

An Access Control Protocol for Embedded Devices

Martin Naedele
ABB Corporate Research
CH-5405 Baden-Dättwil, Switzerland
martin.naedele@ch.abb.com

Abstract—Embedded devices are today important elements of critical infrastructures like networking, telecommunications, electric/gas/water utility automation, and industrial automation. Access to these embedded devices should be restricted and strictly controlled. Current authentication schemes for embedded devices - mainly password based - do not provide the necessary security and scalable manageability. This paper presents the specific requirements for authentication and access control to embedded devices and proposes a novel, public-key based protocol for secure access and communication to embedded servers.

I. INTRODUCTION

A. Motivation

Embedded devices are today important elements of critical infrastructures like networking (routers, managed switches, firewalls), telecommunications, electric/gas/water utility automation (intelligent electronic devices, communicating e.g. using IEC 61850 [1]), and industrial automation (PLCS, process controllers, drives robot controllers).

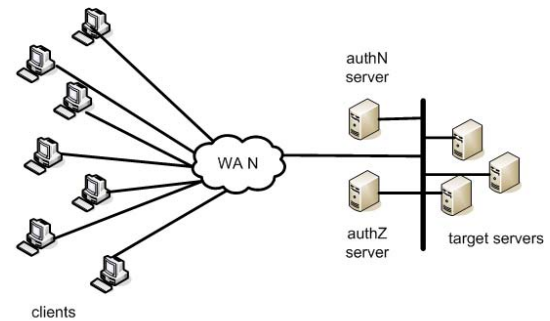
During their operational lifetime, these embedded devices have to be accessed directly and/or remotely by human users and software processes to issue commands, to obtain measurements or status information, to diagnose failures, and to change settings and applications.

As these devices are critical for their respective system, access to them should be restricted and strictly controlled. As will be explained later in this section, current authentication schemes for embedded devices - mainly password based - do not provide the necessary security and scalable manageability.

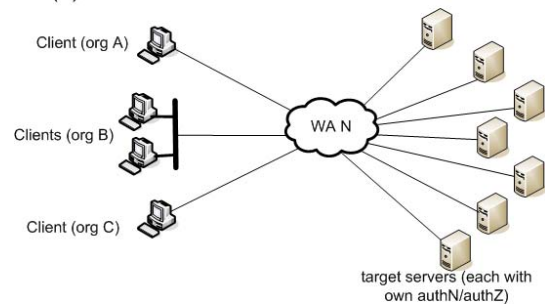
When discussing authentication and access control mechanisms one has to differentiate between two fundamentally different usage scenarios: the commercial scenario and the embedded device scenario:

In the commercial environment (see Fig. 1a), which includes office work and e-commerce applications, hundreds or thousands of users access a relatively small set of services and servers in a limited number of locations. Both the users and the servers are member of a single or very small number of authentication domains (e.g. Windows or Kerberos domains). In case of multiple authentication domains, single-sign on schemes may be used to emulate a single authentication domain to the user. User accounts can be managed efficiently because they are stored on a limited number of centralized authentication servers to which application servers refer the authentication decision via networks that are always available.

In the embedded device environment (see Fig. 1b), in contrast, there is only a comparatively small number of human



(a) Office/commercial remote access scenario



(b) Embedded device remote access scenario

Fig. 1. Key characteristics of the authentication and access control mechanisms for commercial and embedded device scenarios.

users (operators, maintenance staff) and client workplaces. These users typically belong to multiple organizations, and each user, e.g. a vendor service engineer, must have the ability to access a large number of embedded devices acting as servers. These embedded servers are distributed physically and organizationally, and thus belong to different authentication domains. In fact, in the typical embedded environment today each embedded device is its own authentication domain. Each contains its own user base, because of a historical need for each device to be able to operate in full independence of other hosts and outside communication links to maximize resiliency and dependability of the system/plant controlled by this embedded device. Such decentralized user management faces the additional challenge that the local storage and computational capabilities of embedded devices are typically much lower than those of commercial authentication and access control servers.

