European Virtual Smart Grid Lab

SES 11814, Deliverable A1300-AD1305

ICT security for m2m signaling between prosumers in a micro-grid

Per Ljungberg, Ericsson AB, per.ljungberg@ericsson.com
Tobias Söderlund, Ericsson AB, tobias.soderlund@ericsson.com
Stefan Hagbard, Exformation AB, stefan.hagbard@exformation.com

1 Introduction

The security as server solution that is used in the EVSGL project is used by service providers to enable users in a Smart Grid to get access and ability to control devices and services.

This security solution protects the communication as well as validates and authenticates users and devices within the smart grid use cases. It is delivered as a service, Security as a Service, SECaaS, that is presented as a service enablement functionality.

The solution makes it possible for service providers and application developers to achieve the required security level through a service purchase instead of developing it. Hence simplify and speed up the service development and move cost for capex to opex.

It is possible to set different security levels which gives the service flexibility to meet security requirements from a wide range of service.

The main benefits with the SECaaS are:

- Service providers and developers speeds up development and investment for implement security to their service offering.
- The number of security relations between users, service providers and devices can be decreased without affecting the security level.

2 Solution overview

The security as a service set up consists of four domains:

- A **security server** that is part of the service enablement functionality.
- A **service provider** that runs an M2M service communicating with a number of connected devices.

  Having access to a trustworthy security solution is crucial for enterprises in order to protect business critical operations and data. The server part of service that the service provider runs is in this case represented by a web service.
- A number of **connected (M2M enabled) devices** that the service provider controls and or is collecting data from. The connected device contains a device controller that the security server communicates with.
- **End user devices** used by users interacting with smart grid application provide by the service provider. The end user device contains a browser and a security application.

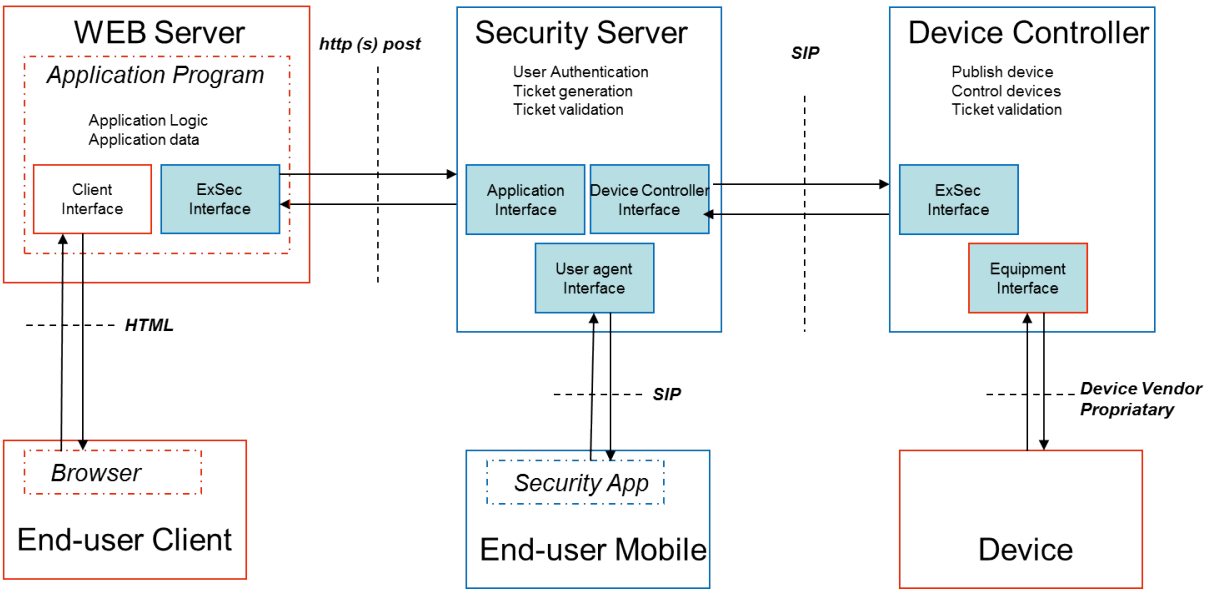

Figure 1: Solution overview

3 Proof-of-concept description

The proof of concept implements was implemented and tested during the project. It provides security to a service provider running a smart grid application. The functionality can be summarized as:

- Secure web login based on digital signature generated in mobile
- Session management of logged in users
- Policy enforcement of user rights depending on user role
- Session management of attached devices
- Electronic ticket generation and validation between device controller and security server

The proof of concept uses credentials on device controller, mobile phone and in the security server. The credential storage is done using software as well as a smart card to show different implementation strategies in devices and mobile applications. In a real life application either implementation strategy could be used or a combination of them.

4 Use Case

The Security as a service solution provides a secure link between the domains that is part of the solution that it is serving to protect. It does also provide means to validate users and devices that interacting with the service run by the service provider.

It is implemented and proven in two different set of services.

One is when an end user initiates charging of an electrical vehicle. In this example is the identity of charging pole and the end user secured. The other example is where a smart grid application provides end user the possibility to control and monitor a set of appliances.

The description below explains the procedure.

4.1 Start using security as a service

A service provider that want to make use of the Security as a Service functionality have to make a business agreement with the actor that runs the service. This agreement gives the service provider a unique digital service identity including private keys and server public key.

When this is done will the service provider have access to a dedicated database for easy integration with WEB applications to request security services and to retain results. This hides the complexity of proprietary security implementations between clients and servers and provides a well-known development environment for application developers.
As a complement or alternative to the database access, a HTTP POST interface can be used to access the services.

It is the service provider that decides what security level that is needed. The SECaaS server will enforce the security requirements for the incoming requests. It is possible to set different security requirements for different features in the application.

The service provider defines what users are allowed for his/her application in the security server. All users are associated with a role when being provisioned. The roles are used to provide different capabilities for different user groups

Each service has its own set of keys to ensure separation between services while still being able to use the same end-user keys.

4.2 End user validation

Digital identities of the end users are registered in the security server that is the central part of the SECaaS solution.

The digital identity can be requested by any server connected the Security as a Service. Examples of that are when a user wants to perform an action on a web page or in an application, the application then asks the SECaaS to generate a digital signature of the request. This request can be a challenge/response, an authentication, a transaction or any other action represented in digital form that can be signed by the user.

The transaction is shown for the user prior to signing by presenting a user signature. The signed transaction is automatically sent back to the security server. Who validates the signature and forwards the result to the application that made the original request.

