Protecting Remote Component Authentication

Rainer Falk, Steffen Fries
Corporate Technology
Siemens AG
D-81739 Munich, Germany
e-mail: {rainer.falk|steffen.fries}@siemens.com

***Abstract*—Component authentication allows verifying the genuineness of various components being part of a machine or a system or connected to control equipment. Various technologies are used, ranging from holograms, hidden marks, special inks to cryptography-based component authentication. Typical cryptography-based mechanisms employ a challenge-response-based component authentication mechanism. These mechanisms have been designed originally for a local verification, i.e., for an authentication performed in direct vicinity of the product to be verified. This paper describes an attack on challenge-response component authentication when supporting a remote component authentication and describes a new security measure to prevent this attack.**

Keywords-device authentication, counterfeiting, tunneled authentication

I. INTRODUCTION

Authentication is an elementary security service proving that an entity in fact possesses a claimed identity. Often natural persons are authenticated. The basic approaches a person can use to prove a claimed identity are by something the person knows (e.g., a password), by showing something the person has (e.g., passport, authentication token, smart card), or by exposing a physical property the person has (biometric property, e.g., a fingerprint, voice, iris, or behavior). Considering the threat of counterfeited products (e.g., consumables, replacement parts) and the increasing importance of ubiquitous machine-based communication, also physical objects need to be authenticated in a secure way. Various different technologies are used to verify the authenticity of products, e.g., applying visible and hidden markers, using security labels (using e.g., security ink or holograms), and by integrating cryptographic authentication functionality (wired product authentication token or RFID (Radio Frequency IDentification) authentication tag).

One important driver for verifying the authenticity of products is safety. For example, counterfeited electrical components as switches or fuses can cause physical damage when they do not fulfill electrical safety norms (e.g., by causing electric shock to humans or a fire). Other examples are provided through electric safety devices as e.g., overvoltage protecting devices or an earth leakage circuit breaker which do not fulfill the expected functionality.

Focus of this paper is a challenge-response authentication of a physical object, e.g., an electric component. This authentication technology has the advantage that a control or supervisory equipment can automatically verify the authenticity of installed components. Local object authentication is used widely e.g., for authenticating battery packs or printer consumables (toner / ink cartridges). Here, cryptographic challenge-response based authentication is applied. Highly cost-optimized solutions are available commercially that allow to use these technologies also in low-cost devices. The authentication protocol is, however, not being designed to protect against man-in-the-middle attacks as these are not relevant in the targeted use case.

Section II gives an overview on challenge-response based object authentication. The scenario for remote object authentication is described in Section III as well as typical technical solutions, and their susceptibility to man-in-the-middle attacks when used for remote component authentication by different verifiers. A new, simple to implement countermeasure protecting against man-in-the-middle attacks is described in Section IV. It enables the re-using of highly cost-optimized object authentication also for remote object authentication, i.e., for a usage scenario not being designed for. This enables to use extremely simple and therefore cost-efficient hardware-based device security mechanisms for purposes not being intended for originally. It can be applied in particular even in those cases when the verifier needs access to the unmodified response value. The application to IP-based smart objects is described in Section VI, providing a highly optimized basis for a secure device identity within the Internet of Things. Related work is summarized in Section VII, before giving a summary and outlook in Section VIII.

II. COMPONENT AUTHENTICATION

A. Overview

Components of a machine (internal or attached) shall be identified securely. This requirements is known for components like ink cartridges, batteries. In industrial machines it applies to replacement parts, sensors, actor devices. Authentication of a device allows a reliable identification of original products.

For authentication a challenge value is sent to the object to be authenticated. A corresponding response value is sent back and verified. The response can be calculated using a cryptographic authentication mechanism or by using a physically unclonable function (PUF). As only an original product can determine the correct response value corresponding to a challenge, the product entity or a dedicated part of the product is thereby authenticated.

