ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA02

SECURING LIGHT SOURCE SCADA SYSTEMS

Leonce Mekinda*, Valerii Bondar, Sandor Brockhauser,
Cyril Danilevski, Wajid Ehsan, Sergey Esenov, Hans Fangohr, Gero Flucke, Gabriele Giovanetti,
Steffen Hauf, David Gareth Hickin, Anna Klimovskaia, Luis Maia, Thomas Michelat,
Astrid Muennich, Andrea Parenti, Hugo Santos, Kerstin Weger, Chen Xu.
European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany

Abstract

Cyber security aspects are often not thoroughly addressed in the design of light source Supervisory Control and Data Acquisition (SCADA) systems. In general the focus remains on building a reliable and fully functional ecosystem. The underlying assumption is that a SCADA infrastructure is a closed control ecosystem of sufficiently complex technologies to provide some security through trust and obscurity. However, considering the number of internal users, engineers, visiting scientists, students going in and out light source facilities, cyber security threats can no longer be neglected. At the European XFEL, we envision a comprehensive security layer for the entire SCADA infrastructure. Karabo the control, data acquisition and analysis software developed at the European XFEL, shall implement these security paradigms known in IT but not applicable off-the-shelf to the FEL context. The challenges are considerable: (i) securing access to photon science hardware that has not been designed with security in mind; (ii) granting limited fine-grained permissions to external users; (iii) truly securing control and data acquisition APIs while preserving performance; and (iv) for integrating external data analysis applications. Only tailored solution strategies, as presented in this paper, can fulfil these requirements.

INTRODUCTION

The security of Supervisory Control and Data Acquisition (SCADA) systems is an increasing concern [1] as they nowadays interconnect a significant number of Commercial off-the-shelf (COTS) computers via IP networks. The massive integration of off-the-shelf IT technologies into the SCADA realm owes to their affordability that keeps capital and operational expenditures low, in comparison to dedicated solutions. Indeed, the fierce competition among providers of Information Technology (IT) hardware and software, the maturity of de-facto standards (x86 PC architecture, Linux, TCP/IP, USB), the availability of number of engineers and scientists already trained on these standards fuel that trend.

These proven technologies have brought countless benefits: thousands of interoperable and often free Internet services, open source software packages, plug-and-play hardware. However, with undeniable benefits come collateral weaknesses: bugs, viruses, Trojan horses, ransomwares, worms [2] like Stuxnet [3] and zero-day exploits. They can spread across networks and systems and are no longer stopped by some intrinsic incompatibility of proprietary legacy SCADA. The situational irony is that light sources, in opposition to power plant facilities, have not been classical cyber attack targets to date. However, due to the proliferation of the aforementioned threats, automated blind intrusion are as likely to occur on light source SCADA as in any other connected place. Hence, the interest of external attackers might be triggered *after* they realize that they unintentionally infected a SCADA system built on vulnerable off-the-shelf software. They may then conduct further customized attacks for challenge, malice, cyber warfare or extortion purposes.

The European X-ray Free Electron Laser [4] is a light source of peak brilliance greater than 10^{33} photons $s^{-1}mm^{-2}mrad^{-2}$ per 0.1% BW [5]. It results from the acceleration of bunches of electrons along a 1.7 km superconducting tunnel to the energy of 17.5 GeV. Throughout the SASE process [6], they deliver coherent laser-grade X-ray pulses culminating to 4.5 MHz bursts. Currently six, but extensible to ten, instruments are attached to this unique machine to perform material science and structural biology experiments only the photon energy range (from 0.2 to 25 KeV), the resolution (less than 0.1 nm) and ultra-short pulses (10 fs) of the European XFEL beam might allow. Such a 1.4 billion-euro facility [7], with unique pieces of technology (AGIPD, LPD and DSSC detectors [8–10]) producing about 15 TB of data each beam day requires special care regarding its security.

This responsibility is shared at two stages. First is the general IT stage. It is in charge of applying state-of-art best practices and tools to secure the *infrastructure* running the SCADA system. It encompasses network security (segregation, firewalling, intrusion prevention and detection) and operating system password and permission management. The second stage consists in securing the SCADA system itself. This intends to mitigate the threats posed by an attacker who accessed the Control Network. The SCADA ecosystem at the European XFEL is Karabo [11]. It is essentially a set of agents denoted *devices* interacting by remote method invocation via a central message broker. Beyond broker-centric messaging, direct TCP/IP connections ensure fast data delivery between communicating devices or between GUI clients and GUI server devices.

The key contribution of this work is on fostering in Karabo, the public-key authentication of every user to device servers whatever their access method: GUI, command line (iKarabo) or any Karabo API call. To this aim, every serialized message in the control ecosystem shall convey a signed token or the token digest.

[*] Leonce.Mekinda@xfel.eu

The paper is structured as follows: we first present the state-of-the-art in this area. Then we sketch what has been put in place at the European XFEL, before describing a proposal for hardening the security at the European XFEL, that pertains also, as we believe, to similar light sources.

RELATED WORK

To the best of our knowledge, the security of light source SCADA systems is scarcely covered by the scientific literature. One can speculate about possible reasons for this, including (i) the topic is sensitive as it may expose the weaknesses of systems under exploitation and even pave the way for potential attackers, and (ii) SCADA experts in scientific or unusual facilities may not be under the influence of the global IT culture, assuming that isolation and secrecy has been and will continue to be a sufficient shield.