B. Requirements

Due to the high demands on availability that are imposed on embedded devices, trade-off decisions on selecting authentication and access control options for embedded devices have so far deemphasized security. There is thus a strong need for a secure and efficiently manageable authentication and access control scheme for embedded devices. Such a scheme has to meet the following requirements:

a) *Access revocation*: It should be feasible to change or revoke the access rights of a user to some or all embedded servers.

b) *Centralized user management*: Changing or revoking of access rights should be possible without having to touch/reconfigure each embedded server. This requires a centralized user and access rights management.

c) *Individual accountability*: Each user access to a device should be traceable to an individually identifiable user (no shared or group credentials).

d) *Offline access control*: The access control scheme should support networked devices as well as embedded servers that are isolated and only accessible via front panel or direct console port access. Main functionality and local emergency device access must not depend on the availability of a central server or communication infrastructure.

e) *Protocol security*: Transmitted credentials have to be protected against modification and replay.

f) *Session protection*: In case of remote access to the embedded server the integrity, and optionally confidentiality, of the communication should be protected.

g) *Simple key management*: There will possibly be hundreds of embedded servers per system, some of them offline in unmanned stations, and these devices should be replaceable in case of failure with minimum configuration effort. Therefore the system should work without the need for distributing secret keys to the embedded servers.

h) *No dependency on security of individual servers*: The scheme will be used with widely distributed embedded servers for which protection against physical attacks can not be guaranteed. It has to be assumed that an attacker may be able to gain physical control over individual servers, but this should not enable the attacker to gain access rights for other servers.

C. Contribution

This paper proposes a novel public key based authentication and access control protocol that meets the security and manageability requirements for access control to embedded devices.

II. PREVIOUS WORK

A. Authentication and access control protocols

A number of different authentication and access control schemes exist in the literature (e.g. [2], [3], [4]) as well as in widely deployed IT systems.

Password-based access control and authentication directly on the embedded device is the most common mechanism

today. It suffers from several major weaknesses: Access is in practice not revokable, because it is based on knowledge, and reconfiguring all affected servers would be impracticable. Also, storage limitations on the devices typically limit the number of user accounts and thus require group credentials, which prevents individual accountability. If users use the same password for multiple devices then the compromise of a single device leads to a compromise of the whole system.

The best known centralized scheme is Kerberos [5], [6], which is also used for Windows domain authentication. Kerberos works with capabilities called "tickets". After successful authentication the user obtains a "ticket granting ticket", which can later be used to obtain an actual ticket for a target server from one or more ticket granting servers. The target server itself only checks the ticket, but not the actual user identity, for its access control decision. Kerberos uses shared secrets and symmetric cryptography to protect authenticity and confidentiality of communication between participants. Every target server has to be configured with an individual secret key which it shares with the central authentication server. This does not scale for embedded device scenarios with hundreds of devices. Kerberos itself is only concerned with authentication and access control, but not with protecting the resulting session.

Variations of Kerberos with public key cryptography have been proposed [7], [8]. They separate the public key based authentication process from the access control process but do not separate the access control process from the access to the target service, as is proposed in this paper. With these protocols it would still be necessary for each server to maintain access control lists for each user.

SPKI [9], [10] is a public key based authentication and access control protocol and authentication message format. Main focus is on sophisticated rights delegation and derivation algorithms. SPKI provides no means for session protection against tampering or replay. The SPKI message format may be used within the context of the protocol described in this paper to transport capabilities.

SAML [11] is an XML based syntax for encoding capabilities, which may potentially be used within the context of the protocol described in this paper to transport capabilities, but is limited due to the fact that standard SAML can only express yes/no type of access decisions, no complex permission statements. SAML also defines a number of authentication and authorization transfer protocols for single sign-on and server federation, but these require online access of the server device to the authentication and access control server.

X509 [12] is a standard to encode digital certificates and additional attributes which may be used within the context of the protocol described in this paper to transport capabilities.