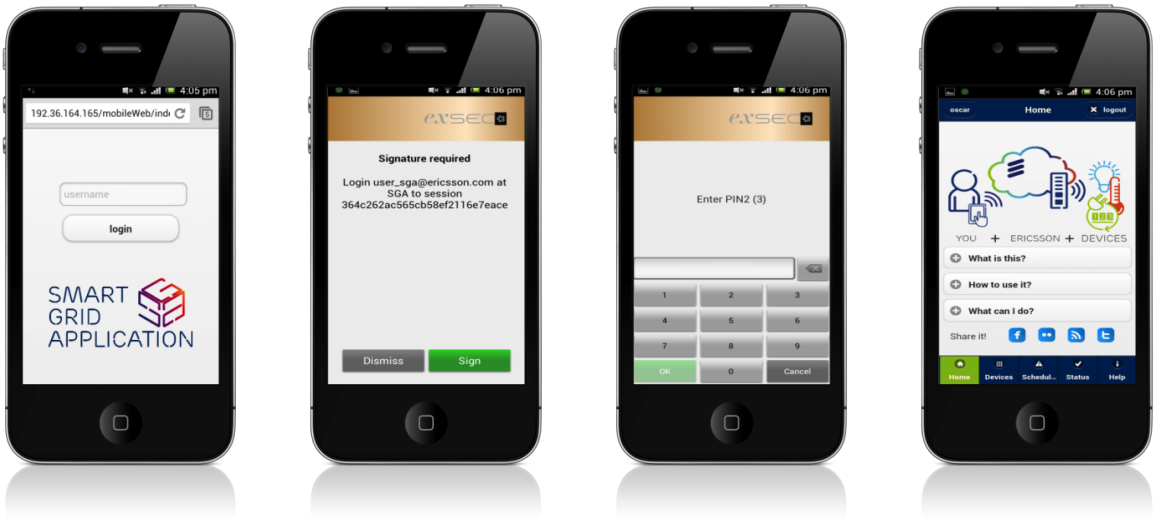

Figure 2: End user validation with mobile device

4.3 Connect a device a server

Devices that are to be controlled or be part of a communication of a service protected by the SECaaS need to be published to the service that it is connected to. It is the device controller in the device that is published. This is done when a new devices is installed or is replaced.

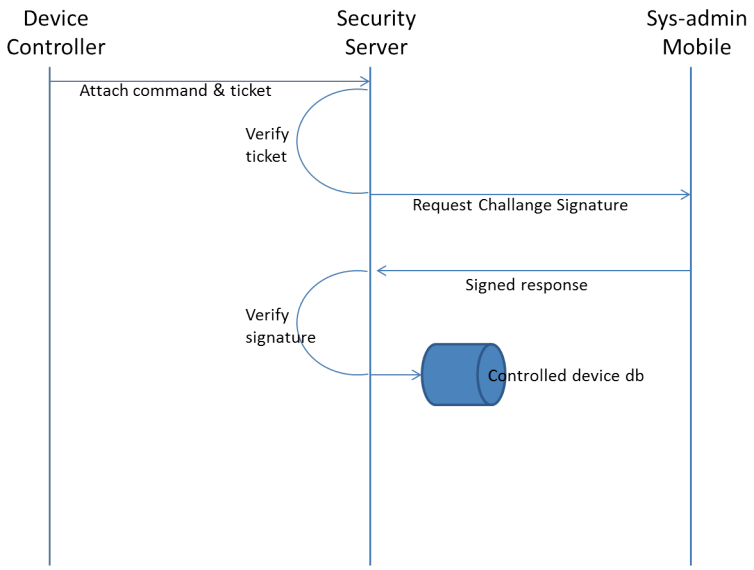

Figure 3: Connection of a device to a server

4.4 Security validation through electronic tickets and end user interaction

To provide access to an application the service provider has request a session start. This session start generates an electronic ticket to each associated servers/resources that tells them that this is an authenticated session and also the privileges of the user.

It is done each time a user logs into a service, wants to control a device or get access to a certain web page.

