The 10 Laws of Smart Object Security Design

Hannes Tschofenig*, Bernard Aboba†
*Nokia Siemens Networks, Email: Hannes.Tschofenig@nsn.com
†Microsoft, Email: bernarda@microsoft.com

I. Abstract

This position paper aims to provide some high level recommendations for integrating security into smart objects connected to the Internet. Many of these recommendations are taken from the experience with the work on Internet security.

II. Design Patterns

1) **Do Not Design for a Single Credential Type.**

There are two basic credential types, namely symmetric as well as asymmetric credentials. However, these credentials can be further categorized into the amount of entropy a symmetric key provides (e.g. a password that can be remembered by a human vs. a randomly created shared secret with 128 bits), into the type of asymmetric credential, and also whether there is a need to integrate the credential into a certain legacy infrastructure (e.g. cellular infrastructure).

Historically, there has not been an agreement on a credential type that has to be supported in all environments. The reasons are manifold and it is very unlikely that there will be an agreement for a single credential type for smart objects as well. It is also not wise to assume a single end user registration and credential provisioning approach since many different approaches are in place today and they depend on the level of security that shall be achieved, the user experience, the business model, the device capabilities (e.g. presence of a display, and input devices) and the availability of certain communication networks.

It should also be understood that energy or CPU considerations do not necessarily preclude use of asymmetric key cryptography. Even though public key cryptography can be CPU intensive, protocols such as TLS can cache computed session keys, and public key cryptography can also be used in provisioning of symmetric keys. Examples for caching keying materials can be found with TLS

Position paper for the 'Interconnecting Smart Objects with the Internet' workshop, Friday, 25th March 201, Prague. This paper represents the views of the authors in their role as individual contributors to the Internet standards process. They do not reflect the consensus output of any IETF working group or the IAB. Referenced documents may, however, reflect IETF consensus. Hannes, and Bernard are members of the Internet Architecture Board (IAB) and have contributed to the standardization efforts discussed in this document.

session resumption [1], session resumption without server-side state [2], or IKEv2 session resumption [3].

2) **Do Not Design for a Single Authentication and Key Exchange Protocol.**

Once a specific credential type has been chosen there are still a number of design choices for developing an authentication and key exchange protocol. As an example, some commonly demanded security requirements for two party authentication and key exchange protocols are listed in [4] (with a focus on network access authentication). These requirements already indicate the wide range of protocols one could design. With three party protocols there are even more design choices and [5] documents some fundamental choices. Interestingly, the requirements also change over time. The requirement for user identity confidentiality was not often provided by protocols designs a couple of years ago while the increasing awareness for privacy has substantially increased the interest. It is quite likely that new requirements may be added over time, such as the need to consider energy efficiency.

It is unrealistic to assume that there is an agreement on a single protocol even for a single environment and even less so for a large number of environments.

3) **Consider Crypto-Agility From the Beginning.**

Crypto-agility is the ability for a protocol to adapt to evolving cryptography and security requirements. This may include the provision of a modular mechanism to allow cryptographic algorithms to be updated without substantial disruption to fielded implementations. It may provide for the dynamic negotiation of cryptographic algorithms within protocols.

Standardization of protocols takes a long time; changing an installed base is even harder. When cryptographic algorithms, such as hash functions, get compromised it is important to offer a smooth migration to new algorithms. In today's standardization environments it is common to consider this unavoidable transition already in the design of the initial protocol by offering modularity and the capability for algorithm and ciphersuite negotiation.

To avoid potential downgrade attacks, it is particularly important to consider the longevity of ciphersuites used to secure the negotiation itself.

4) **Security Requirements Vary.**

While it may seem obvious, it is important to point out that security requirements vary from environment to environment, from application to application, and also over time. For this purpose, a threat analysis is done for every protocol and application. In the IETF such analysis is demanded already since the publication of RFC 1543 [6], which was later updated by RFC 2223 [7].

Most important, while it is easy to define a beautiful security architecture on paper it is hard to actually deploy it. This is one of the most painful experiences observed with today's Internet. For example,

the security threats around the Domain Name System (DNS) have been known for a long time and the standardization work on DNSSEC has also long been finalized. However, it required years to start deployment and the DNS cache poisoning security vulnerability discovered by Dan Kaminsky. In real world deployments the cost-benefit tradeoffs heavily impact decision making and even more so if the costs and the benefits are distributed among different parties.

While there are some common security services demanded by every protocol, such as channel security, the details often vary. This is particularly true for more complex multi-party protocols. Since the range of applications for the area of smart objects is as probably as large as those of the Internet today it is unlikely that even a small fraction of these application requirements can easily captured today, particularly since many of these application needs are not yet know.

Over time, as we have seen in the history of security on the Internet, the security needs also change: new threat models arise (e.g. attacks from script-kiddies to those motivated by governmental organizations, adversaries develop new attack tools, and the cost of certain technologies also change (such as the cost for bandwidth). With increasing popularity of a certain communication environment the number of attacks also increase as a result of pure economical calculations of those who are responsible for those attacks.