Among the contributions on a closely related subject is [12]. The authors introduced by the way of partial differential equations, stealthy deception attacks on SCADA-controlled irrigation canal systems. They observe that the most common assumptions on IT security do not foresee sensor and control data compromise by an insider. Similarly, [13] demonstrated the relevance of replay attacks faking normal behaviour on cyber-physical systems. The authors proposed and analysed a detection mechanism based on adding a zero-mean Gaussian authentication noise to the optimal control.

The cyber security of electric power plants has also been investigated in [14], which reminds us that the absence of password, factory-default passwords, guest accounts and more generally poor password policies are major causes of unauthorized access to power systems.

Closer to our domain, [15] emphasizes the reality of the threat towards Accelerator Control Systems with several real world examples. The authors of [16,17] summarize the countermeasures in place in various High-Energy Physics facilities and light sources ten years ago (APS, CERN, Diamond, FNAL, NSLS, SLAC, SPring-8). Their complaint about the lack of cyber security vision in the design of Programmable Logic Controllers (PLC) is still relevant. Whenever possible, general IT security is deployed: segregated networks bridged by dual-homed SSH bastions, firewalls, IDS, access lists, anti viruses, vulnerability scanners (Nmap, Nessus) and patching, fixed IP addresses to registered hosts only, no-wireless rule.

Interestingly, the Large Hadron Collider uses Role-Based Access Control (RBAC) [18, 19], very similar to the proposal of this paper. Every authenticated user is granted a signed token via SOAP over SSL. This token is verified for adequate authorization by the control middleware of every front-end device. Gysin et al [18] make clear that what hampers the performance of the authorization chain is the token verification due to the public key size.

This paper proposes an authorization scheme that mitigates this performance cost by digesting the token with a cryptographic hash function.

AN OVERVIEW OF THE SECURITY AT THE EUROPEAN XFEL

The cyber security at the European XFEL is mainly enforced at the IT stage. A thorough deployment of IT security best practices and tools guarantees a security coverage to the grade of other facilities. We distinguish the Office Network from the Control Network. A bastion grants access to the Office Network. Inside this already secure environment, access to the Control Network must be requested through another bastion. Within the Control Network, the access to Control Servers is restricted to a group of experts. Sensitive hardware is segregated and only reachable from dedicated dual-homed hosts. The whole infrastructure forms a "security onion" sketched in Figure 1.

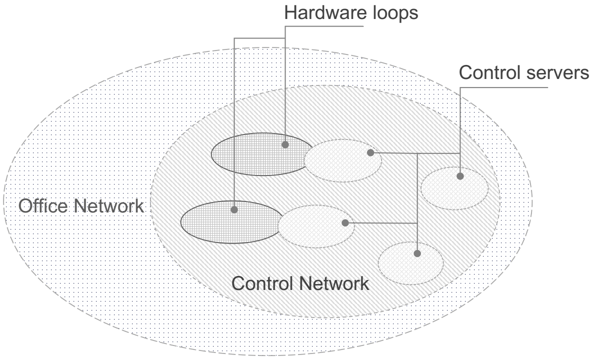


Figure 1: The security onion: at the core of the secure network are the hardware loops, physically isolated. The access to the control servers is restricted to a handful of functional accounts. The control network is isolated from the office network but accessible through bastions. Another bastion may allow remote access to the the secure office network.

Authentication

Every login, file transfer, code versioning or deployment runs over Secure Shell (SSH). Diffie-Hellman drives the session key exchange with SHA-2 as preferred cryptographic hash function. AES-CBC (Rijndael) [20], which is among the strongest ciphers, encrypts the session. Indeed, it was selected by the U.S. National Institute of Standards and Technology to protect information classified SECRET and TOP SECRET. Then the host key is authenticated using RSA to avoid spoofing. A valid password or a Kerberos [21] ticket is always required.

Above the general IT security lies Karabo authentication. Upon successful authentication via Kerberos (Figure 2), a Karabo user is granted an access level to the Karabo ecosystem. Five access levels (Figure 3) from “Observer” to “Administrator” restrict the user visibility to sets of device servers, devices or properties, within the GUI client. More precisely, when a user provides a name and a password in a Karabo login window, these parameters are conveyed though the Simple Object Access Protocol (SOAP) to an authentication web service. The HTTP communication carried over Transport Layer Security (TLS). When authenticated, users receive a signed token that encloses their global access level and a list of exceptions per device.

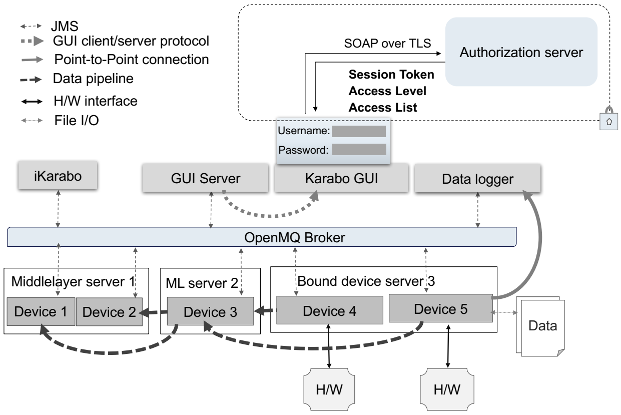


Figure 2: Login to the Karabo ecosystem. The Karabo ecosystem is protected by the underlying secure IT infrastructure.