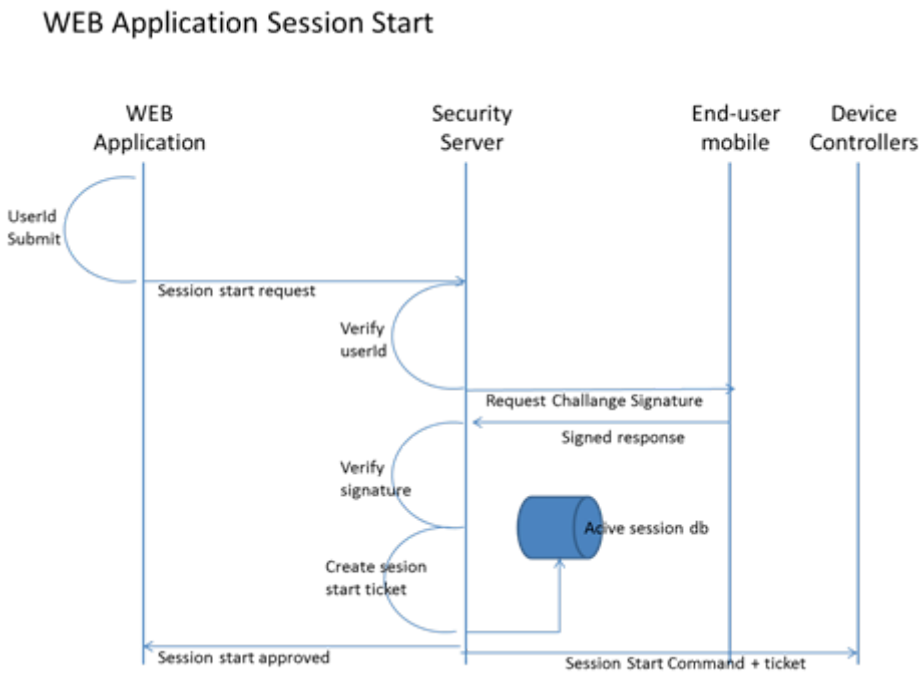

Figure 4: Session start of Web application

The Security as a Service will support the application, not only by validation of the user but also in the validation of a security profile for the devices exposed in the application. This includes validation of user privileges compared to the device requested privileges and that no device control commands are sent to the device unless a valid session is at hand. The server will also inform the device controller of the valid session by issuing an electronic ticket. This ticket is used to make validation of each command very fast and avoid that the device controller has to know about end-users, it is enough to be able to validate the ticket and hence trust the command based on the validation of end-user performed by the security server.

The application can provide any number of actions during the duration of the session and, when user logs out; send a session stop to the server. Commands to control devices or associated resources will always pass through the security server interface to ensure security policy enforcement. When session stop is sent, it will disable the current user session and inform the associated servers that the session now is invalid and inhibit any new commands to be sent to the device.

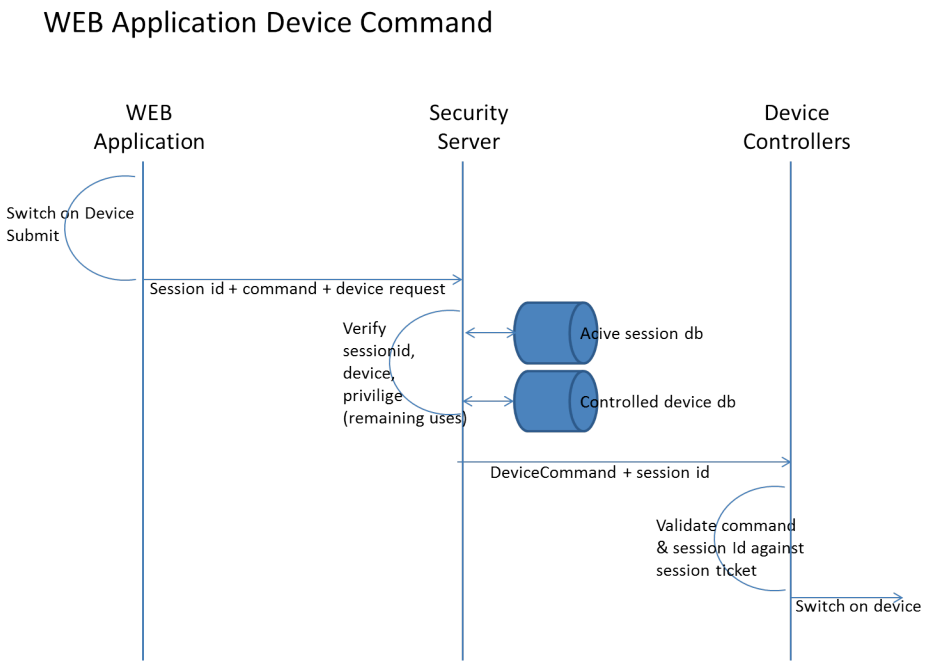

Figure 5: Processing of the device command

The application, or the resources associated with the application, may or may not add additional security requests for individual features by requesting them from the security server.

