2012 Third International Conference on Emerging Security Technologies

Identity Management and Integrity Protection in the Internet of Things

Anders Fongen
Norwegian Defence Research Establishment
Norway
anders.fongen@ffi.no

***Abstract*—Authentication and Identity Management help to protect resources and justify trust in "bona fide" operation by service client and service provider. Besides, identity management can support hardware assisted integrity protection. In the Internet of Things (IoT), the high number of lightweight devices requires scalable and lightweight solutions to trust management. The paper proposes a framework for authentication and integrity protection well suited for an IoT environment.**

I. Introduction

The expression "Internet of Things" (IoT) refers to an arrangement of the world where "everything" is uniquely identifiable and addressable through some kind of communication device, and where things can be located, employed, maintained and inspected for different purposes.

"Things" are, or represent, valuable resources, either as material value, or in the form of the service they offer. Resources need to be managed throughout their lifecycle, and access to resources need to be controlled and audited. The identity of resources and their clients (where applicable) should be managed, protected and maintained by an identity management system, and there must be mechanisms in place for authentication of identities, access control to the resources and protection of usage data for the sake of privacy protection.[1]

In the area of Identity Management (IdM), the concept of *identity assurance* is considered as an important tool for resource control. An IdM offers services for authentication, which means some kind of mechanism to identify the entity (e.g. a person) which operates on a resource. Authentication of a Thing does not necessarily mean assurance of identity, but rather assurance of its *genuineness* or *integrity*. Authentication is a prerequisite for auditing, accounting and access control, as well as personal application profiles and other services not related to security or accountability.

The principles of IdM are well understood, but IdM systems are complex and mostly found in relatively homogeneous environments where pervasive standards for information representation and network protocols can be enforced. The Internet of Things have inherent properties which represent challenges to the deployment of an IdM:

- The sheer scale of the system, potentially billions of things with short life cycle and high "birth rate".
- The heterogeneity of units, ranging from RFID chips with a minimum of processing power, communication capabilities and internal memory, to large scale computers with plenty of resources. No common standard for representation or transportation can be enforced on this range of equipment and several different standards are likely to co-exist.
- A high number of management domains. IoT devices will be managed and operated by a large community of enterprises and service providers. They are likely to employ vastly different naming policies, security frameworks, protocol requirements and access controls. In order to bridge these differences, traditional gateway nodes may need to be replaced by semantic processors.

In the rest of this position paper, some problems and research objectives related to IdM in the Internet of Things will be discussed. A set of proposed mechanisms for control of genuineness will be presented. Essential properties of the mechanisms will be their simplicity, prudence and adaptability.

II. Necessary system properties

The Internet of Things is expected to reach a scale which has never been observed before, and with a wider range of equipment with greatly different resources and capabilities. Besides, the future applications of IoT will surely come as surprises and will challenge our present view on how the IoT devices could collaborate to make business.

Therefore, the deployment of technology for IoT purposes should follow well known principles from large scale computing[2]:

A. Service Orientation

"Things" may look like passive objects, but may well be regarded as service providers. The simplest possible service is to reveal one's identity, which even a simple RFID device is able to do. It is therefore useful to put the Internet of Things into a Service Oriented perspective, in which all transactions have an initiator and a responder.

The loose coupling between client and service, and the clear separation of interface from implementation increase the chances for future interoperability between things.

B. Identity Management

Cryptography is easy, but key management is hard. Most security mechanisms relies on some crypto algorithms, and

978-0-7695-4791-6/12 $26.00 © 2012 IEEE
DOI 10.1109/EST.2012.15

they all employ keying material which must be deployed securely in the parties. Key management involves key generation, deployment, updating and removal, and to associate the keys with identities in a way that can be validated by everyone. Identity management also offers attribute management, in the form of properties securely bound to an identity in a validatable way.

For large scale IoT operation, the services of a slim and effective identity management are essential. The protocols and algorithms must be adapted to the resources available in and near the devices. E.g., extensive communication with central servers may be prohibitive and availability of cryptographic programming libraries may be limited.