SSH [13] is a session protection protocol with an optional public key based authentication scheme. This scheme does not scale for our approach, because access control is based on ACLs stored on the server devices and for authentication the public key of each legitimate user has to be preconfigured on each server device.

In sensor networks a multitude of sensors communicate as peers with each other and each sensor is constrained in its computational power, which precludes use of asymmetric cryptography. Here secret key schemes are used to create authenticated communication relations between certain nodes. Due to the peer-to-peer communication model there are multiple possible paths between each node. Statistical considerations are used to reduce the number of necessary node-pair secrets while still guaranteeing secure paths between any arbitrary pair of nodes. The main research focus in sensor network authentication is on resiliency in the face a partly compromised network [14].

B. Industrial access control protocols

Access control mechanisms in existing industrial field bus technologies are discussed in [15]. [16] proposes the use of a public key infrastructure (PKI) in an industrial environment. Aspects of digest-based password authentication for access to web servers in embedded systems are examined in [17]. A security scheme for distributed control systems based on IEC61499 function blocks is proposed in [18]. The proposed system uses either passwords or a certificate/PKI protocol for authentication. As access control is apparently still the responsibility of each server, the scalability problem is not solved.

III. PROPOSED SCHEME

A. Protocol

1) *Preliminaries:* The protocol assumes the following system setup: There are embedded servers S, clients C, and an access authority server AA.

All the embedded servers S hold the public key pub_{AA} of the access authority server. Note that this is the same key for all devices, so it can be efficiently pre-installed, and it is the public key, thus not a secret. A (physical) compromise of any server S does not give an attacker information to compromise the protocol. Each server also knows some kind of identifier Id_S for itself. S and AA share a common notion of time with a certain degree of synchronicity (technically realized e.g. via GPS). Synchronicity deviations may cause false negatives and false positives on expiration time verification.

Each client has its own key pair $pub_C/priv_C$ of which $priv_C$ is secret and only known to that client C.

The access authority server AA has its own key pair $pub_{AA}/priv_{AA}$ of which $priv_{AA}$ is secret and only known to AA. It also holds additional information needed to make access control decisions:

- $R(C, S)$ is a matrix of access rights for all clients C on all servers S. The specific rights in the matrix may be generic (read/write/update/delete) or application/server specific. They only need to be interpretable by the respective server S and can thus be different for each server. This way, the same scheme can be used to manage access control for a variety of different devices.
- $T_{exp}(C, S)$ is a rule set determining the expiration time of capabilities granted to a client C for access to a server

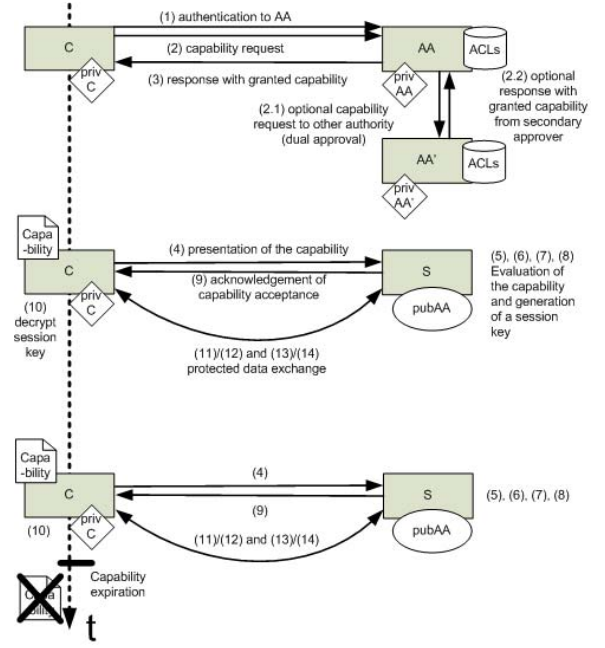


Fig. 2. Steps of the proposed authentication and access control protocol.

