Advances in 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 originality of various components being part of a machine or a system, or being connected to control equipment. Various technologies are available, ranging from holograms, hidden marks, special inks to cryptography-based component authentication. Typical applied cryptography-based mechanisms employ a challenge-response-based component authentication mechanism. These component authentication mechanisms have been designed originally for local genuineness verification, i.e., for an authentication performed in direct vicinity of the component to be verified. However, it may be useful to support also a remote component authentication, e.g., to verify the integrity of the control system including its periphery from a central monitoring station. This paper describes an attack on a challenge-response component authentication protocol when using it in addition for a remote component authentication. A new security measure, that binds a challenge value to a specific remote verifier, is described to prevent this attack. The challenge value for which the response is calculated by the component authentication mechanism can therefore not be selected by the remote verifier. This has the advantage on one hand that a potential remote adversary cannot use the component as oracle to collect challenge response pairs. On the other hand, the response value can be provided to the verifier directly.**

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-oriented communication, also physical objects need to be authenticated in a secure way. Various 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 electrical 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 units or an earth leakage circuit breaker that do not provide reliably the expected functionality. Further examples can also be provided through metering or measurement systems like Phasor-Measurement-Units, which measure voltage and current magnitude and phase angle values at diverse locations in the energy transmission grid. If they report wrong values, the reliability of transmission grids may be endangered. Component authentication may be used also be vendors to identify replacement parts or consumables like ink or toner.

An approach for remote component authentication has already been described in [1] using a challenge-response authentication of a physical object. Focus of this paper is a further elaboration of this approach to provide more background to and insight into the proposed solution for.. Authentication of a physical object has the advantage that 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 component authentication is applied. Highly cost-optimized solutions are commercially available that allow to use these technologies also in low-cost devices. However, local object authentication are, often not being designed to protect against man-in-the-middle attacks as these may not be seen as relevant in the targeted use case. Hence, the applied protocols or methods may not directly be applicable to remote authentication. Here, especially Man-in-the-Middle attacks are in scope, if public networks are traversed.

Section II gives an overview on challenge-response based component authentication. The usage scenario for remote component authentication is described in Section III. This section also considers typical available 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 component authentication also for remote component authentication, i.e., for a usage scenario for which it has not

been designed in the first place. This enables to use extremely simple and therefore cost-efficient hardware-based device security mechanisms for new usages that have not been addressed originally. The countermeasure can be applied in particular even in those cases when the verifier needs access to the unmodified, raw 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

This subsection provides an overview about component authentication in general and specifically about commercially available implementation examples. Besides the pure mechanism overview application, some use cases are shortly presented to provide more insight into the value component authentication can provide.

A. Overview

Components of a machine (internal or attached) shall be identified securely. This requirement 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. This object calculates a corresponding response value which is returned to the requestor and verified. The response can be calculated using a cryptographic authentication mechanism or by using a physical unclonable function (PUF).

For cryptographic authentication different mechanisms may be used. Examples are keyed hash functions like HMAC-SHA1 or symmetric ciphers in cipher block chaining (CBC-MAC, see [10]) mode or symmetric ciphers in Galois counter mode (GMAC, see [19]) up to digital signatures. For the symmetric ciphers AES would be a suitable candidate. Common to keyed hashes or symmetric key based cryptographic authentication approaches is the existence of a specific key, which is only available to the object to be authenticated and the verifier, One resulting requirement from this fact is obviously the protection of the applied key, as the leakage of this key leads to attack options, which can be easily exploited. This may in turn lead to a higher administrative effort on both sides, the component and the verifier. Another requirement that may be derived is that the key should ideally be unique for a dedicated component and not only for a batch or version of that component. A reasoning for this requirements is that if the key of one component gets compromised, an attack is only successful for this specific component, but not for other components from the same series. Also asymmetric cryptography can be used for component authentication. A suitable procedure based on elliptic curves has been described in [17].

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.