5) **Link Layer Security is not a Substitute for Application Layer Security.**

A well established concept in the design of protocols and architectures is to utilize layers as an abstraction concept. A natural question is therefore whether security has to be provided only at one layer or at multiple layers. The simple response is that you have to do it at multiple layers since the communication end points at these various layers are different. Nevertheless, when the same entity is providing both network connectivity as well as application layer services then such an ”optimization” seems worthwhile to explore. However, making such an assumption in the architectural design often leads to additional complexity later on when it is realized that there are use cases where such a relationship does not exist and two independent entities provide network access and application services. Furthermore, the additional benefit from a performance point of view is often lessened because a security context needs to be bound to the higher layer protocol and cryptographic keying material need to be derived such that it is fresh and unique. Furthermore, authorization needs to be considered and that may add additional exchanges.

In addition to the different end points for the executing the security protocols there are often, if not always, different security requirements being demanded at higher layers were additional semantics becomes really important.

6) **Application Layer Security is not a Substitute for Link Layer Security either.**

Considering the previous claim that link layer security is not a substitute for application layer

security one could easily come up with the idea to just go for application layer security and ignore security at the lower layers. That is, however, a naive approach that misses the reason for doing security at the lower layers in the first place. The main reason for link layer authentication is for getting access to a specific network, which may provide access to the Internet.

A similar function is the access to the an enterprise network that may require a VPN tunnel (IPsec, SSH, or TLS/SSL-based). There, all traffic needs to data origin authenticated, integrity and confidentiality protected to be granted access to the company network.

7) **Consider Updates.**

While proper protocol design is important for security it is not the only aspect to consider. In fact, most security vulnerabilities are a result from buggy software. If you do not have a story for how to provide software updates in an easy and timely fashion then problems will occur.

Some solutions provision software updates as part of device management, while other approaches allow downloading the latest code whenever the devices are connected to the server-side infrastructure. For a smart object, the mechanism by which updates are provided needs to be compatible with energy management concerns. For example, the smart object may need to be woken up on receipt of an update notification, or when awakening from a long sleep, may need to protect itself against compromise while checking for updates.

Smart objects typically do not have a user sitting in front of them nor the ability to easily let the user confirm the software update or to decide about extent of the updates required; therefore updates need to be designed to function in the absence of an administrator. Just because a smart object has been authenticated to access the network does not mean that it can be completed trusted. Devices operating outside of their design points should be considered suspect, regardless of the security of the cryptographic mechanisms they imply. If possible, mechanisms should be employed to verify the integrity of code running on the device, or to enable compromised devices to be refreshed even while under the control of an attacker. Technical solutions, for example, have been developed within the IETF NEA working group [8] to verify software versions, software patches, etc.

Failed software updates can also render the system unstable, particularly with firmware updates. Many embedded system operating systems lack more advanced security concepts widely used in desktop and server operating systems. Failures in the update of one application may have an impact to other applications.

The economics of product life cycles also needs to be taken into consideration. Some vendors may see broken devices as an ”opportunity” to sell new hardware. ”End of life” means no more security updates and bad code; for certain vendors this may mean the path to new revenue.

8) **Don’t Skip Security.**

Security is expensive: additional code, more memory consumption, added protocol exchanges, negative performance impact, operational costs, and lets to a more complex architecture. Specifically for constrained devices, like smart objects, this is a serious issue. While it might be tempting to reduce security functionality developers have to select a platform that offers capabilities sufficient for performing the security mechanisms as identified by a prior threat analysis. Additionally, to deal with software updates related to discovered security vulnerabilities additional storage needs to be provided. Performance concerns include volatile and persistent memory, and CPU speed.

Without capabilities to protect against basic security threats, these devices should better be deployed in isolation rather than connected to the Internet to avoid creating a new platform for adversaries in mounting attacks against other devices. In a nutshell: If you cannot afford security do not connect to the Internet. However, as we have learned with the StuxNet virus, even systems on private networks can be attacked, so that isolation from the Internet is not guarantee against compromise.

9) **Reuse is Good, if Done Wisely.**

When functionality is expensive then a frequent idea is to perform the heavy function only once and re-use it for other transactions. That's clear a good strategy and the core idea behind the split between authentication and key exchange protocols, which leads to the establishment of security associations, for the purpose of establishing a secure channel (with data origin authentication, integrity, and confidentiality protection). The performance impact by the initial authentication and key exchange will be considerably larger than the subsequent performance impact for protecting data traffic using block and stream ciphers, or applying key message digest. Optimizations should therefore focus on the initial, performance intensive steps.

However, re-using a security association when the end points are different, or when the security goals are different is not possible without further computation (and potentially protocol exchanges). Granting one application access to security association parameters established by another application is, from a security point of view, not encouraged or even harmful.