Denial of Service Mitigation

Distributed systems are collaborative to a certain extent. Any intentional or unintentional misbehaviour of one of its components might jeopardize the whole. This is the reason why the deployment of any Karabo device must be subject to the careful code review of the Control and Analysis Software (CAS) experts.

In Karabo, we apply network separation and provide a specific bridge [22] for allowing external users and their scientific data analysis code to be incorporated into the control system itself. This is needed as they usually wish to get near real time data analysis feedback for the control of their experiments [23–25].

The clustering of message brokers, added to a proper dimensioning of broker resources and thresholds is necessary to drop unusual traffic patterns. Automated tools frequently prune the file system of control servers hosting control message loggers. Old log files are compressed and archived on a remote GPFS (General Parallel File System) volume. As these logs are plain ASCII files with rather low entropy, they are usually shrunk with a lossless compression ratio of 10:1. The GPFS volume size available at the European XFEL guarantees that such archives could be stored for a lifetime.

Beside automated mitigation means, we relentless monitor the Control Network to track misbehaving agents. Mail alerts, alarms and logs report abnormal resource consumption, while an alarm system reports abnormal software and hardware behaviours.

More fundamentally, a broker-independent architecture is under preparation to circumvent that single point of failure.

A NEW SECURITY LAYER FOR KARABO

In what follows be propose a design to address the challenges outlined above.

The secure token every authenticated user receives must be used throughout the Karabo ecosystem to grant visibility and authorize actions. However, it must be made sure that these privileges can not be forged by a malicious insider or any attacker who broke into the Control Network. For example, moving a robotised arm, deflecting the intense X-ray beam toward unprotected part of a million-euro detector, modifying vacuum settings, monitoring an experiment must be protected operations.

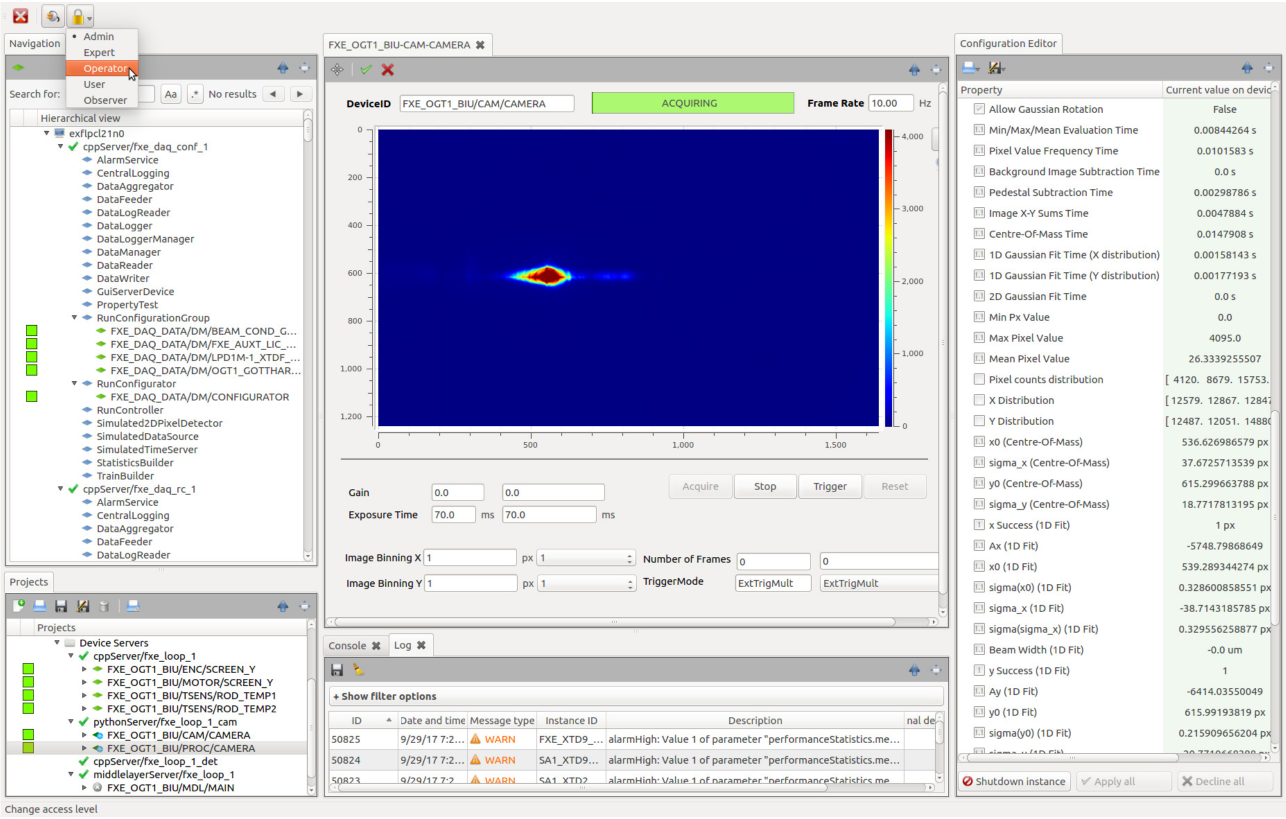
In the absence of a cryptographic authorization, reverse-engineering a few applications of the SCADA system might be sufficient for privilege escalation. For this reason, we propose a lightweight Public-Key Infrastructure that would allow device servers to authenticate and authorize user operations without sacrificing the overall SCADA performance. Such a secure layer could be seamlessly inserted into the current ecosystems.