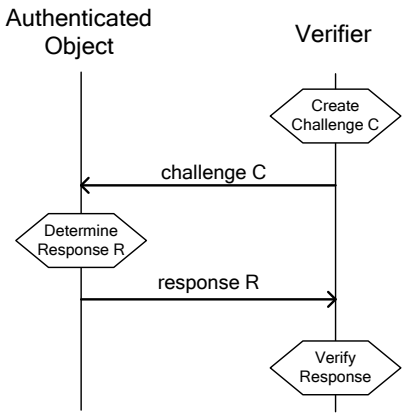

Figure 1. Challenge-Response Object Authentication

Figure 1 shows the schematic data flow between a verifier and an object to be authenticated. The verifier sends a random challenge to the component that determines and sends back the corresponding response. The verifier checks the response. Depending on the result, the component is accepted as genuine or it is rejected. The further handling upon detecting that a component or system has been compromised is a matter of the security policy and is not detailed here. Nevertheless, the reaction on detecting a failed authentication is use case specific and may vary form an immediate rejection of the component (like in the case of faked batteries, to ensure the safety of the device using the battery) to a graceful degradation mode of operation (e.g., in case of toner cartridges certain modes may not work, but the general functionality is still provided).

B. Implementation Examples

As stated in the previous section various cryptographic mechanisms can be used to realize a challenge-response product authentication. Basically, symmetric cryptography, asymmetric cryptography, or physical 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 himself. 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 could 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 cases, 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, without letting the verifier know.

Implementation examples of product authentication are summarized in the following list, providing one example per category (symmetric, asymmetric, and PUF based authentication):

- Atmel CryptoAuthentication [1], [3]: 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 [4]: 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. The product itself may provide its public key as a digital certificate (using internal memory), allowing for an offline verification of the response.
- Verayo RFID Tag "Vera M4H" [5]: 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

Applications of component authentication have already been given in the introduction targeting safety on one hand and protection of business models (and interests) on the other. As shown, a reliable identification of products is needed in various use cases. For safety reasons, components can be verified to ensure that no counterfeited products or product components have been installed. Detection of unverifiable product components may not necessarily lead to outages. Depending on the use case, the component may be operated with safe, conservative operating conditions (e.g., maximum charging current of battery pack) to prevent damage.

Component authentication can also be leveraged for centralized control, as the information about originally or falsified system parts may be use to provide system integrity information as part of an inventory management. Moreover, the component authentication can also be used to support the authenticated setup of a protected communication session for field level device communication. This approach is outlined in the following section.

III. Remote Component Authentication

One important class of use cases is remote component authentication. Here, a machine equipped with or connected to several field devices (sensors, actuators) performs local component authentication of installed machine parts or connected periphery in the first place. Additionally, it also supports a remote component authentication by a supervisory system, e.g., a control center or an inventory management. Remote component authentication is required in scenarios were sensors are used in a widely dispersed area. One example for such sensors are quality monitoring devices like synchrophasors (Phasor Measurement Units – PMU), which are used in the power transmission network to measure the phase angle and magnitude of sinus waves of voltage and currents. Based on this information, the network (and system) condition may be deduced.

A. Use Case Description

In a remote object authentication, the verifier is distant 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.

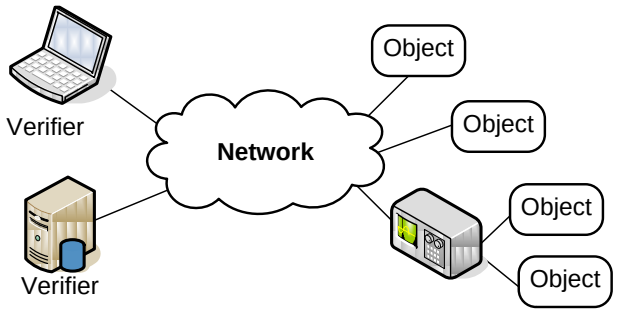

Figure 2. Remote Object Authentication

A verifier may be a service technician performing remote maintenance, or an automatic device tracking server or an inventory management server, see left part of 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, see right part of Figure 2.

The challenge response component authentication operation, i.e., the calculation of the response value for a given challenge value, is performed by the object to be authenticated as described above. However, here the verifier is not in close vicinity to the object to be authenticated so that an intermediary (man in the middle) may intercept and manipulate the exchange of challenge and response values. In some cases, a protected communication channel may be used between the verifier and the intermediary, e.g., IPsec or SSL/TLS (Secure Socket Layer / Transport Layer Security), to transport challenge and response values over an encrypted communication channel.

B. Man-in-the-Middle Susceptibility

In the case of remote object authentication, an adversary node may manipulate the exchange of challenge and response values when they are sent over a communication network. The adversary may act as remote verifier towards an object to be authenticated, and as authenticated object towards a well-behaving remote verifier. Such a malicious remote verifier acts as a man-in-the-middle attacker. Already the simple forwarding of challenge and response messages may constitute an attack, see Figure 3.