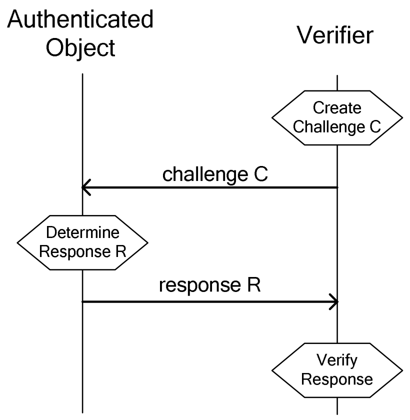

Figure 1. Challenge-Response Object Authentication

Figure 1 shows the schematic data flow between a verifier and an object to be authenticated. The verifier sends a random challenge to the product that determines and sends back the corresponding response. The verifier checks the response. Depending on the result, the product is accepted as authenticated or it is rejected.

B. *Implementation Examples*

Various cryptographic mechanisms can be used to realize a challenge-response product authentication. Basically, symmetric cryptography, asymmetric cryptography, or physically unclonable functions can be used. While in the case of symmetric cryptography also the verifier is in possession of the cryptographic device key and can therefore calculate the expected response value, in case of asymmetric cryptography or PUFs the verifier might not be in a position to calculate the correct response. He has only the option to verify the received response.

This has consequences on whether it is possible to include binding parameters in the response, i.e. to calculate a derived response value. In the symmetric case, the verifier can calculate the expected response and perform the expected response modifications and compare this obtained expected result with the actually received result. However, in the PUF and asymmetric case, this is not possible as the verifier can perform only a verification operation on the received result, but cannot determine a valid response on its own. The verifier needs therefore access to the original, unmodified response value to perform the verification operation. It is not possible to use a derived response value, e.g., by using a keyed hash function or a key derivation function that uses freely definable binding parameters during the response derivation.

Note that the application of symmetric methods may also imply a higher administrative overhead, as the necessary symmetric shared key needs to be securely available on both sides, the product and the verifier.

Implementation examples of product authentication are summarized in the following, describing one example of each category.

- Atmel CryptoAuthentication [1], [2]: A symmetric-key based authentication is performed, intended for example for authenticating battery packs. A challenge-response authentication based on the SHA256 hash algorithm is implemented to compute a keyed digest for the provided challenge value. The input parameter to the SHA256 algorithm is the concatenation of the secret key, the challenge value, and optionally other chip-specific data (serial number, fuses). The challenge is an arbitrary 256 bit value selected by the verifier.
- Infineon ORIGA [3]: An asymmetric authentication based on elliptic curves is performed. A cryptographic operation is performed by the product to be authenticated using the product’s private key. The verifier checks the received response using the corresponding public key by performing a cryptographic verification operation on the received value. The verifier does not need access to the products private key.
- Verayo RFID Tag “Vera M4H” [4]: An integrated circuit comprising a physically unclonable function is used to determine a response value depending on the challenge value and hardly to reproduce physical characteristics of the product to be authenticated. Therefore, no cryptographic key has to be stored on the RFID tag as a physical fingerprint of the RFID tag is employed.

C. *Applications / Use Cases*

A reliably identification of products is needed in various use cases. For safety reasons, components can be verified to ensure that no counterfeited products are installed. Also an unverifiable product may be used with conservative operating conditions (e.g., maximum charging current of battery pack) to prevent damage. Another example is authenticated setup of a protected communication session for field level device communication.

III. Remote Component Authentication

One important class of use cases is remote component authentication. A machine equipped with or connected to several field devices (sensors, actuators) performs not only a local authentication, but supports a remote authentication of components by a supervisory system.

A. *Use Case Description*

In a remote object authentication, the verifier is remote to the object to be authenticated. The challenge and response values are encoded in messages that are transported over a communication network, e.g., an IP-based network, see Figure 2.

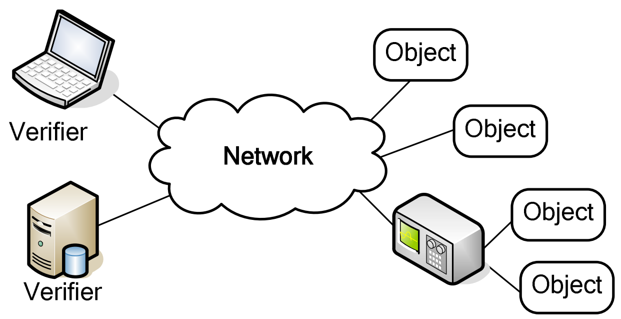

Figure 2. Remote Object Authenticationn

A verifier may be a service technician performing remote maintenance, or an automatic device tracking server or an inventory management server, see Figure 2. The objects to be authenticated may be connected directly to the network, or they may be attached to an intermediary device, e.g. a programmable logic controller.