A Public-Key Infrastructure (PKI) is a set of algorithms, protocols, software and tiers that make possible the use of asymmetric encryption and signature in a distributed ecosystem. In a public-key or asymmetric cryptosystem, a public key is used to encrypt a message only the owner of the private key can decrypt. Similarly, a public key is used to verify a signature only the owner of the private key could have made.

Public-key encryption is known to have a high computational cost due to its mathematics. For instance, Rivest-Shamir-Adleman (RSA) [26] performs modular exponentiations with very large exponents (more than 2048-bit integers). This paper's ultimate proposal denoted Karabo advanced PKI architecture aims at alleviating this issue.

We present along the way the rationale behind its construction by iteration over intermediate models. The milestones of this progressive modelling are:

- The simple PKI architecture where a signed token is verified by every device but can be replayed.
- The once-d (nonced) PKI architecture where a signed token is made unique to a device by an initialization nonce but can be replayed by an eavesdropper on that device.
- The nonced PKI with digest architecture where nonced tokens are no longer systematically sent for authorization but their cryptographic hash, for faster processing. However, like the previous approach, it does not resist to user identity spoofing.
- The **advanced PKI** architecture gathering the lessons learned from previous models. Here every message sent during a session is authenticated by a different cryptographic hash to prevent digest replay attacks (Figure 8 and 7). Note that *the messages themselves are not encrypted*. They are simply signed.



exflpcl21n0
cppServer/fxe_daq_conf_1
AlarmService
CentralLogging
DataAggregator
DataFeeder
DataLogReader
DataLogger
DataLoggerManager
DataManager
DataReader
DataWriter
GuiServerDevice
PropertyTest
RunConfigurationGroup
FXE_DAQ_DATA/DM/BEAM_COND_G...
FXE_DAQ_DATA/DM/FXE_AUXT_LIC_...
FXE_DAQ_DATA/DM/LPD1M-1_XTDF_...
FXE_DAQ_DATA/DM/OGT1_GOTTHAR...
RunConfigurator
FXE_DAQ_DATA/DM/CONFIGURATOR
RunController
Simulated2DPixelDetector
SimulatedDataSource
SimulatedTimeServer
StatisticsBuilder
TrainBuilder
cppServer/fxe_daq_rc_1
AlarmService
CentralLogging
DataAggregator
DataFeeder
DataLogReader

Projects
Device Servers
cppServer/fxe_loop_1
FXE_OGT1_BIU/ENC/SCREEN_Y
FXE_OGT1_BIU/MOTOR/SCREEN_Y
FXE_OGT1_BIU/TSENS/ROD_TEMP1
FXE_OGT1_BIU/TSENS/ROD_TEMP2
pythonServer/fxe_loop_1_cam
FXE_OGT1_BIU/CAM/CAMERA
FXE_OGT1_BIU/PROC/CAMERA
cppServer/fxe_loop_1_det
middlelayerServer/fxe_loop_1
FXE_OGT1_BIU/MDL/MAIN

DeviceID FXE_OGT1_BIU/CAM/CAMERA ACQUIRING Frame Rate 10.00 Hz

Gain 0.0 0.0
Exposure Time 70.0 ms 70.0 ms
Image Binning X 1 px 1
Image Binning Y 1 px 1
Number of Frames 0 0
TriggerMode ExtTrigMult ExtTrigMult

ID	Date and time	Message type	Instance ID	Description	nal de
50825	9/29/17 7:2...	WARN	FXE_XTD9_...	alarmHigh: Value 1 of parameter "performanceStatistics.me...	
50824	9/29/17 7:2...	WARN	SA1_XTD9...	alarmHigh: Value 1 of parameter "performanceStatistics.me...	
50823	9/29/17 7:2	WARN	SA1_XTD2	alarmHigh: Value 1 of parameter "performanceStatistics.me	

Property	Current value on devic
☑ Allow Gaussian Rotation	False
Min/Max/Mean Evaluation Time	0.00844264 s
Pixel Value Frequency Time	0.0101583 s
Background Image Subtraction Time	0.0 s
Pedestal Subtraction Time	0.00298786 s
Image X-Y Sums Time	0.0047884 s
Centre-Of-Mass Time	0.0147908 s
1D Gaussian Fit Time (X distribution)	0.00158143 s
1D Gaussian Fit Time (Y distribution)	0.00177193 s
2D Gaussian Fit Time	0.0 s
Min Px Value	0.0
Max Pixel Value	4095.0
Mean Pixel Value	26.3339255507
Pixel counts distribution	[4120. 8679. 15753.
X Distribution	[12579. 12867. 12847
Y Distribution	[12487. 12051. 14880
x0 (Centre-Of-Mass)	536.626986579 px
sigma_x (Centre-Of-Mass)	37.6725713539 px
y0 (Centre-Of-Mass)	615.299663788 px
sigma_y (Centre-Of-Mass)	18.7717813195 px
x Success (1D Fit)	1 px
Ax (1D Fit)	-5748.79868649
x0 (1D Fit)	539.289344274 px
sigma(x0) (1D Fit)	0.328600858551 px
sigma_x (1D Fit)	-38.7143185785 px
sigma(sigma_x) (1D Fit)	0.329556258877 px
Beam Width (1D Fit)	-0.0 um
y Success (1D Fit)	1
Ay (1D Fit)	-6414.03550049
y0 (1D Fit)	615.99193819 px
sigma(y0) (1D Fit)	0.215909656204 px