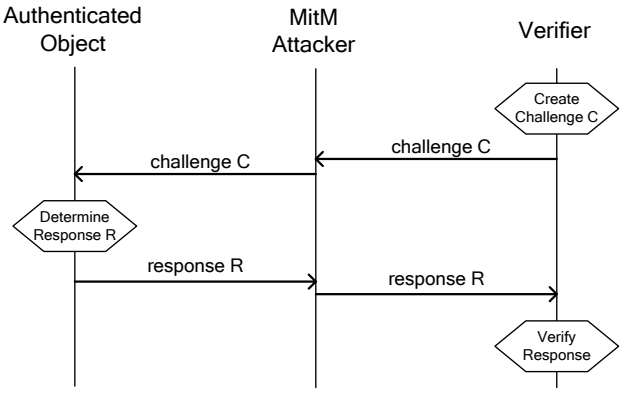

Figure 3. MitM Attack on Object Authentication

An adversary node as "man-in-the-middle" forwards unchanged messages towards and from the genuine object that is authenticated. The remote verifier having authenticated the object as genuine assumes that it is communicating in fact with the device in the following data exchange. However, in fact it is communicating with the adversary. An attacker can use an arbitrary, remote genuine object as an oracle that provides valid responses for an arbitrary challenge value. In consequence, any remote entity that has access to the object authentication functionality of a genuine object 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 remote object authentication can be manipulated.

When furthermore the authentication response is used for deriving cryptographic session keys, these keys could 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 [6]. In the considered usage environment where authenticated object and verifier are in direct physical vicinity, 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. The attack becomes relevant when the component authentication mechanism is applied within a usage environment in which not only a single verifier exists, but where a component may be authenticated by multiple verifiers, at least one of them connected only remotely or at least not in close vicinity of the object.

IV. Example for Cryptographic Binding Requirements

Binding cryptographic data as e.g., a session key to a specific usage context is a basic countermeasure to prevent attacks in which valid cryptographic data is applied by an adversary in a different usage context. The general need for cryptographic binding is motivated by describing a well-known weakness of the TLS protocol. 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 (Internet Message Access protocol), to realize VPNs (Virtual Private Networks) or for remote administration. Meanwhile the protocol is available as standard in version 1.2 as RFC 5246 [7].

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 establish fresh security parameters for an existing TLS 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 in a man-in-the-middle attack. A possible attack scenario – a request to a web server – is explained in the following, see Figure 4.

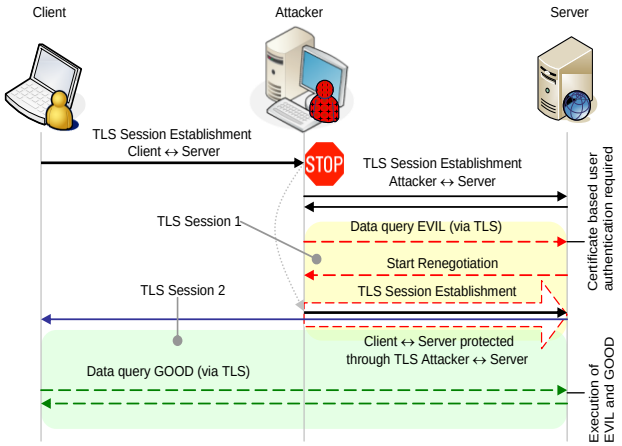

Figure 4. Man in the Middle (MitM) Attack on TLS Protocol

A potential attacker controlling the data path between a Web client (Web browser) and a Web server 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. In a variety of applications this connection is used with unilateral authentication, i.e., only the server authenticates as part of the TLS handshake. Now, the attacker himself establishes an own TLS connection to

the server, which is (as the original connection attempt by the client) unilaterally authenticated by the server side. So the client itself has not been authenticated by the server. As soon as the TLS connection has been established, the attacker sends a request *EVIL* over the established TLS connection. This request requires the authentication of the client. If the client is to be authenticated using certificates, it can be directly done within the TLS handshake, requiring the certificate based authentication of the client by the server. The attacker, however, does not possess the required client credentials. This client authentication is now invoked by the server through starting the TLS renegotiation. The *EVIL* request, however, which has not been authenticated at this point in time, is stored by the server and will be executed only after a successful client authentication. Web servers are typically configured in this way, i.e., to request a client authentication not at the beginning of a session, but only when a client performs a restricted operation, e.g., when data from an access protected directory shall be read.