The challenge response authentication operation is performed by the authenticated object as described above. However, the verifier is not in close vicinity to the object to be authenticated. In some cases, a protected communication channel may be used between the verifier and the intermediary, e.g., IPsec or SSL/TLS.

B. Man-in-the-Middle Suceptibility

In the case of remote object authentication, a (malicious) remote verifier may act as a man-in-the-middle attacker, see Figure 3.

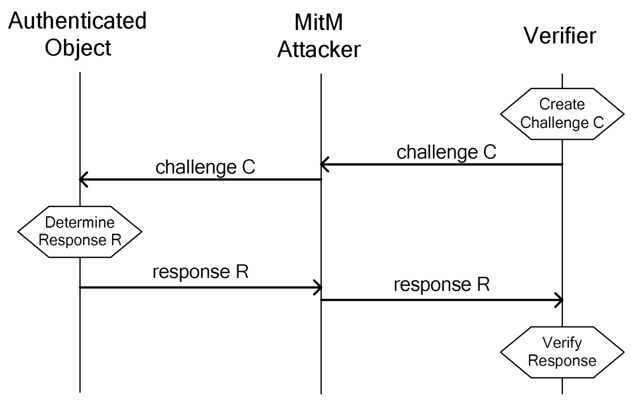

Figure 3. MitM Attack on Object Authentication

An attacking node as “man-in-the-middle” forwards unchanged messages towards and from the object. The verifier having authenticated the object as genuine assumes that it is communicating in fact with the device in the following data exchange. An attacker can use an arbitrary object as oracle that provides valid responses for freely chosen challenges. In consequence, any remote entity that has access to the object authentication functionality can act as a man-in-the-middle and may authenticate itself as the authenticated object if it has a sufficient number of challenge and response pairs. Hence, the object authentication can be manipulated.

When furthermore the authentication response is used for deriving cryptographic session keys, these keys can be derived by an attacker as well.

The fact that such a simple challenge-response authentication is prone to man-in-the-middle attacks is well known and documented also in the corresponding product documentation. For example, the man-in-the-middle attack is mentioned in [5]. In the considered usage environment where authenticated object and verifier are in direct physical connection, the attacker needs both a direct physical access to the attacked object and measurement equipment like e.g., a logic analyzer to analyze the information exchanged between the components. This increases the overall effort of the attack.

IV. Example for Cryptographic Binding Requirements

This section motivates the need for cryptographic binding by describing a known weakness. Transport Layer Security (TLS) is a very popular security protocol, which is used to protect web transactions in applications like online banking, to protect the mail communication via IMAP, to realize VPNs or for remote administration. Meanwhile the protocol is available as standard in version 1.2 as RFC 5246 [6].

Early November 2009, a vulnerability has been discovered allowing an attacker to inject data into a TLS connection without being noticed by the client. Such attacks were facilitated by a protocol weakness concerning renegotiation of security parameters. Renegotiation is a TLS feature to exchange fresh security parameter for an existing session. The problem arose due to the missing cryptographic binding between the initially negotiated security parameters and the new parameter set resulting from the renegotiation process. This can be exploited by an attacker as man-in-the-middle attack. A potential attack – a request to a web server – is described in the following, see Figure 4. :