Figure 3: Karabo GUI client access: five access levels to Karabo ecosystems: Observer, User, Operator, Expert and Admin. In this example the Beam Imaging Unit of the Femtosecond Crystallography Experiments (FXE) diagnoses the alignment of the X-ray beam. Administrative credentials are mandatory for turning the interlock surveillance on or off on a PLC.

These architectures have in common three actors:

- Users being granted access to the Control Network.
- The Certification Authority (CA) / Authorization Server. It is primarily in charge of generating and signing authorization tokens with its private key. Its public key is known to all other actors for signature verification.
- Devices or device servers, in charge of token verification. They authorize user operations in accordance with the enclosed global access level and exception list.

Karabo Simple PKI Architecture

A first naive approach to a public-key architecture has been investigated. In this architecture, every Karabo installation includes the public key of an Authorization Server (acting as Certification Authority). It authenticates the user via Kerberos and replies with a signed token *@token* that encloses a Kerberos session token, the user global access Level, an exception list per device and an expiration date. Every API call presents this signed token to every device to get accepted (for example, by appending it to every Karabo Hash before serialization and transfer). Note that a Karabo Hash [11] is a sort of multi-purpose cross-language dictionary. Every devices can verify the validity of user tokens using the Certification Authority's public key. Figure 4 depicts the overall architecture.

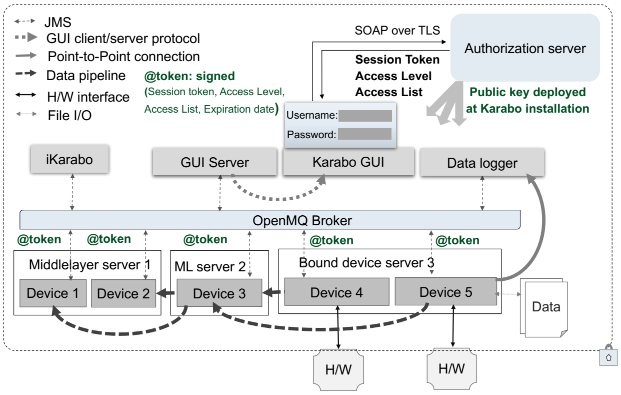


Figure 4: Karabo simple PKI architecture: thanks to the signed token, every device can authenticate a user. This allows to verify the the signature on every token offline. However, this architecture is vulnerable to cross-server token **replay** attacks.

Unfortunately, a rogue device or an attacker passively listening to the network traffic can capture *@token* and act on behalf of the authenticated user. The attacker is called a **man-in-the-middle**. He can **replay** the token across servers *i.e.* insert the token in its own messages and being granted the privileges of the original user.

Karabo Nonce-d PKI Architecture

This approach is the next iteration on the simple PKI architecture. Here also, every Karabo installation includes the public key of the Authorization Server (acting as Certification Authority). On user connection, every device server i is requested to provide a nonce $s[i]$ it remembers (Figure 5).

A **nonce** is a random number that the Certification Authority combines with the token before signing it. It ensures that an eavesdropper can not be granted the user privileges by replaying that token on a different server. The Authorization Server replies with a set of signed tokens $@stokens[]$. Each one encloses a Kerberos session token, the global access level, the list of exceptions for a device, an expiration date, and the nonce $s[i]$. Every subsequent message presents that nonced-and-signed token $@stokens[i]$ to the relevant device server to get accepted. For this, it might be appended to every Karabo Hash before serialization and transfer. Every device server can verify the validity of user tokens using the Certification Authority's public key.

However, adversaries are still able to replay a captured token, spoof the user identify on the server this token was generated for and escalate their privileges. Moreover, the signature verification in a public-key cryptosystem is computationally expensive. This cannot be performed on every single message at wire speed. For this reason, tokens of authorized users should be put in cache. Still, large signed tokens need to be conveyed with every single message and compared against those in cache. In addition, the requirement to fetch a nonce from every device server on user connection slows down their login to the ecosystem.

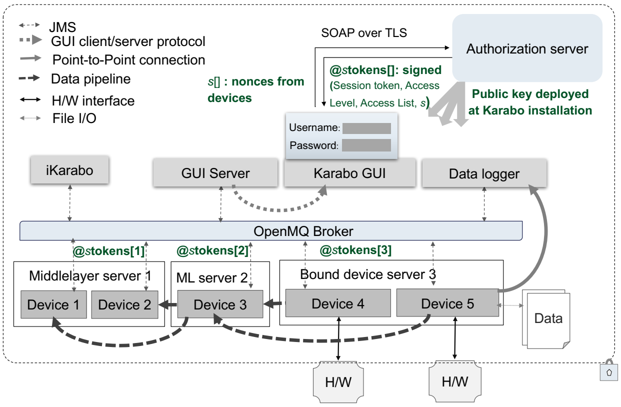


Figure 5: Karabo nonced PKI architecture: every **device server** can authenticate users. The token signature can be verified offline. This architecture is robust against cross-server token replay. Nonetheless, it is still vulnerable to token replay obtained by a **man-in-the-middle**, in addition to causing slower connections.