The service provider could also be someone exposing a device or an on-line service that is used in other applications. It could for example be an optimization algorithm controlling the smart grid or a physical device attached to the grid or any other digital reachable feature such as sensors or actuators.

To make such resource available to other services and applications, the service provider will use an attach command to publish the resource to other service providers using the Security as a Service. In a similar way, a detach command is used to remove the devices from the other service providers applications.

To enforce the access security of connected devices, each attach command also states the requirements on user privileges required. Each device controller may also implement its own policy and add any security validation on top of the standard service.

5 Security as a Service Architecture

The security solution connects end-user devices (mobiles), application service providers (web applications) and device controllers implemented in M2M devices (e.g.,EV-charging controller, home gateways) to the Security as a Service server.

The security server provides interfaces for

- Connect and disconnect devices to/from a web service
- Authenticate a user and link the user to a web session
- Configure the role for a user
- Request a device to switch on or off during a web session
- Provide a limited number of device service to a user

Depending on where and how each web service is implemented, the command interface may be used over http POST (GET), SIP messaging or SQL.

Each device or end-user client has to store some sensitive keys used to identify the user, device or server it communicates with. This is called credential storage. The credential storage can either be a smart card, a secure element or secure execution mode in the CPU or a software implementation. The choice is depending on the capabilities of the device as well as the risk and business value at stake.

The Security as a Service uses two of those in its proof of concept; smart card and software implementation of credential storage.

The mobile client uses smart card based security when it communicate with the security server and the smart card supports a number of pre-defined operations such as

- Digital signing of challenge/response message
- Validation of signatures (not used in this proof of concept)
- Message encryption (not used in this proof of concept)
- Plain text messages (not used in this proof of concept)
- Secure multimedia services (not used in this proof of concept)

The security card is a smart card in the micro-SD form factor containing certificates and operations to support PKI services and key-stream generation for multimedia services.

Each security card has a unique ICCID that also serves as the logical SIP address to the device. This allows for device portability between mobiles, pc and devices without any need of reconfiguration of the security for the user. The credentials are

always reached by the correct ICCID independent of what device is currently used. The SIP communication can be changed to any other HTTP based addressing scheme.

The security server generates a one-time challenge when a user requests a login. The challenge is sent to the mobile (i.e. security card) and the challenge is signed by users' private key stored in the smart card. The access to the smart c+ard is protected by a unique signature PIN.

The mobile security application is activated by a SIP message that in turn triggers a push notification to alert the user. The security application can run in parallel with e.g. a browser application to add security to other services in a generalized way.

The device controller, controlling the connected devices, may either be equipped with security card or software storage of the credentials in an embedded module. In any case the device controller may be identified by a unique SIP identity or any other http protocol provided by connectivity servers. The credential storage can utilize any form of hardware support including smart card, processor security element, processor secure execution modes or software implementation depending on the available features of the particular devices controller. The security level is of course different depending on the support where a smart card alternative is considered the most secure. Still the security level should be chosen based on the commercial risks at stake for each application and in many cases a software solution may well prove to be the best trade-off between threats, cost and business value.

The device controller provides a function to attach or detach the devices to/from a web service controlled by the security server. Each attach or detach command will require the appointed administrator to approve the action by signing the request in the same way as for a login request. The device controller may also define own roles for access restrictions. In this proof of concept we have assigned the user and peruse roles as a requirement to control a device connected to the device controller. The peruse is an example of limitation of the number of actions a user is allowed, in this case the peruse user will automatically get 3 attempts to switch on devices and after that be requested to reload his/her account to again be able to control a device. In a real life deployment, this translates into pre-paid services or caps on usage per time.

The communication between device controller and the actual device is proprietary for each device vendor and not part of this security architecture.

It is of course also possible to change the SIP communication to a plain http interface for communication with the device controller. The SIP is mainly used to overcome NAT network addressing issues in IP networks without having persistent HTTP allocated for all connected devices.