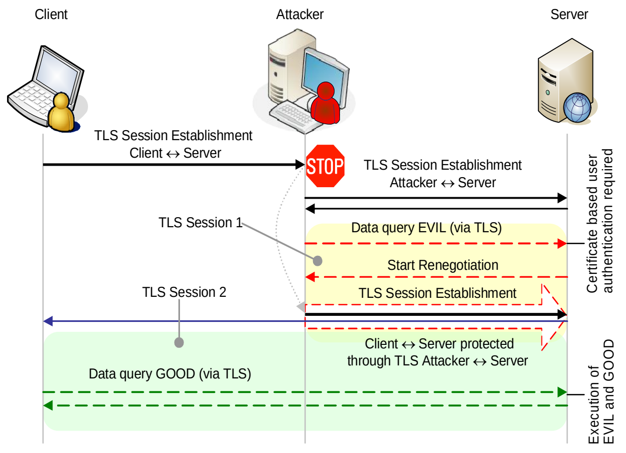
Client
Attacker
Server
TLS Session Establishment Client ↔ Server
STOP
TLS Session Establishment Attacker ↔ Server
TLS Session 1
Data query EVIL (via TLS)
Start Renegotiation
TLS Session 2
TLS Session Establishment
Client ↔ Server protected through TLS Attacker ↔ Server
Data query GOOD (via TLS)
Certificate based user authentication required
Execution of EVIL and GOOD

Figure 4. MitM Attack on Object Authentication

A potential attacker controlling the data path is waiting for a connection attempt by a client. As soon as the client establishes a TLS connection to the server, the attacker delays the client request. Now, the attacker establishes his own TLS connection to the server which is most often server-side authenticated. As soon as the TLS connection is established, the attacker sends a request *EVIL*, requiring the certificate based authentication of the client. This authentication is now being invoked by the server through starting the renegotiation. The *EVIL*-request, which is not authenticated at this time, is stored and executed after successful client authentication. Web servers are typically configured to request client authentication, e.g., when data from an access protected directory shall be read.

The attacker now sends the delayed *Client request* to the server which interprets this message as part of the renegotiation phase, over the TLS protected link between the attacker and the server. This enables an end-to-end key negotiation between the client and the server. All subsequent messages are now secured and the attacker is not able to access them. But, the stored *EVIL*-request was submitted and is executed by the server.

This attack showed on the one hand a potential weakness of TLS due to the missing cryptographic session binding, on the other hand it shows the insufficient integration of TLS into the application, as the web server in this example should have requested an affirmation of the *EVIL* request over the renegotiated TLS session before executing it. This weakness could be exploited for instance for stealing passwords or cookies from Web applications. The weakness has been addressed as part of an update of the TLS protocol using a binding of the initial session to the renegotiated session [7].

V. CHALLENGE BINDING (PRE-CHALLENGE)

The problem originates from the fact that the same authentication mechanism resp. the same authentication key is used in different contexts. Following common security design different keys would be used for different purposes, and to bind the cryptographic material to the intended context (i.e., to derive context-bound session keys from the response).

As in important implementations of component authentication the verifier needs access to the unmodified response, the response value cannot be modified. Therefore, challenge binding is proposed as countermeasure: When a remote verifier cannot select the challenge value, it cannot use the authenticating object as oracle to determine responses for arbitrary challenge values.

A. Challende Binding

The challenge selected by a verifier is bound to the verifier context. This binding operation can be performed by the authenticated object itself or by a (trusted) intermediary node, see Figure 5.

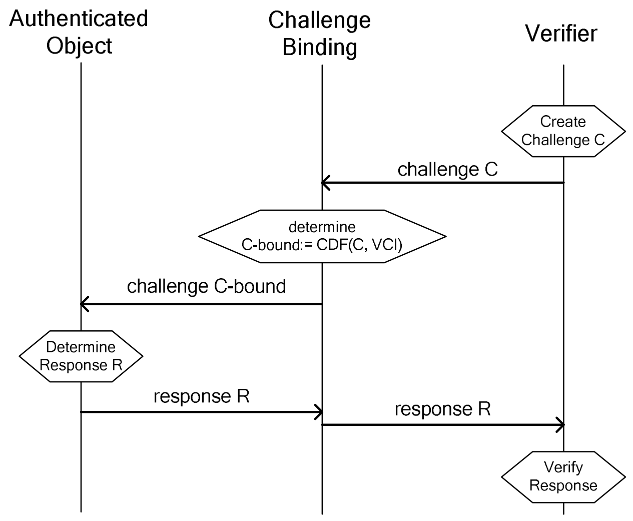

Figure 5. Challenge Binding

The challenge C selected by the verifier is sent to the object directly or to an intermediary node in close vicinity of the object to be authenticated (e.g., a control unit to which a sensor or actuator is directly connected), see Figure 5. This challenge is modified by deriving a bound challenge value C-bound using a non-invertible function (challenge derivation function, CDF). Verifier-dependent context information (VCI) is used as derivation parameter to bind the challenge to the respective verifier. In particular, the verifier's network address, node identifier, or a session key established between the verifier and the intermediary can be used. The challenge derivation can be performed by both, the object to be authenticated itself, or by a trusted intermediary node.