S. The actual expiration time on a capability may depend on C, S, the requested access rights, or contextual information like the intended method of accessing the server (online, or offline via direct physical/console access).

- pub_C for all registered clients C. This is manageable because there is only one central location where these keys need to be registered.

It is also assumed that there is a mechanism in place that allows the clients to authenticate to the AA server and communicate with AA in a sufficiently secure way. For this purpose, the scalability issue mentioned in Section I-A is not a concern, because the clients are authenticating themselves to a single, central server. Common approaches like Windows domain authentication or Kerberos can be used for this purpose.

In the following, the individual steps of the protocol will be explained (see Fig. 2).

2) Requesting a capability:

- 1) C successfully authenticates to AA.
- 2) C requests a capability, self contained, transportable, offline usable set of permissions, with access rights R' and expiration time T'_{exp} for the target server identified by Id_S . Note that Id_C may be an explicit part of the request message or may be obtained by AA from contextual information out of the authenticated session between C and AA.

$$C \rightarrow AA: \{Id_S, R', Id_C, T'_{exp}\}$$

- 3) AA sends the granted capability to C. The capability is signed with the private key of AA. Note that in practice only the hash will be signed. The unique identifier Id_C of client C is included to enable the target server to log the actual client even though S does not itself authenticate C, in order to allow retrospective identification

of the client user if necessary. The public key pub_C of the client C is included for use in later message exchanges between C and S, so that S can be sure only C (defined by ownership of $priv_C$) will be able to decrypt messages encrypted with pub_C . The unique identifier of the addressed server Id_S is necessary to allow the server to verify that the capability is issued for itself, and not for any other server. $R'' = \{R1, R2, R3, \dots\}$ indicate the set of granted access rights formulated in any access rights language acceptable for and understandable by the specific server S. R'' is calculated as the intersection of the requested rights R' and the rights available to the client according to the authorization database $R(C, S)$. T''_{exp} specifies the expiration time of this capability. T''_{exp} is a function of the requested expiration time T'_{exp} and the expiration time calculation rule set $T_{exp}(C, S)$.

$C \leftarrow AA$:

$Cap_{C,S} = \{Id_C, pub_C, Id_S, R'', T''_{exp}\}priv_{AA}$

As mentioned above, authentication of C to AA and vice versa and protection (confidentiality, integrity) of these messages are outside the scope of this protocol.

- 3) *Initiating a session with the target server:*
- 4) Client C sends the capability to target server S. This may be done over a network or offline via direct console or front panel access.

$C \rightarrow S: Cap_{C,S}$

- 5) S verifies the signature of AA on the capability using the public key of AA.

$S: \{Cap_{C,S}\}pub_{AA}$

$=? = \{Id_C, pub_C, Id_S, R'', T''_{exp}\}$

- 6) S verifies that the capability is indeed addressed to S.
- 7) S verifies that the capability has not yet expired.

$S: t < T''_{exp}$

Note: (5.), (6.) and (7.) may be interchanged.

- 8) S generates a random symmetric session key K_{sess} and a session identifier Sid. For practical reasons, K_{sess} may actually be a 4-tuple with different keys for encryption and integrity protection in each direction.

- 9) S encrypts the session key, Sid, and a nonce with the public key of the client obtained from the capability signed by AA. The nonce need not be random. It may even be a long counter or timestamp. Note: In practice a symmetric key K' may be used for encrypting K , N and only this key K' will be encrypted using pub_C .

$C \leftarrow S: \{K_{sess}, N_1, Sid\}pub_C$

- 10) C decrypts the session key using its private key.

$C: K_{sess}, N_1, Sid = \{\{K_{sess}, N_1, Sid\}pub_C\}priv_C$

- 4) *Data exchange session:*

- 11) C sends data/commands to S. The shared symmetric session key may be used for encryption or integrity protection using e.g. a MAC/keyed hash. Via the nonce S can verify that this is an expected, not injected or replayed, message.