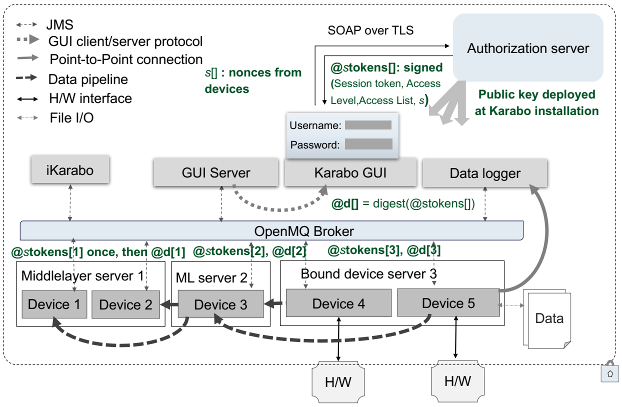


Figure 6: Karabo nonced PKI with digest architecture: beyond offline token signature verification and robustness against cross-server token replay, this architecture increases performance in checking the digests of nonced tokens. Unfortunately it remains vulnerable to token interception by man-in-the-middle attack and privilege escalation.

Karabo Nonce-d PKI With Digest Architecture

An improvement to the previous approach would consist in alleviating the token verification overhead. Indeed, as the signature verification is an expensive process, the nonced tokens $@stoken[]$ shall only be presented once to the device servers. Then a cryptographic hash function like SHA-256 shall generate digests $@d[]$ of these tokens. The digests shall be presented to the device server in every subsequent message (Figure 6).

A nonced cryptographic digest cannot be forged. This makes it interesting for authentication. Indeed, the one-way function generating it guarantees an extremely low collision probability and the infeasibility to infer the original token and nonce from their digest. This second property is not important in the current approach since $@d[]$ is plaintext anyway, but it will be useful in the next and final architecture.

A digest is much smaller than the signed token itself. A device server can use them to quickly find in a hash table the cached token of a user it has already authorized.

Despite these benefits, the nonced PKI approach is still vulnerable to digest replay attacks. The digest $@d[i]$ of an administrator can be intercepted and serve any attacker to act as an administrator on the device server i.

Karabo Advanced PKI Architecture

Reusing the insight gained from the preceding attempts, the architecture we advocate would include the public key of the Certification Authority in every Karabo installation. *Every valid Karabo device server shall possess a private/public key pair*. The public key shall be signed by the Certification Authority. This signature on a public key makes it a certificate.

As shown in Figure 8, on user connection (from a client) to the Karabo ecosystem, the Certification Authority replies with a signed $@token$ that contains the result of an OAuth2

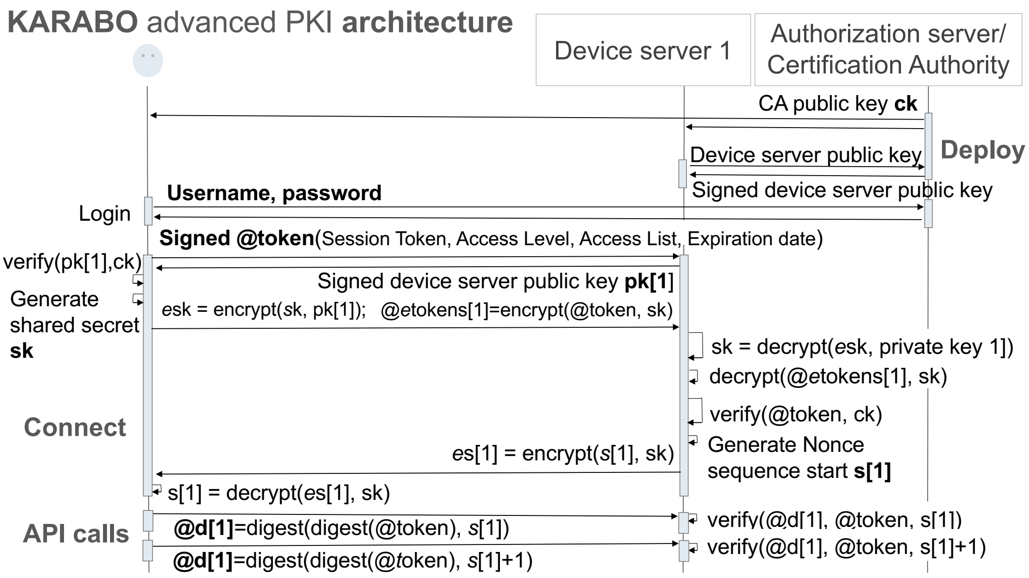

KARABO advanced PKI architecture
Device server 1
Authorization server/ Certification Authority
CA public key ck
Deploy
Device server public key
Signed device server public key
Login
Username, password
Signed @token(Session Token, Access Level, Access List, Expiration date)
verify(pk[1],ck)
Signed device server public key pk[1]
Generate shared secret sk
esk = encrypt(sk, pk[1]); @etokens[1]=encrypt(@token, sk)
sk = decrypt(esk, private key 1])
decrypt(@etokens[1], sk)
verify(@token, ck)
Connect
Generate Nonce sequence start s[1]
es[1] = encrypt(s[1], sk)
s[1] = decrypt(es[1], sk)
API calls
@d[1]=digest(digest(@token), s[1])
verify(@d[1], @token, s[1])
@d[1]=digest(digest(@token), s[1]+1)
verify(@d[1], @token, s[1]+1)