The contribution of this paper builds on an existing identity management system which was developed with mobile systems in mind, and which is fully portable to Android smart phones[3]. To the author's knowledge, very few high end (to-way authentication, cross domain operation etc,) IdMs offer this level of portability. With the proposed modifications, it can also include lightweight units like those which are found in an IoT.

III. Authentication and the Internet of Things

The traditional meaning of the word *authentication* is "establishment of identity". This makes a lot of sense when humans are the target of the operation: an assurance that the operator of a system is a person otherwise known to be loyal and competent. The person is given a password or a hardware token to aid the authentication, and the authentication may be false if these are lost or compromised.

When a *service* is authenticated, we would not be interested in the identity of the server computer, but of the service itself, regardless if the server runs in a single computer, a cluster or in a public cloud. The organization that runs these computers ensures that only the given service can present the necessary credentials during authentication, and that they also ensure the integrity of the operation.

Common to these two variants is the assurance of a "bona fide" operation; That the parties do not *mislead* each other. A service or a client compromised by malware etc. will not be detected through a normal authentication operation.

A. Tamperproof authentication

An authentication operation of a person may be subverted if the token is lost or stolen, or if the password is revealed to others (voluntarily or forcibly). This risk raises the need for *revocation* of the token, i.e., that the tokens is deemed invalid and made useless. From research on Public Key Infrastructure (PKI) we know that the revocation mechanism is expensive and difficult to operate in a scalable fashion.

Authentication of a thing, on the other hand, may me made *tamperproof* during the manufacturing process. The tokens may be stored in hardware devices and sealed in such a way that the token or the thing is destroyed if they are separated. So, in order to authenticate a false Rolex watch one has to destroy a genuine one, which is not a viable business model for criminals. The integrity of a goods container or a computer box may be sealed with an authentication token so that the token is destroyed if the box or container is opened.

In the context of an identity management system, the existence of a tamperproof authentication mechanism can alleviate some of the cost related to the revocation of tokens. It is a reasonable assumption that an authentication operation based on a tamperproof mechanism can be validated with less scrutiny, and that the revocation status of the keying material can be assessed less often. This argument becomes important when the necessary networking capacity near the things is being planned.

B. Identity management without revocation lists

The experimental IdM described in [4] and [5] have been built on the assumptions that credentials should not be subject to revocation, but rather repeatedly issued with short lifetime [6]. It is currently being extended to incorporate not only validation of identity, but also of *service integrity*, a concept which will be described later in the paper.

IV. Trust through attested genuineness

Authentication supports the assumption that a transaction will take place according to expectations and agreements and in a "bona fide" manner. What the client also needs is assurance of the service integrity, i.e., that it is unadulterated from malware or other hostile modifications.

There are several approaches to support service integrity. The traditional approach is to employ an operating system with rigid separation between processes, so that malware in one process cannot affect other processes. This separation property is found in Multi Level Security (MLS) operating systems, but they still require sophisticated software maintenance to avoid vulnerabilities in the code running in the privileged processes. Besides, MLS systems are few, expensive and incompatible with COTS software.

A different approach, feasible only under certain conditions, is to *seal* the software configuration with a hash value safely stored in an external repository. During operation, the hash value is regenerated and compared to the "sterile" value in order to detect any infections. The generation and check of the hash value must be *non-bypassable, tamperproof, protected from replay attacks and independently attested*.

A successful integrity check must be attested by a trusted third party found inside an IdM which is called an "Identity Provider" (IdP). An IdP issues credentials for authentication and access control and may also attest the validated hash value given from the integrity control device. A client of that service must be able to validate the attestation before accepting the service response.

A. Trusted Platform Module

A range of hardware units for authentication and integrity control may be considered for this application. Simple units exist for basic challenge response authentication exists, they either employ symmetric keys or asymmetric keys. For hardware assisted integrity control the only unit known to the author is the Trusted Platform Module (TPM) designed by the Trusted Computing Group consortium.[7] The TPM is a crypto processor able to perform a variety of key storage operations, crypto operations and configuration checking. While space does not allows a detailed description of TPM operations, suffice it to say that it cooperates well with an Identity Management system in order to attest a valid configuration, but the effectiveness of the mechanism relies on a quite detailed cooperation with the operating system. More details can be found in [8].