In the attack scenario, the attacker now forwards the delayed *Client request* intercepted from the valid client to the server over the TLS protected link that has been established by the attacker with the server in response to the TLS renegotiation request. The server accepts and interprets the forwarded client message as valid part of the renegotiation phase. Note that the response to a *Start Renegotiation* message is the same message as used for the initial connection attempt – the *Client* request. During the renegotiation phase the server will require client side authentication in a subsequent message. This enables an end-to-end key negotiation between the client and the server. So a valid client authentication is performed. All subsequent messages are now secured end-to-end and the attacker is not longer able to access them. But then, the stored *EVIL*-request that has been sent by the attacker is executed by the server with the permissions of the authenticated client.

This attack shows on the one hand a weakness of the TLS protocol due to the missing cryptographic session binding of the two TLS sessions, i.e. the one established before performing the session renegotiation, and the one established as result of the session renegotiation performing also client authentication. If there would be a session binding, the Web server would realize that the *Client requests* does not refer to a former session in which the original client was not involved. The consequence is that the Web server is in fact communicating with two different entities (the attacker and the client), while it assumes that it is communicating only with a single entity. On the other hand, the attack 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 in the *ClientHello* and *ServerHello* messages [8]. To achieve this, client and server store the individually sent *verify_data* from the *Finished* message of the previous handshake and reuse this information in the *ClientHello* resp. *ServerHello* of the ongoing renegotiation handshake.

V. CHALLENGE BINDING (PRE-CHALLENGE)

The problem of the man-in-the-middle susceptibility of simple challenge-response component authentication originates from the fact that the same component authentication mechanism or the same associated authentication key respectively is used in different usage contexts. Following common security design, different keys would be used for different purposes. Furthermore the cryptographic material should be bound to the intended usage context (i.e., to derive context-bound session keys from the response).

As in important commercially available implementations of component authentication, the verifier needs access to the unmodified response; the response value cannot be modified practically. Therefore, challenge binding is proposed as countermeasure that can easily be integrated with existing component authentication mechanisms: When a remote verifier cannot select an arbitrary challenge value, it cannot use the authenticating object as oracle to determine responses for an arbitrary received challenge value.

A. Challenge Binding

The basic idea of challenge binding is to use a derived challenge value (bound challenge), which in turn is derived from a challenge value selected by a verifier (pre-challenge). For this bound challenge the response is calculated. The derived challenge is bound to a verifier by using an information of the verifier from which the (pre-)challenge has been received as derivation parameter. The (pre-) challenge selected by a verifier is thereby 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.

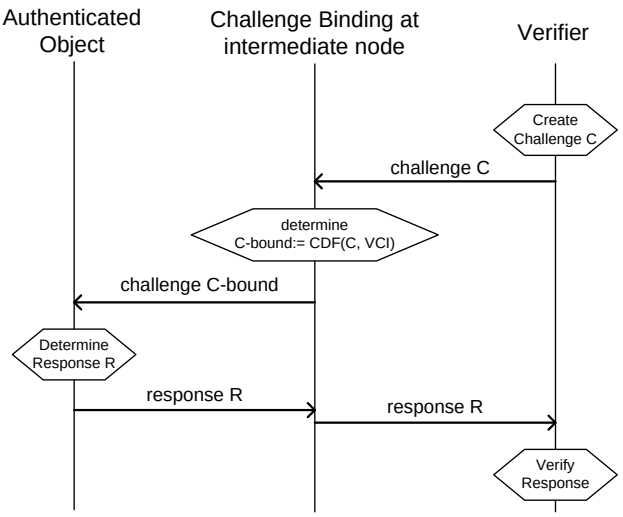

Figure 5. Challenge Binding

Note that the trusted intermediate node should be physically close to the authenticating object to avoid the

already described Man-in-the-Middle attack between the object and the intermediate node.

The challenge C (pre-challenge) 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. The challenge derivation can be performed by both, the object to be authenticated itself, or by a trusted intermediary node .For the following description, the challenge binding is performed by an intermediate (trusted) component. This challenge C is now 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.

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 in practice it must not be possible 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 [8] and [9] 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 [10] or in GMAC mode [19] as already noted in the overview of section II.