Figure 7: Karabo advanced PKI architecture: interaction diagram.

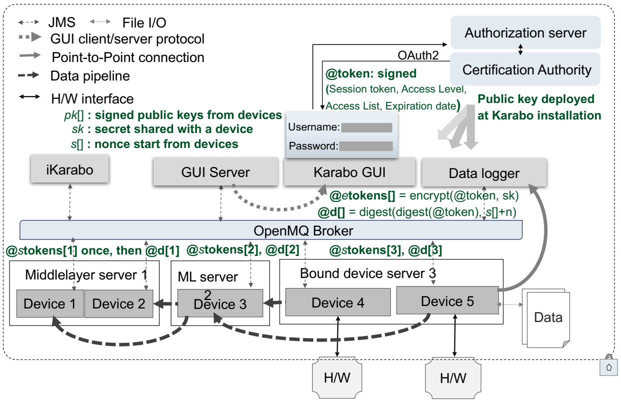

JMS
File I/O
GUI client/server protocol
Point-to-Point connection
Data pipeline
H/W interface
pk[] : signed public keys from devices
sk : secret shared with a device
s[] : nonce start from devices
Authorization server
Certification Authority
OAuth2
@token: signed (Session token, Access Level, Access List, Expiration date)
Public key deployed at Karabo installation
Username:
Password:
iKarabo
GUI Server
Karabo GUI
Data logger
@etokens[] = encrypt(@token, sk)
@d[] = digest(digest(@token), s[]+n)
OpenMQ Broker
@stokens[1] once, then @d[1]
@stokens[2], @d[2]
@stokens[3], @d[3]
Middlelayer server 1
ML server 2
Bound device server 3
Device 1
Device 2
Device 3
Device 4
Device 5
Data
H/W
H/W

Figure 8: Karabo advanced PKI architecture: device servers and users can mutually authenticate using the CA public key. The SCADA performance is preserved by the verification of signed token digests in place of the original user tokens. This architecture resists replay attacks as every message must carry a distinct digest.

transaction: a user session token, a global access level, the list of exceptions per device and an expiration date. On client connection to device server, the client verifies the device server public-key validity thanks to the Certification Authority's public key. Then the client generates a shared secret sk, encrypts it with the device server public key, encrypts $@token$ with the shared secret using a block cipher such as AES-256, and sends them all to the device server. The device server decrypts the share secret using its own private key and decrypts the token using the shared secret. Every **certified** device server can decrypt and verify user tokens using the CA's public key. Then, the device server generates a **nonce sequence start**, encrypts it with the shared secret and sends it to the client. To circumvent the expensive signature verification procedure, the digests of tokens $@token[]$ are nonced *for every message*, hashed and presented to a device server as cryptographic digests $@d[]$. The server can then pick in its cache the related token. An orchestrating device inherits from the last command issuer's privileges. Every single operation shall be logged along with the issuer identity. Digest replay attacks are mitigated by the incrementation of the nonce in every message. Client and device server both know the next nonce and can calculate the next digest. In case of message loss, the client can request a nonce reset. Observe that $@token$ is known to the certification authority, the client and the certified device servers. To resist brute force attacks perpetrated from a potentially compromised device server, the nonce sequence start must be a large integer (> 32 bits).

The whole interaction is summarized by the diagram in Figure 7.

CONCLUSION

Cyber security is a growing concern in an interconnected world. If even simple home computers are often attacked, light sources, which are often publicly praised, must be considered at risk. In this paper, we propose to shield the SCADA ecosystem beyond general IT security measures in securing every operation onto the device servers. This primarily aims at limiting the severity of the harm an adverse insider behaviour or an intruder may cause.

The cryptographic scheme we designed should not degrade the overall system performance. Messages are not encrypted but signed. In our advanced Public-Key Infrastructure proposal for Karabo, every user shall access the SCADA using a token signed by a Certification Authority. Device servers have their public key signed by this same Certification Authority. Users communicate their session token only to certified device servers, encrypted with the device server public key. Therefore, the session token is only known to the certification authority, the user and the certified device servers. A different nonced digest of the session token is sent within every exchanged message for preserving performance while preventing man-in-the-middle attacks.

REFERENCES

[1] H. Sandberg, S. Amin, and K. H. Johansson, "Cyberphysical security in networked control systems: An introduction to the issue," *IEEE Control Systems*, vol. 35, no. 1, pp. 20–23, 2015.

[2] J. Park, J. Noh, M. Kim, and B. B. Kang, "Invi-server: Reducing the attack surfaces by making protected server invisible on networks," *Computers & Security*, vol. 67, pp. 89–106, 2017.