B. Simple protection units

Whereas the TPM is a device meant for PCs, cheap and battery operated service providers may prefer a simpler device with appropriate cost and power consumption. These devices could operate with a simple challenge-response mechanism involving an HMAC function over a secret key, and an integrity protection unit that would intercept the reset vector of the processor, scan the memory for the hashing function and include the hash value as a part of the HMAC function parameters.

$$hmac = f(K, h(mem), challenge) \tag{1}$$

The hash function h is not evaluated for each call to f, but during bootstrap and later when felt necessary. The result is cached for use as a parameter in the f function. It is imperative that the secret key K is never employed in any other operation than f, where $h(mem)$ always is a parameter. The parameter *challenge* is client-supplied.

The arrangement shown in Equation 1 enables a tamper-proof hardware unit to ensure that the *hmac* response from a challenge is dependent on the memory content, and that validation of the value guarantees the correctness of the secret key as well as the memory content. The processor design should inhibit execution of memory locations not covered by the h function.

A HMAC-based authentication does not secure the subsequent message traffic from a man-in-the-middle attack, but the HMAC function may be used to establish a shared secret between the client and the service which can be used for message encryption in order to solve that problem. The next section will describe a protocol through which a trusted third party (an Identity Provider) can manage keys and identity information and cooperate with the hardware unit to attest the genuineness of the service.

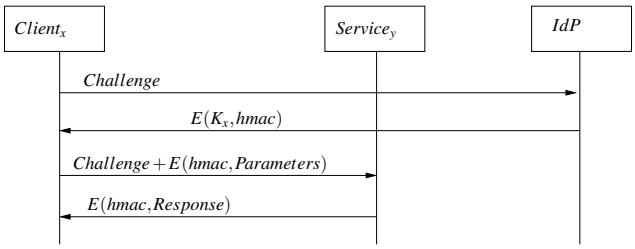

Figure 1. The protocol for protection of genuineness, based on symmetric crypto keys and the HMAC function given in Equation 1.

C. Protocols for attestation of genuineness

1) Symmetric keys: It will now be shown how a tamper proof HMAC function as described in Section IV-B can support a protocol which allows a client to trust the genuineness of a service. The protocol will have these properties:

- Only the identity provider (IdP) knows the secret key and the valid memory hash value of the service.
- No online connections between the service and the IdP is necessary.
- Genuineness control and service invocation happen in the same protocol round trip in order to save network resources.
- Client and Service need software libraries for symmetric crypto operations.

Figure 1 shows the transactions of this protocol. It assumes that the function in Equation 1 and the secret key K_y is available in the IdP and the service. Is also assumes the existence of the crypto operations $C = E(K,P)$ and $P = D(K,C)$. Note that the *hmac* value is not transferred to the service, but calculated in the service and used as a key for subsequent message transport. The client need the *hmac* value which is sent from the IdP encrypted with its secret key $E(K_x, hmac)$. The parameters for the service request are included with the challenge in encrypted form, and the service will only be able to decrypt it with a correct secret key and correct $h(mem)$ value.

The existence of the client secret key K_x indicates that the client must identify itself to the IdP, and the IdP could employ access control to decide if the operations should complete. The IdP could also employ means to detect repeated requests as a chosen plaintext attack on the HMAC function.