A further approach to be named here is the application of key wrapping as described for instance in [20] using AES. Here, the challenge could be used to derive an encryption key for the key wrapping algorithm. The information encrypted includes additional information provided by the verifier and generated by the authenticating component. Both pieces of information (encrypted part and generated part) need to be sent to the verifier. By decrypting this information with the key derived from the challenge, the verifier can proof if the decrypted content equals with the generated content. In the positive case, the component was successfully authenticated. This is depicted in the following figure.

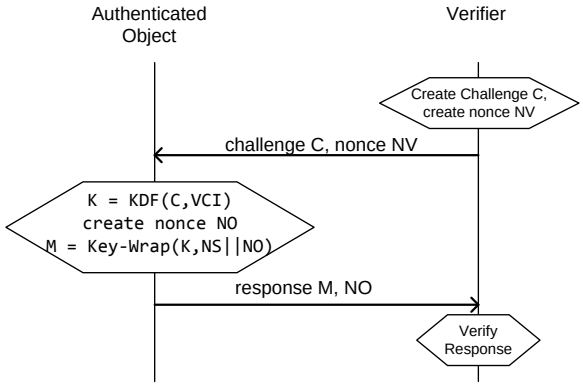

Figure 6. Application of Key Wrapping for Challenge Binding

General recommendations for key derivation functions using pseudorandom numbers are also provided in [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 networks wants to verify the identity of the smart object or 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 (Enhanced Authentication Protocol), 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 purpose and the used communication protocol can be used as challenge derivation parameter (e.g., *"DeviceComponent-Authentication/HTTP-DIGEST/TLS"* || *Server-IP*).

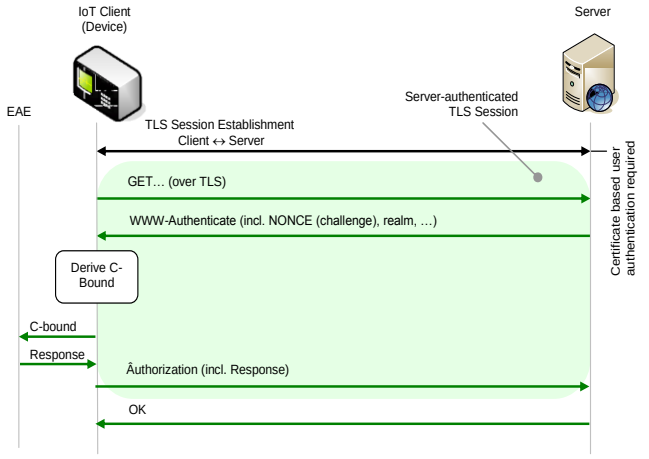

Figure 7. IoT Device Authentication Using HTTP DIGEST

Figure 6 shows an exemplary use for Internet of Things (IoT) device authentication. An IoT device includes a low-cost object authentication IC as embedded authentication element (EAE). The EAE realizes only a simple challenge response authentication mechanism. When the IoT device communicates with another device, e.g., a server, a server-authenticated TLS session is established. The device authenticates itself over HTTP using a modified HTTP digest. From the received NONCE value, i.e. the challenge value, a bound challenge value C-Bound is derived. The derivation parameters can include, besides the received realm, also further context information of the session, e.g. server IP address, server DNS name, server public key hash or server digital certificate hash, or the common name included in the server certificate. Also, the approach described at the end of section IV for TLS can be leveraged directly. This would result in the binding to the actual TLS session context, over which the authentication is being performed. The bound challenge is sent internally of the IoT device to the embedded authentication element EAE included in the IoT device. The received response is provided to the server for verification. While TLS has been used as example, also different protocols as datagram TLS (DTLS) or network / link layer security could be used as well.

The described IoT device authentication using the embedded authentication element may be used for device authentication during regular operation. Advantageous is, however, its use for automatically setting up the initial device configuration after the IoT device has been installed. This automatic security bootstrapping allows the IoT device to authenticate towards a bootstrapping server that provides security configuration data that is then used during the operational phase. Moreover, this bootstrapping server may additionally provide the functionality of a secure inventory management, which may need to be contacted by the IoT device in regular intervals. Thus, the server can check the component authenticity also regularly.

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 [11] 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 [12]. 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 (Protected EAP). 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 [8]. Another example is the pseudorandom function used within TLS [7], 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 [13].

It is also known to bind an authentication to properties of the used communication channel [14]. 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 [15]. 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.