Equally important for a memory constrained device is to re-use security libraries. For example, the TLS [1] code used within EAP [9] methods like EAP-TLS [10], EAP-TTLSv0 [11], PEAP [12] or EAP-FAST [13] can be used not only for network access but may also be used by application access, as currently envisioned by the ABFAB working group [14]. The same can be said about libraries for IKEv2 [15], SASL [16], GSS-API [17], or CMS [18].

10) **Take Available Specifications into Consideration.**

While computer science is a relatively young disciple, compared to other sciences, there are still a number of research results, standardized security technologies, and deployed systems available. While not popular among researchers it is still a good practice to re-use existing work, even if it is only to

demonstrate awareness of the state-of-the-art. While this is a generic guideline, not only useful for the re-use of security technology, there are a bunch of widely used security techniques available. [19] aims to provide a short overview of some of them.

Sometimes existing building blocks cannot be re-use, even do they offer lots of interchangeable parts. In that case it is useful to fully understand why other tools are not appropriate to gain from the experience obtained from the previous research, standardization, and deployment efforts.

III. Conclusion

From a security point of view it is extremely unlikely that there is a one-size-fits-all solution. Since security functionality is incorporated into each and every protocol and since there is very likely no single ”right” protocol architecture (particularly at the application layer) either there will be no single story either.

There are many people out there searching for the ”best” protocol stack that every device has to support and who believe into a ”community design approach”, i.e. a forum or organization where members from all different stakeholders just make the decision. The author believes that such an approach is flawed and will not lead to the innovation people are looking for. For this purpose, other approaches will quickly emerge, like they do on the Internet today.

As an alternative approach, design for flexibility and leave room for innovation. Requirements change, the technical limits of today will be forgotten tomorrow, and design philosophies constantly evolve. An architecture that places lower requirements on interoperability will allow innovation at a faster speed and will more likely win in the market place.

As an example, consider today’s Internet where everyone is free to develop and deploy the applications they like best. It turned out that the Web, as one application utilizing HTTP as transport, was more successful than others in the degree of deployment, flexibility, and widespread support. In a recently published position paper [20] we illustrated factors that contributed to that success. Can we support a similar environment to let the smart object space to flourish?

To summarize this paper, design your system with innovation in mind, be prepared for constant change, and let others explore their application ideas.

References

[1] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” Aug. 2008, RFC 5246, Request For Comments.

[2] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, “Transport Layer Security (TLS) Session Resumption without Server-Side State,” Jan. 2008, RFC 5077, Request For Comments.

[3] Y. Sheffer and H. Tschofenig, “Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption,” Jan. 2010, RFC 5723, Request For Comments.

[4] D. Stanley, J. Walker, and B. Aboba, “Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs,” Mar. 2005, RFC 4017, Request For Comments.

[5] P. Horster and H.-J. Knobloch, “Protokolle zum austausch authenischer schlüssel,” in VIS, ser. Informatik-Fachberichte, A. Pfitzmann and E. Raubold, Eds., vol. 271. Springer, 1991, pp. 321–328.

[6] J. Postel, “Instructions to RFC Authors,” Oct. 1993, RFC 1543, Request For Comments.

[7] J. Postel and J. Reynolds, “Instructions to RFC Authors,” Oct. 1997, RFC 2223, Request For Comments.

[8] IETF, “Network Endpoint Assessment (nea),” Feb. 2011, http://datatracker.ietf.org/wg/nea/charter/.

[9] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Simple Authentication and Security Layer (SASL)," Jun. 2004, RFC 3748, Request For Comments.

[10] D. Simon, B. Aboba, and R. Hurst, "The EAP-TLS Authentication Protocol," Mar. 2008, RFC 5216, Request For Comments.

[11] P. Funk and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)," Aug. 2008, RFC 5281, Request For Comments.

[12] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson, "Protected EAP Protocol (PEAP) Version 2," Oct. 2004, IETF draft (work in progress), draft-josefsson-pppext-eap-tls-eap-10.txt.

[13] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)," May 2007, RFC 4851, Request For Comments.

[14] J. Howlett, S. Hartmann, H. Tschofenig, and E. Lear, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture," Dec. 2010, IETF draft (work in progress), draft-lear-abfab-arch-01.txt.

[15] C. Kaufman and P. E. P. Hoffman, Y. Nir, "Internet Key Exchange Protocol Version 2 (IKEv2)," Sep. 2010, RFC 5996, Request For Comments.

[16] J. Myers, "Simple Authentication and Security Layer (SASL)," Oct. 1997, RFC 2222, Request For Comments.

[17] J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1," Jan. 2000, RFC 2743, Request For Comments.

[18] R. Housley, "Cryptographic Message Syntax (CMS)," Sep. 2009, RFC 5652, Request For Comments.

[19] F. Baker and D. Meyer, "Internet Protocols for the Smart Grid," Nov. 2010, IETF draft (work in progress), draft-baker-ietf-core-11.

[20] B. Aboba, J. Peterson, H. Schulzrinne, and H. Tschofenig, "The Future of Web Applications or How to Move into the Post Standardization Area," Oct. 2010, position Paper for the 'Real-Time Communication in the Browser' workshop, Mountain View,.