This modified, verifier-context bound challenge C-bound is forwarded to the object to be authenticated. The object determines the corresponding response and sends it back to the intermediary that forwards the response to the (remote) verifier. The verifier determines the bound challenge C-bound as well, using the selected challenge C and the verifier context information VCI. Note that the VCI can be determined either by the verifier and the intermediary, if both are configured in a way to determine the VCI on available information (like certain address information of the verifier, see also section B below). Alternatively, the VCI may be sent as part of the communication from the intermediary or the verifier.

The remote verifying party can therefore not freely select the challenge for which a response is computed. Anyhow, it can be sure about the freshness of the challenge C-bound for which it received the response as it depends on the pre-challenge C selected by the verifier.

B. Verifier Context Information

Verifier dependent context information is used as derivation parameter to bind the challenge to the respective verifier. There is a variety of parameters that can be used to specify a verifier context. In particular, the verifier identity, e.g., IP or MAC address, DNS name or URL, an (unpredictable) session ID, or a digital certificate or security assertion may be used. This information can be determined by the intermediary, without direct involvement of the verifier. This verifier context is used as parameter to separate two different verifiers. So it must not be possible in practice for a verifier to act successfully within a verification context belonging to a different verifier.

C. Challenge Derivation Function

Requirements on a challenge derivation function are similar as for a key derivation function, i.e. being non-invertible and pre-image resistant (see [7] and [8] for more specific information on key derivation functions). Therefore, the functions that are typically used for key derivation can be used as challenge derivation function as well. For example, the bound challenge C-bound could be derived as HMAC-SHA1(C, VCI), using the challenge instead of a key, and using VCI as textual string determining the verifier context. Alternative key derivation functions may be the higher SHA methods like SHA256 or SHA512 in combination with the HMAC or symmetric algorithms like the AES in CBC-MAC mode [9].

VI. Application to IP-based Smart Objects

One possible application of protected remote component identification is IP-based communication within the Internet of Things. A node communicating with a smart object ("thing") over IP-based communication wants to verify the identity of the smart object resp. of a component being part of or being integrated into the smart object. Communication can be realized e.g., using HTTP-based Web Service protected by TLS or by IP-based communication protected by IPsec. A challenge-response based smart object authentication can be integrated in well-know protected communication protocols, as HTTP Digest over unilaterally authenticated SSL/TLS, or EAP, or within IKEv2 for IPsec.

However, the challenge is modified using verifier context information as derivation parameter. Here, besides the nodes identifier (server name resp. IP address) also the used communication protocol can be used as challenge derivation parameter (e.g., "HTTP-DIGEST/TLS" || Server-IP).

VII. Related Work

Most similar to our proposal is the binding of an authentication challenge for a PUF authentication to the hash of the requesting program, see [10]: The verifier selects a pre-challenge, from which a bound challenge is derived using the hash of the verifier program as input to the challenge derivation. Note that the binding to the hash of the verifier program alone, without address information is weaker, as the hash is supposed to be the same on different hosts. Thus, an attacker possessing the verifier program may still perform the attacks described in section II.

The insecurity of tunneled authentication protocols has been analyzed [11]. In real-world environments, often an existing security deployment and authentication shall be re-used for a different purpose. In particular tunneled EAP authentication was considered, e.g., based on PEAP. The described countermeasure was binding cryptographically the results (session keys) of the two authentication runs, i.e., the inner and the outer authentication, or by binding the session key to an endpoint identifier.

Performing a key derivation is a basic building block for designing secure communication. Various required session keys can be derived from a common master session key. NIST recommended a key derivation function, using a usage-describing textual string as derivation parameter [7]. Another example is the pseudorandom function used within TLS [6], which uses secret keys, seeds and textual strings (identifying label) as input and produces an output of arbitrary length. The same approach is taken in the Multimedia Internet Keying MIKEY [12].

