Disconnect the connection from the control centre to the substation (Key Management Server and SCADA server).
- Send a control command from RTU to the relay to control I/O.
- Document behaviour.
- Force a session reestablishment between RTU and relay.
- Send a control command from RTU to the relay to control I/O.
- Document behaviour.
- Record Zeek logs to check if new key can be acquired from TGT without raising an error.

Test Results:

- Pass.
- Zeek logs and pcap files indicate device and encryption operated properly when communication with outside devices was lost.

Spoofing/man-in-the-middle masquerading

This test case establishes

- TPM chip on device is tied to device identity;
- If Zeek communication monitoring successfully detects attempts to manipulate the key management process, Zeek checks the DevID and addresses of each communication path and whether the corresponding tickets are issued and not expired/revoked;
- Expired/broken tokens will not grant access.

Test Procedures:

- Spoofing/masquerading
- Capture token delivered to key.
- Edit token/craft new token with changed/manipulated values.
- Capture logs from Zeek.
- Analyse to see if Zeek alerted on manipulation.
- Man-in-the-middle
- Set up an additional Kerberos with untrusted certificate that accepts every user and password and forwards it to the real Kerberos.
- Capture logs from Zeek.
- Analyse to see if certificate mismatch is found.

Test Results:

- Partial pass.
- Used a variety of methods to spoof/masquerade:
- Replayed an unedited pcap with tcpreplay. This was the most successful test in that Zeek detected partial Kerberos traffic–tgs request.
- Used WirEedit to change IP info, timing info. This test did not seem to generate any Zeek logs.
- Used a python module called Scapy to replay both unedited and modified pcap files. This test saw tgs requests in the logs when using unedited pcap files, but nothing when using modified pcap files.
- Open source C program KDCReplay – used to capture pcap files of specifically Kerberos authentication and replay them. This program seemed promising but did not behave as expected.
- No attempt resulted in a successful attack.
- Zeek logs and pcap files show partial Kerberos traffic–tgs request, but nothing more. No alerts or error messages.

Stolen device

This test case establishes

- TPM chip prevents re-use of device identity;
- Ephemeral credential life span is short enough to prevent long-term use of it;

- Zeek successfully detects that keys are no longer valid and raises an alert, specifically that Zeek has successfully added these keys to a revocation list and correctly maps the captured key to the list. This implies the keys are manually added to the revocation list.

Test Procedures:

- Remove device identity from Key Management Server.
- Add device to blacklist, and check if Zeek received this updated blacklist.
- Attempt to establish a session with the 'stolen' device.
- Record success/failure of session establishment.
- Check Zeek logs to see if successfully raised an alarm.

Test Results:

- Pass.
- Zeek files and pcap files show attempted secure session that was rejected.
- Zeek files show Kerberos error, but details of error didn't indicate a stolen device.

Exposed key

This test case establishes

- Zeek successfully detects that keys are no longer valid and raises an alert, specifically that Zeek has successfully added these keys to a revocation list and correctly maps the captured key to the list.

Test Procedures:

- Add device to blacklist and check if Zeek received this updated blacklist.
- Attempt to establish a session with the 'stolen' device.
- Check Zeek logs to see if successfully raised an alarm.

Test Results:

- Pass.
- Zeek files indicate the key was revoked and gave a warning; however, a secured connection was still established.

Security audit

This test case establishes whether Zeek can successfully print all active connections.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850 MMS, and connect.
- Import ICD file into OPC server, and verify all data blocks have been imported.
- Delete prior connection.

- Configure OPC server for connection again, and connect.
- Select self-description, and verify all data blocks have been imported.
- Print all active connection from Zeek, and confirm the list is accurate and complete.

Test Results:

- Pass.
- Zeek found all devices in use (active connections) and printed them. List was accurate and complete.

Scanning of ADTKM assets

This test case establishes that attempts to scan DTKM assets that meet certain predefined criteria and/or exceed a certain threshold (such as number of IP or ports addresses affected) are detected by Zeek monitoring system and an alarm is raised.

Test Procedures:

- Use Nmap to scan relays with aggressive settings.
- Check Zeek list to see if it reports scanning from IP address from/to relays.

Test Results:

- Pass.
- Zeek configured to scan for number of ports.
- A notice was flagged in the Zeek logs.

Failure/loss of key management server

This test case establishes

- Zeek can still operate even when CRTC is inoperable;
- Zeek alerts when the CRTC stops communicating or when key exchanges do not complete (for example, by looking for a lack of an ACK packet from the CRTC).