$C \rightarrow S: \{Id_C, Sid, \{N_1, data\}_{K_{sess}}\}$ (encrypted)

or

$C \rightarrow S: \{Id_C, Sid, \{N_1, data\}, MAC(N_1, data, K_{sess})\}$
(integrity protected)

- 12) S verifies that capability still has not expired and that the nonce is correct (as expected).

$S: t < T''_{exp}; nonce = N_1$

- 13) S verifies that the request/command sent by the client is authorized by the presented capability.

- 14) S returns data to client with a new nonce. Again, the shared symmetric session key may be used for encryption or integrity protection. The nonce may be omitted if no further client message is expected.

$C \leftarrow S: \{Id_S, Sid, \{N_2, data\}_{K_{sess}}\}$

or

$C \leftarrow S: \{Id_S, Sid, \{N_2, data\}, MAC(N_2, data, K_{sess})\}$

B. Explanations and considerations

This section explains a number of design considerations for the previously described protocol.

a) *Configuration scalability:* By configuring the same public key of the access authority AA in each server no server specific key management is necessary. Multiple AA keys could be installed in each server to allow parallel use of multiple access authorities (increased availability) or for partitioning the access rights to certain access authorities. This may be used e.g. in a scenario where a diagnosis service provider is allowed to run its own access authority to access the servers of a customer, but this access authority is limited on the server side to read permissions only.

b) *Protection against protocol compromise via server compromise:* The servers store no secrets, which could be used to compromise the protocol. Note that this implies that there is, like with conventional password-based authentication and access control, no authentication of the server to the client. This is a trade-off between security and manageability. In consequence, an attacker could spoof a server towards a client. However, this attack would only allow denial of service and the injection of wrong data into the client, while it can not be used to inject malicious data or commands into the server.

c) *Quick roll-out of authorization changes:* Client access rights to target servers (the access matrix) are maintained in one (or a small number of) central servers under control of the access granting organization. Role based access control may be used on this central server for scalable client rights assignment. Thus, any change can be rolled out by means of a single change action and will be effective after at most the longest expiration time for a capability issued before the roll-out.

d) *Control of the access authority server:* The access authority server may be managed by the organization owning the embedded servers or its operation may be outsourced to a trusted service provider, e.g. an automation system vendor. Combined schemes with two chained (AND) or parallel (OR) authorization requests are possible.

e) *Offline use:* In some situations a physical presence at the embedded server is necessary for maintenance actions and no network connectivity to any outside service is available.

For this scenario, the service technician acquires a capability with a suitable expiration period from the access authorization server before traveling to the embedded server location. The capability contains the access rights which it authorizes and thus can be evaluated offline by the target server. Appropriate, limited capabilities, e.g. for shutdown, might also be stored onsite for emergency situations.

f) Limit of validity of a capability: Knowledge-based authentication and access control rights (passwords) can not be revoked without reconfiguring all servers. Other existing revocation schemes do not work with offline servers. The proposed protocol addresses this issue by using single purpose capabilities with short expiration periods (minutes/hours), for online remote access, and days, if a physical visit to the target server is necessary. Note that expiration-based revocation requires some degree of time synchronization between the access authorization server and the embedded servers.

g) Auditability: Each client accessing the target server shall be unambiguously individually identified. For this purpose the capability contains a client identifier Id_C , which may be logged by the accessed embedded server.

h) Prevention of client spoofing via copied capabilities: The capability signed by the access authority (AA) contains the public key of the client. The client has to demonstrate that it has access to the corresponding private key $priv_C$ in order to successfully participate in the protocol (decrypt the session key).

i) Ensuring that a capability can only be used for the intended server: Server identifier Id_S in the capability allows the server to verify that this capability has been issued for access to itself. Note that this requires some kind of unique identifier for each server, such as a device name or communication address, that can not be changed without be subject to access control. Of course, this opens the scheme to a potential privilege escalation attempt, if the permissions needed for changing the server identity do not constitute the highest level of access rights for the server.

j) Computational efficiency: The established session uses symmetric cryptography. Either encryption (more computationally intensive) or integrity protection only are possible.

k) Replay prevention: Each message from the server to the client contains a nonce that has to be included in the next message from the client to the server. The server keeps track of nonces without reply. Any message with an unexpected (e.g. old) nonce is rejected. Note that a lost message (loss of synchronization) may break communication. A window scheme may be used to partially mitigate this problem with a small increase of the risk of replay attacks.