The security server consist of

- Interfaces to external devices/applications
- Policy enforcement
- PKI and validation

The interface module provides SIP, HTTP POST and in some cases SQL interfaces to service providers' applications, users and device controlling applications. The interface simply manages to receive and send messages in a format suitable for the sender/receiver while maintaining an internal format for processing. The external messaging formats are mainly based on HTTP request formats for query strings.

Policy enforcement verifies the user role as determined at login to the web service with the device or service privilege required for a command. If the user role is equal or greater than the required privilege, the command is forwarded to the device controller. The device controller may or may not choose to validate the request e.g. by validate the session key or deploy additional signature requests. In the proof of concept, the device controller will validate the session and user privilege based on the session ticket.

PKI validation is made by verifying any signature provided by devices or users. User signatures are validated using ECC certificates while device signatures in this example are validated by RSA certificates. Validation of signatures are made

- At session start, i.e. login to web service
- At attach or detach of device controller
- On request by local security policy in service provider application(s)

The basic web application provided to illustrate the use of Security as a Service is implemented as a PHP/Javascript AJAX web application using SQL and HTTP POST interface towards the security server.

The security server interface is used to hide the SIP communication from the application layer.

A system overview is shown in the figure below.

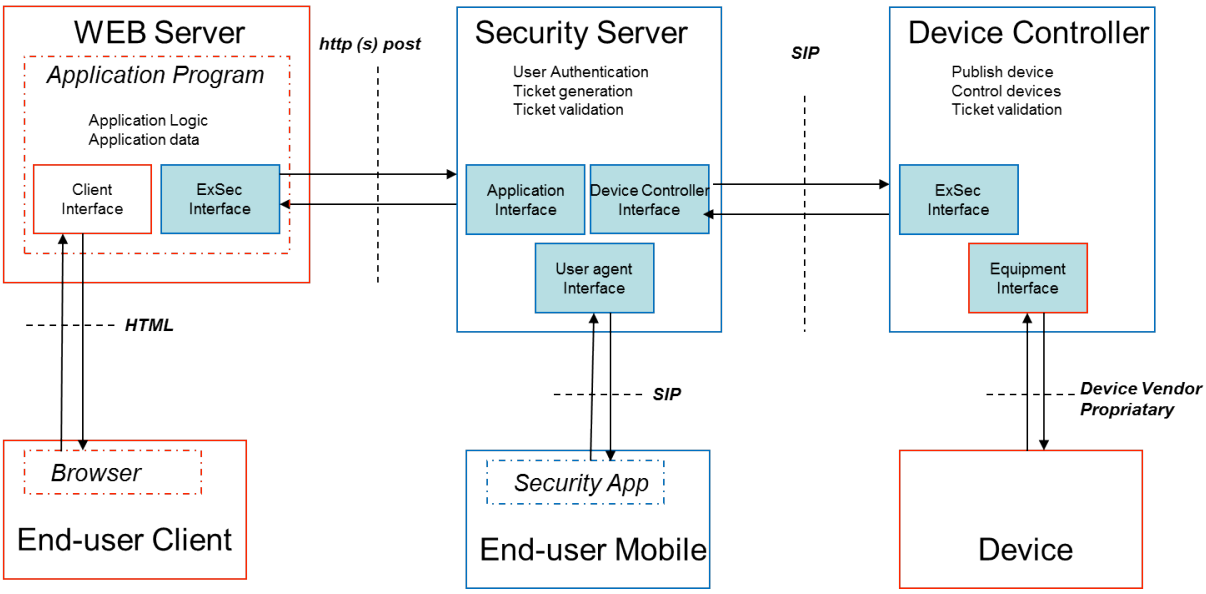

Figure 6: System overview

A node in red indicates that they are part of the end-to-end flow however not directly part of the Security as a Service.

One advantage with the Security as a Server use of credentials is that it can assign a credential to any user or device identity such as a mail address or a device ICCID. This would not be possible if for example a PKI based on national e-id was used. On the other hand the Security as a Server could be used as a front end to a national e-id to validate end-user if that is needed in an application.