Test Procedures:

- Take CRTC down.
- Attempt to establish secure connections between relay and OPC server.
- Check if Zeek reports connection attempts as unsuccessful.

Test Results:

- Pass.
- Zeek was still functional with CRTC inoperable.
- Zeek alerted error messages when secured session did not complete.

Unauthorised access attempts to ADTKM assets

This test case establishes that the Zeek sensor detects when there are attempts to attack the ADTKM services.

Test Procedures:

- Use Nmap to scan relays with aggressive settings.
- Check Zeek list to see if it reports scanning from IP address from/to relays.

Test Results:

- Partial fail.
- Reports on scanning were minimal.
- No alerts or error messages were made.

Noncompliant Device

This test case

- Establishes Zeek validates all certificates sent over the network against the root certificate(s) and raises an alarm if validation fails;
- Determines if this is successful such that Zeek correctly identifies a device attempting to use expired or otherwise noncompliant keys that will cause an alarm from Zeek monitoring.

Test Procedures:

- Have a non-compliant device try to use the IEC61850 port.
- Zeek should report an error of untrusted device communicating.

Test Results:

- Pass.
- Zeek logs and pcap files indicate secure connection was attempted but failed.
- Zeek reported error of device being untrusted/invalid key.

Replay Attack

This test case establishes

- Packets with invalid or expired time stamps are rejected, that is, a replay attack will be ineffective;
- Zeek can raise an alarm if a replay attack occurs and the key has expired (Zeek cannot identify the payload of packets for keys that have not expired because they are encrypted, and the keys are not yet known to Zeek).

Test Procedures:

- Perform authentication/authorisation with field device to get service key.
- Capture token delivered to key.
- Replay token to service.
- Document success/failure of session establishment.
- Capture logs from Zeek.
- Analyse to see if Zeek alerted on replay.

Test Results:

- Partial pass.
- Zeek logs and pcap files included an attempted secured connection.
- Zeek files reported a TGS request that was processed, but no more information.
- No attempted replay attack was successful, as indicated by logs.

Attack against self-monitoring system

This test case establishes

- Zeek can detect (certain) kinds of attacks, specifically if the attacker is blocking the communication between the central Zeek instance and the Zeek instance at the field device level, then revocation list transfers, and heartbeats to Central Zeek are not acknowledged; and as a consequence, an alarm is raised at the central location;
- An adversary is not able to fake acknowledgements as it is not in possession of the correct encryption keys that Zeek is using.

Test Procedures:

- Block Zeek to Zeek communication (example, with a firewall).
- Check for alert from Zeek that communication is blocked.

Test Results:

- Pass.
- Connection severed between the two Zeek devices, errors and warnings were being printed.

Test Phase 3
IEC 61850/MMS to OPC server connection

This test case establishes

- Device can successfully connect to an OPC (Object Linking and Embedding for Process Control/ OLE for Process Control) server via the IEC61850/MMS protocol with IEC 62351 security;
- Device data structures can be accessed on OPC server via supplied ICD file;

- Device data structures can be accessed on OPC server via device self-description in compliance with IEC61850/MMS spec.;
- Zeek successfully detects the connection, extracts the correct certificates, and does not cause a false alarm.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850/MMS, and connect.
- Import ICD file into OPC server, and verify all data blocks have been imported.
- Delete prior connection.
- Configure OPC server for connection again, and connect.
- Select self-description, and verify all data blocks have been imported.
- Capture 10-minute pcap for analysis of time characteristics (latency, jitter, round trip time).

Test Results:

- Pass.
- The OPC server was able to successfully connect with the RTU running the IEC 61850 server using the ICD file.
- OPC server was not able to obtain tags via self-description from the RTU software. This was observed to be an issue with Triangle MicroWorks' Distributed Test Manager.
- Based on the 10-minute pcap, round trip time: ~200 ms between the RTU and OPC server, ~200 ms between the relay and RTU with IEC 62351 security implemented.

Key update/new session

This test case establishes how much latency will be incurred due to the key management processes to authenticate and establish secure communication.

Test Procedures:

- Configure OPC server for connection to device IP via IEC61850/MMS, and connect.
- Import ICD file into OPC server, and verify all data blocks have been imported.
- Delete prior connection.
- Configure OPC server for connection again, and connect.
- Select self-description, and verify all data blocks have been imported.
- Force new session establishment.
- Capture PCAP of the session establishment.