[3] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, 2011.

[4] M. Altarelli, R. Brinkmann, M. Chergui, W. Decking, B. Dobson, S. Düsterer, G. Grübel, W. Graeff, H. Graafsma, J. Hajdu, *et al.*, "The european x-ray free-electron laser," *Technical design report, DESY*, vol. 97, pp. 1–26, 2006.

[5] M. Altarelli and A. P. Mancuso, "Structural biology at the european x-ray free-electron laser facility," *Phil. Trans. R. Soc. B*, vol. 369, no. 1647, p. 20130311, 2014.

[6] E. Saldin, E. Schneidmiller, and M. V. Yurkov, *The physics of free electron lasers*. Springer Science & Business Media, 2013.

[7] E. Cartlidge, "European XFEL to shine as brightest, fastest x-ray source." American Association for the Advancement of Science, 2016.

[8] T. Hatsui and H. Graafsma, "X-ray imaging detectors for synchrotron and XFEL sources," *IUCrJ*, vol. 2, no. 3, pp. 371–383, 2015.

[9] M. Kuster, D. Boukhelef, M. Donato, J.-S. Dambietz, S. Hauf, L. Maia, N. Raab, J. Szuba, M. Turcato, K. Wrona, *et al.*, "Detectors and calibration concept for the european XFEL," *Synchrotron radiation news*, vol. 27, no. 4, pp. 35–38, 2014.

[10] U. Trunk, A. Allahgholi, J. Becker, A. Delfs, R. Dinapoli, P. Göttlicher, H. Graafsma, D. Greiffenberg, H. Hirsemann, S. Jack, *et al.*, "AGIPD: a multi megapixel, multi megahertz x-ray camera for the european XFEL," in *31st International Congress on High-Speed Imaging and Photonics*, pp. 1032805–1032805, International Society for Optics and Photonics, 2017.

[11] B. Heisen, D. Boukhelef, S. Esenov, S. Hauf, I. Kozlova, L. Maia, A. Parenti, J. Szuba, K. Weger, K. Wrona, *et al.*, "Karabo: An integrated software framework combining control, data management, and scientific computing tasks," in *14th International Conference on Accelerator & Large Experimental Physics Control Systems, ICALEPCS2013. San Francisco, CA*, 2013.

[12] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, "Stealthy deception attacks on water SCADA systems," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pp. 161–170, ACM, 2010.

[13] Y. Mo, R. Chabukswar, and B. Sinopoli, "Detecting integrity attacks on scada systems," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1396–1407, 2014.

[14] C.-W. Ten, C.-C. Liu, and G. Manimaran, "Vulnerability assessment of cybersecurity for SCADA systems," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1836–1846, 2008.

[15] S. M. Hartman, "Protecting accelerator control systems in the face of sophisticated cyber attacks," *Oak Ridge National Laboratory*, 2012.

[16] S. Lüders, "Summary of the control system cyber-security (CS) 2/HEP workshop," tech. rep., 2007.

[17] S. Lüders, "Securing control systems against cyber attacks," *Proceedings of PAC2009, Vancouver, BC, Canada*, pp. 1785–1789.

[18] S. Gysin, A. Petrov, P. Charrue, W. Gajewski, V. Kain, K. Kostro, G. Kruk, S. Page, and M. Peryt, "Role-based access control for the accelerator control system at cern," *K. Kostro et al.,"Role-Based Authorization in Equipment Access at CERN*, 2007.

[19] I. Yastrebov and N. Yastrebova, "Securing controls middleware of the large hadron collider," *Computing and Informatics*, vol. 31, no. 6, pp. 1151–1172, 2013.

[20] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[21] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994.

[22] Fangohr, Beg, Bondar, Boukhelef, Brockhauser, Danilevski, Ehsan, Esenov, Flucke, Giovanetti, Hauf, Heisen, Hickin, Klimovskaia, Kuster, Lang, Maia, Mekinda, Michelat, *et al.*, "Data analysis support in Karabo and European XFEL," *Proceedings of ICALEPCS2017, Barcelona, Spain*, 2017.

[23] M. Basham, J. Filik, M. T. Wharmby, P. C. Chang, B. El Kassaby, M. Gerring, J. Aishima, K. Levik, B. C. Pulford, I. Sikharulidze, *et al.*, "Data analysis workbench (DAWN)," *Journal of synchrotron radiation*, vol. 22, no. 3, pp. 853–858, 2015.

[24] U. Zander, G. Bourenkov, A. N. Popov, D. De Sanctis, O. Svensson, A. A. McCarthy, E. Round, V. Gordeliy, C. Mueller-Dieckmann, and G. A. Leonard, "Meshandcollect: an automated multi-crystal data-collection workflow for synchrotron macromolecular crystallography beamlines," *Acta Crystallographica Section D: Biological Crystallography*, vol. 71, no. 11, pp. 2328–2343, 2015.

[25] S. Brockhauser, R. B. Ravelli, and A. A. McCarthy, "The use of a mini-κ goniometer head in macromolecular crystallography diffraction experiments," *Acta Crystallographica Section D: Biological Crystallography*, vol. 69, no. 7, pp. 1241–1251, 2013.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.