IV. CONCLUSION

This paper presents a novel authentication and access control scheme for embedded servers like automation systems and network devices. This protocol meets an urgent need in industry because it significantly improves various security properties compared to conventional password-based schemes today in use for those systems, while at the same time improving

scalability and manageability. Wide use of this scheme in an electric utility may e.g. be a key component for meeting NERC CIP requirements on roll-out time limits for access revocation (CIP 004-1/B/R4.2, [19]). Future work will be to select the actual cryptographic parameters such as algorithms, modes, and key length for symmetric and asymmetric components, to implement a prototype and to obtain data on performance for various relevant execution platforms, as well as to formally prove the correctness of the proposed protocol.

REFERENCES

- [1] IEC, "IEC61850 communication networks and systems in substations," International Standard, IEC61850, Part 1 to 10, 2003.
- [2] B. Schneier, *Applied Cryptography*, 2nd ed. Wiley, 1996.
- [3] N. Ferguson and B. Schneier, *Practical Cryptography*. Wiley, 2003.
- [4] W. Mao, *Modern Cryptography: Theory and Practice*. Prentice-Hall, 2003.
- [5] J. Kohl and C. Neuman, "The Kerberos network authentication service (V5)," RFC 1510, September 1993.
- [6] B. Neumann and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, September 1994.
- [7] M. Sirbu and J. Chuang, "Distributed authentication in kerberos using public key cryptography," in *Proceedings of the Internet Society Symposium on Network and Distributed System Security NDSS'97*, 1997.
- [8] A. Medvinsky, M. Hur, S. Medvinsky, and C. Neuman, "Public key utilizing tickets for application servers (pkapp)," draft-ietf-cat-kerberos-pk-tapp-04.txt, 2000.
- [9] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen, "Simple Public Key Certificate," draft-ietf-spki-cert-structure-06.txt, July 1999.
- [10] C. M. Ellison, "SPKI Requirements," RFC 2692, September 1999.
- [11] J. Hughes and E. Maler, "Security Assertion Markup Language (SAML) V2.0 Technical Overview," sstc-saml-tech-overview-2.0-draft-08, September 2005.
- [12] ITU, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks," ITU Recommendation X.509, March 2000.
- [13] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture," RFC 4251, January 2006.
- [14] H. Chan, V. Gligor, A. Perrig, and G. Muralidharan, "On the distribution and revocation of cryptographic keys in sensor networks," *IEEE Transactions on dependable and secure computing*, vol. 2, no. 3, pp. 233–247, July/September 2005.
- [15] D. Dietrich, T. Sauter, and A. Treytl, "IT-Sicherheit für Feldbus-Netzwerke," in *Proc. of VDE Kongress*, October 2004.
- [16] F. J. Dafelmair, "Improvements in process control dependability through Internet security technology," in *Proceedings Safecom 2000*, ser. LNCS, vol. 1943. Springer, 2000, pp. 321–332.
- [17] T. von Hoff and M. Crevatin, "HTTP digest authentication in embedded automation systems," in *Proc. IEEE Int. Conf. on Emerging Technologies for Factory Automation (ETFA'03)*, vol. 1, 2003, pp. 390–397.
- [18] Y. Xu, R. Song, L. Korba, L. Wang, W. Shen, and S. Lang, "Distributed device networks with security constraints," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 4, pp. 217–225, November 2005.
- [19] NERC, "Standard CIP-002-1 to 009-1 - Cyber Security, Draft 4," January 2006.