Test Results:

- Pass.
- Used configuration file in OPC server to load all data blocks as the self-description feature did not work on the Device Type Manager RTU software provided by Triangle MicroWorks.
-

- 10 sessions were established, and the average latency for session establishment was found to be ~10-12 ms.

Device input/output

This test case establishes that the IEC 62351 system does not interfere with control of Device I/O.

Test Procedures:

- After connection with OPC server is established, verify all analogue input data blocks are visible in OPC server.
- Verify all digital input data blocks are visible in OPC server.
- Verify all digital output data blocks are visible in OPC server.
- Use OPC server to command digital outputs to close and verify they do.
- Use OPC server to command digital outputs to open and verify they do.

Test Results:

- Pass.
- All the analogue and digital input data blocks were visible in the OPC server.
- The digital output commands sent out to the relay were also appropriately reflected both in the RTU and relays.

Loss of communication to control room

This test case establishes that the device and encryption/decryption continue to operate properly in the event of communications failure with outside devices and/or networks.

Test Procedures:

- Disconnect the connection from the control centre to the substation (OCSP and SCADA server).
- Send a control command from RTU to the relay to control I/O.
- Document behaviour.
- Force a session reestablishment between RTU and relay.
- Send a control command from RTU to the relay to control I/O.
- Document behaviour.

Test Results:

- Partial fail.
- The RTU and relay communication continued properly as expected even when the connection to the OPC server was disconnected. However, it was also observed that the Distributed Test Manager software threw an unexpected error when a session reestablishment was forced causing the secure communication to fail between the RTU and the relay.
- The Distributed Test Manager software only provides support with static CRLs, which is

a supported standard mechanism but is not the default mechanism described in the IEC 62351; the default is OCSP. Traditional CRL distribution is to utilise some online distribution mechanism (such as webpage or share) to enable updates by end devices on some set time schedule (daily, weekly). The distribution lag problem with CRLs leaves the devices in an unprotected state of allowing connections with no longer trusted identities for a period of time. OCSP on the other hand provides an online service that actively provides dynamic response on validity of certificates and identities. However, it requires constant connectivity. OCSP stapling is a technique that allows an entity to pre-grab their authenticity response from an OCSP service to overcome this connectivity issue, but it lacks the authorisation and central policy control features that Kerberos does.

Spoofing, man-in-the-middle, masquerading

This test case establishes that certificates bind identities to devices.

Test Procedures:

- Man-in-the-middle:
- Capture session establishment packets.
- Replay packets.
- Document success/failure.

Test Results:

- Pass.
- Any attempts to establish a connection to the relay from a spoofed RTU device were reset by the relay at the TCP layer by sending a reset.

Stolen device

This test case establishes

- Certificate revocation works;
- How long it takes to take effect.

Test Procedures:

- Revoke certificate for ‘stolen’ device.
- Attempt to establish a session with the ‘stolen’ device.
- Record success/failure of session establishment.

Test Results:

- Pass.
- The stolen certificate was revoked successfully, and this change was established within ~1.04s of updating the CRLs in the RTU. This test assumed instant delivery of the CRL to the devices where in general scenarios there would be a lag for distribution.

- The RTU rejected and terminated the connection attempt from a stolen relay by sending a TCP reset.

Failure/loss of OCSP/CRL server}

This test case establishes that the system can still operate even when the core service of the key management system is lost.

Test Procedures:

- Take OSCP/CRL server down.
- Attempt to establish secure connections between relay and OPC server.
- Document success/failure.

Test Results:

- Unable to complete.
- This test was not performed as the Distributed Test Manager software, which was used to simulate the IEC 62351.
- Enabled relays did not support integration with a CRL server via the network. The CRL files were generated and updated as a file upload whenever there was a change.
- Consequently, both the RTU and relay would continue to work without any issues as there is no requirement for the connection to the OCSP/CRL server to be up.

Replay attack

This test case establishes that packets with invalid or expired time stamps are rejected, that is, that a replay attack will be ineffective.

Test Procedures:

- Perform authentication/authorisation between field device OCSP and CRL server.
- Capture authorisation.
- Perform another authentication authorisation with unauthorised device.
- Replay authorisation message from OCSP to field device.
- Document success/failure of session establishment.
- Fail is the expected outcome.

Test Results:

- Pass.
- There was no response from the RTU when an unauthorised field device tried to establish a session with the RTU by replaying packets from an earlier session.