It is also known to bind an authentication to properties of the used communication channel [13]. Two end-points authenticate at one network layer and bind the result to channel properties to prevent against man-in-the-middle attacks where the attack would result in different channel binding properties from the viewpoint of the authenticating nodes.

Furthermore, non-interactive key agreement schemes allow to derive a common, shared key material between nodes that have received a key bound to the own identity [14]. No protocol exchange is required to derive this shared key, but the key is derived similar as with a key derivation function. However, the two derivation steps for binding a root key to two node identifiers can be performed commutatively.

VIII. Summary and Outlook

An attack on component authentication has been described where a single genuine component is used as oracle to compute valid authentication responses. A single malicious verifier may use an obtained valid response value to authenticate as the genuine component towards other verifiers. The described attack is made possible by the fact that the cryptographic solution for component authentication is used within a different usage environment than it has been designed for: The attack is relevant when the component authentication for verifying the genuineness of a component is performed not only locally, but also remotely. This attack is also an example that a small functional enhancement – here making an existing functionality accessible remotely – can have severe implications on security.

This paper proposed a challenge binding mechanism as countermeasure for the described attack. The available, extremely cost-efficient object authentication technology can thereby been used securely also for a different purpose than the one it has been designed for originally. An authentication challenge is bound to the verifier so that a remote verifier can neither simulate a local, unbound authentication nor can it simulate an authentication towards a different remote verifier having a different associated verification context. A possible application of this general challenge-binding mechanism is the cost-efficient authentication of components within the Internet of Things.

References

[1] Atmel CryptoAuthentication, http://www.atmel.com/products/cryptoauthentication/, last access March 2011

[2] Atmel CryptoAuthentication Product Uses, Application Note, 2009. http://www.atmel.com/dyn/resources/prod_documents/doc8663.pdf, last access March 2011

[3] Infineon ORIGA, http://www.infineon.com/ORIGA, last access March 2011

[4] Verayo http://www.verayo.com/product/pufrfid.html, last access March 2011

[5] Atmel CryptoAuthentication High level Security Models, Application note, 2009. http://www.atmel.com/dyn/resources/prod_documents/doc8666.pdf, last access March 2011

[6] T. Dierks and E. Rescorla: The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, August 2008, http://tools.ietf.org/html/rfc5246, last access March 2011

[7] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov: Transport Layer Security (TLS) Renegotiation Indication Extension, RFC 5746, February 2010, http://tools.ietf.org/html/rfc5746, last access March 2011

[8] Lily Chen: Recommendation for Key Derivation Using Pseudorandom Functions, NIST Special Publication 800-108, 2009. http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf, last access March 2011

[9] John Black and Philipp Rogaway: A Suggestion for Handling Arbitrary-Length Messages with the CBC MAC, http://www.cs.ucdavis.edu/~rogaway/papers/xcbc.pdf, last access March 2011

[10] Srini Devadas: Physical Unclonable Functions and Applications, Presentation Slides (online), http://people.csail.mit.edu/rudolph/Teaching/Lectures/Security/Lecture-Security-PUFs-2.pdf, last access March 2011

[11] N. Asokan, Valtteri Niemi, and Kaisa Nyberg: Man-in-the-Middle in Tunnelled Authentication Protocols, LNCS3364, Springer, 2005. http://www.saunalahti.fi/~asokan/research/tunnel_extab_final.pdf, last access March 2011

[12] J. Arkkom, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman: MIKEY: Multimedia Internet KEYing, RFC3830, August 2004, http://tools.ietf.org/html/rfc3830, last access March 2011

[13] N. Williams: On the Use of Channel Bindings to Secure Channels, Internet RFC5056, 2007. http://tools.ietf.org/rfc/rfc5056.txt, last access March 2011

[14] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen: Strongly-Resilient and Non-Interactive Hierarchical Key-Agreement in MANETs, Cryptology ePrint Archive, 2008. http://eprint.iacr.org/2008/308.pdf, last access March 2011