2) Asymmetric keys: As an alternative, a hardware unit could be wired to conduct operations on asymmetric keys, something that would create a simpler and safer key management. The hardware would offer a *decrypt* function related to Equation 1 in the following fashion:

$$K_{sess} = decrypt(K'_y, h(mem), C) \tag{2}$$

where

$$C = encrypt(K_y, h(mem), K_{sess}) \tag{3}$$

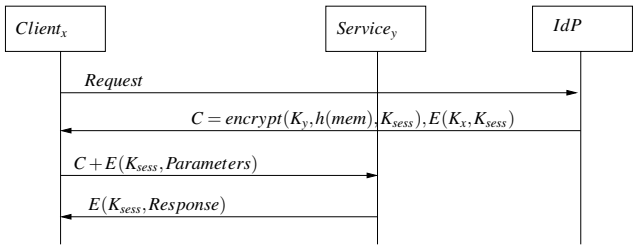

Figure 2. The protocol for protection of genuineness, based on asymmetric crypto keys and the *decrypt* function given in Equation 2.

and (K'_y, K_y) is the key pair belonging to service y and K_{sess} is a session key chosen by the IdP. The client receives the session key encrypted with either its symmetric or asymmetric key, allowing this asymmetric scheme to be employed regardless of the client's capabilities.

The only way the service can retrieve the session key is to employ the *decrypt* function with the correct keying material and the correct memory content. If the operation fails, the service is unable to participate in subsequent communication with the client. This may seem as an nontraditional outcome of a failed authentication, but it saves extra protocol round-trip operation, which conserves resources and saves time. Figure 2 provides the detail of the message sequence.

The main advantage of the asymmetric scheme is that no secret key needs to be stored in the IdP. This allows for a more relaxed protection of the IdP service endpoint, as well as *cross-IdP* operation, where the client and the service rely on different IdP instances.

D. Access control

The client access to a service should be subject to access control. The service could do its own access control, which requires frequent access to an IdP for credential validation. The choice of this paper has been to leave that control with the IdP, which can make the necessary decisions based on identity information kept in the IdP (or issued from other trusted sources). A client being denied access will not receive the data necessary to invoke the service.

In a large scale, multi-domain community the access control must be role based (RBAC). The RBAC method will relate privileges to roles and roles to identities, allowing "bulk control" of user groups, also across management domains.

V. Related research

Related research include works by Gilbert et al. [9] and and Saroiu et al. [10], both of which attempt to protect the integrity of sensor readings through a tamperproof signing mechanism akin to a TPM. This author believes that the protection of no less than the entire sensor node is required. A "protected" sensor can assure the values of a sensor reading, but not its *timeliness*, and protection against replay attacks is infeasible without trust in the communication stack of the service. The papers mentioned also disregard the complexity of key distribution in a network with a large number of nodes. Our contribution underpins the necessity of a scalable system for identity and key management.

VI. Conclusion

The provision of adequate authentication and identity management service in a community of extremely lightweight units has been the objective of this position paper. The combined control of authenticity and service integrity is offered by a simple hardware unit and simple cryptographic operations.

The limitations of the approach is the static view of program memory in the service. In virtual memory systems where program code is places on arbitrary physical locations and program libraries are loaded into memory on demand, this approach will be less useful. In those cases a more sophisticated OS will serve the application, and the TPM hardware unit would be a better choice.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] L. Tan and N. Wang, "Future internet: The internet of things," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 5, aug. 2010, pp. V5–376 –V5–380.

[3] A. Fongen, "Federated identity management for android," in *SECURWARE 2011*. Nice, France: IARIA, July 2011.

[4] ——, "Identity management without revocation," in *SECURWARE 2010*. Mestre, Italy: IARIA, July 2010.

[5] ——, "Architecture patterns for a ubiquitous identity management system," in *ICONS 2011*. Saint Maartens: IARIA, Jan. 2011.

[6] ——, "Optimization of a public key infrastructure," in *MILCOM 2011*. Baltimore, MD: IEEE/MILCOM, Nov 2011.

[7] Trusted Computing Group, "TPM Main Specification," http://www.trustedcomputinggroup.org/resources/tpm_main_specification, online, Accessed Mars 2012.

[8] A. Fongen and F. Mancini, "Identity management and integrity protection in publish-subscribe systems," in *Under review for MILCOM 2012*. Orlando, FL, USA: IEEE/MILCOM, Oct 2012.

[9] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '10. ACM, 2010, pp. 31–36.

[10] S. Saroiu and A. Wolman, "I am a sensor, and i approve this message," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '10. ACM, 2010, pp. 37–42.