In the “cuckoo attack”, an attacker causes a user of a computer including a trusted platform module (TPM) that a different TPM that the adversary controls resides in the user’s computer [16]. The adversary’s TPM can therefore make false assertions about the software running on the user’s computer. This attack is an example of a man-in-the-middle attack where the adversary sits between the verifier and the TPM.

Another option to bind an initial authentication to further challenges and authorizations tokens is the Kerberos, a well established security protocol, which has native support in several operating systems. The general proceeding of

Kerberos is described in [18] and depicted in Figure 8. In the target use case, the Kerberos server resembles the binding of an authorization request to a dedicated environment. It performs as a challenge response three party protocol, while the solution described in section V binds the challenge "on the way" to the object without having a three party (challenge-response) protocol. Furthermore, the solution proposed in this paper assumes physical closeness of the binding node to the actual object to be verified.

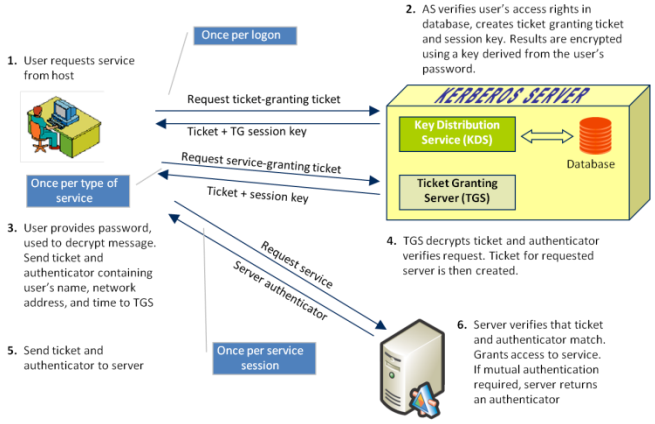

Figure 8. Kerberos Protocol Overview

An advantage of the proposed solution is that is uses less message exchanges between the participating nodes to bind the authentication to a dedicated environment compared to Kerberos stated above.

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 devices within the Internet of Things.

References

[1] R. Falk, S. Fries: Protecting Remote Component Authentication , The Fifth International Conference on Emerging Security Information, Systems and Technologies SECURWARE2011, 21-27 Aug. 2011, Nice / Saint Laurent du Var

[2] Atmel CryptoAuthentication, http://www.atmel.com/products/cryptoauthentication/, last access Jan. 2012

[3] Atmel CryptoAuthentication Product Uses, Application Note, 2009. http://www.atmel.com/dyn/resources/prod_documents/doc8663.pdf, last access Jan. 2012

[4] Infineon ORIGA, http://www.infineon.com/ORIGA, last access Jan. 2012

[5] Verayo http://www.verayo.com/products/unclonable-rfids, last access Jan. 2012

[6] Atmel CryptoAuthentication High level Security Models, Application note, 2009. http://www.atmel.com/dyn/resources/prod_documents/doc8666.pdf, last access Jan. 2012

[7] T. Dierks and E. Rescorla: The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, August 2008

[8] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov: Transport Layer Security (TLS) Renegotiation Indication Extension, RFC 5746, February 2010

[9] 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 Jan. 2012

[10] 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 Jan. 2012

[11] 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 Jan. 2012

[12] N. Asokan, Valtteri Niemi, and Kaisa Nyberg: Man-in-the-Middle in Tunnelled Authentication Protocols, LNCS3364, Springer, 2005.

[13] J. Arkkom, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman: MIKEY: Multimedia Internet KEYing, RFC3830, August 2004

[14] N. Williams: On the Use of Channel Bindings to Secure Channels, Internet RFC5056, 2007

[15] 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 Jan. 2012

[16] Bryan Parno: Bootstrapping Trust in a Trusted Platform, 3rd USENIX Workshop on Hot Topics in Security, July 2008, http://www.usenix.org/event/hotsec08/tech/full_papers/parno/parno_html/ , last access Jan. 2012

[17] M. Braun, E. Hess, and B. Meyer, "Using Elliptic Curves on RFID Tags," International Journal of Computer Science and Network Security, Volume 2, pages 1-9, February 2008

[18] C. Neuman, S. Hartman, K. Raeburn: RFC4120:The Kerberos Network Authentication Service, July 2005

[19] Morris Dworkin: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST SP 800-38D, November 2007, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf, last access June 2012

[20] NIST, AES Key Wrap Specification, November 2001, http://csrc.nist.gov/groups/ST/toolkit/documents/kms/AES_key_wrap.pdf , last access June 2012