An Ownership Model and a Security Policy Definition Language to Security Bootstrap Ad-hoc Distributed Systems

Christian Rohner

Uppsala University, Dept. of Information Technology, SE-751 05 Uppsala
christian.rohner@it.uu.se

Abstract. We believe that the critical aspect in providing security for pervasive environments is building security and trust relations between devices without requiring the users to be experts. In this tech-note we present two complementary mechanisms to bootstrap security relations between networked devices: the ownership model and a security policy definition language. The ownership model assures security relations between devices owned by the same user, and the security policy language defines security relations to other devices, assigns rights to relations, and supports authentic key exchange.

1 Introduction

We observe the trend that more and more devices get equipped with some computing power and a communication interface. The communication interface is typically a short-range wireless technology that allows the devices to interact whenever and wherever they meet, without relying on a central infrastructure. Such networked devices often store personal and private information, have limited resources, and operate in an open and unknown environment. It is therefore important to restrict access to device information and resources. While the necessary security mechanisms (i.e., access control and potentially encryption) are well known, it is not clear how to build security relations on which these mechanisms rely.

A security relation between two devices can be a common shared cryptographic key or an authentic copy of a public key. It allows the two entities to exchange messages in an authentic and/or confidential way. Security relations need to be bootstrapped, that is, keys have to be generated, authenticated, and distributed. These operations cannot be expected to be done by an average user because he might not be aware of the importance and subtleness of the different steps.

In this paper we present an approach to bootstrap security relations by forming federations of devices owned by the same entity. Having a federation of own devices seems to be an intuitive concept to the user and through redundancy of relations simplifies recovery from exceptions like loss of devices. A security policy definition language is proposed to extend the security relations to other devices and assigning rights to relations.

1.1 Pairing

Building security relations between devices in an environment when one cannot rely on pre-defined security relations, central services (e.g., an administrator), or the availability of dedicated entities such as trusted parties is typically done by bilateral pairing of devices upon need. The goal of pairing is the authentic key exchange between two devices. Because it is often not desired nor needed to build relations to just any device, dedicated pairing is done between devices in one another's proximity and initiated by the user.

Implementing pairing is delicate because the security of the system relies on the resulting security relations. Weak pairing results unavoidably in weak system security. Although some wireless communication technologies such as short range radio or infrared reduce the probability of attacks to a limited range, one must not rely on that fact. There are three typical approaches to implement pairing: physical contact, variations of the Diffie-Hellman key exchange protocol, and password authenticated key exchange. Physical contact is a rigid approach avoiding sniffing and man in the middle attacks at all. Variations of the Diffie-Hellman protocol (e.g., [1], [2], [3]), which by itself is prone to man in the middle attacks, relax the requirements on the communication technology used. They can be combined with approaches using (human memorable) passwords to guarantee authenticity (e.g., [4]). New user interface paradigms abstract from traditional PIN-like passwords by, for example, using a shaking pattern as a password [5].

1.2 The Resurrecting Duckling Policy Model

Frank Stajano and Ross Anderson were the first that recognized the importance for building security relations between networked devices. In their Resurrecting Duckling Policy Model [6] [7], they propose two basic elements to extend pairing:

– *Secure Transient Association*: Exchange of a shared secret during pairing (physical contact) representing a master–slave relation between the devices.
– *Default policy*: Assigning rights to associations. The master device is the only device that can access services on the slave device.

While the relation between devices is a static master–slave relation in the original paper [6], an extension to pair-wise peer-to-peer relations is presented in [7] by introducing policy updates that describe relations to other devices in the security policy. The two basic elements are necessary and sufficient to bootstrap security relations.

1.3 Contributions

Master–slave relations between devices introduce static and dedicated dependencies between devices that limit the usability and are prone to loss of devices. Although pair-wise peer-to-peer relations partially address this shortcoming, the resurrecting duckling policy model does (1) not suggest how the credentials (i.e., keys and certificates) to represent peer relations could be described in the policy in an authentic way. Further (2), the lifecycle of a security association is rather static, either imprinted or not: Delegation and exception handling (i.e., loss of devices) are not supported.

In the next three sections we present an extension to the Resurrecting Duckling Policy Model that addresses these two shortcomings. The *ownership model* builds peer-to-peer security relations between all devices owned by the same user and thereby strictly defines trusted devices. The *security policy definition language* defines relations to other devices in an authentic way, assigns rights to security relations, and supports authentic key exchange. The third extension addresses *delegation* to support passing on or loaning devices to new owners and to leverage of previously built security relations, for example, initialized by the manufacturer of the device. The basic idea of the ownership model and the security policy is illustrated in Figure 1.

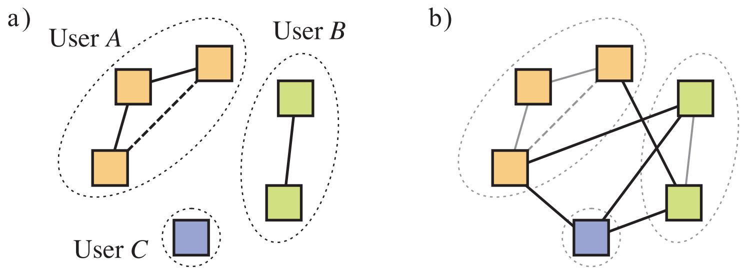

Fig. 1. (a) The ownership model forms federations by building security relations between devices owned by the same user. (b) The security policy defines security relations to other devices and assigns rights to security relations.
Legend: *solid line* - security relation represented by a certificate. *dashed line* - security relation implied by a certificate chain.

By building federations of devices owned by the same entity, we clearly define trust as trusting another device to follow the protocols and to forward only authentic information. At the same time, simplify the dependencies between devices and we introduce redundancy that can be used to handle exceptions. We have implemented the proposed mechanisms and were successful in hiding the complexity of key handling and security protocol configurations from the user [8].

2 Ownership Model

We propose to build federations of devices owned by the same user or organization. Devices owned by the same user are likely to interact with one another, and federating them allows at least an informal association of them to the user as a person. Our approach uses a pre-installed cryptographic public key as identifier of a device and the corresponding cryptographic private key as credential to prove the ownership of that identity.

2.1 Building a Federation of Devices

When two devices, one device creates a certificate for the other device by signing the public device key of the other device. The resulting certificate chain leads towards one of the user's devices and is used to recognize other devices owned by the same user. The initial security policy of a device defines the same default rights as the Resurrecting Duckling Policy Model.

Figure 2 illustrates the achievements of the ownership model. A particular advantage of this approach is that a new device has to be paired only with one device of a federation to get an authentic relation to all other devices of that federation. Devices of the same federation are recognized based on their certificates leading to a common device. Inefficient certificate chains are optimized automatically when a device interacts with a device higher in the certificate hierarchy. The ownership model further provides redundancy to cope in situations where devices get lost or delegated, because a device is not dependent on a dedicated other device such as in master–slave or pair-wise peer-to-peer models.

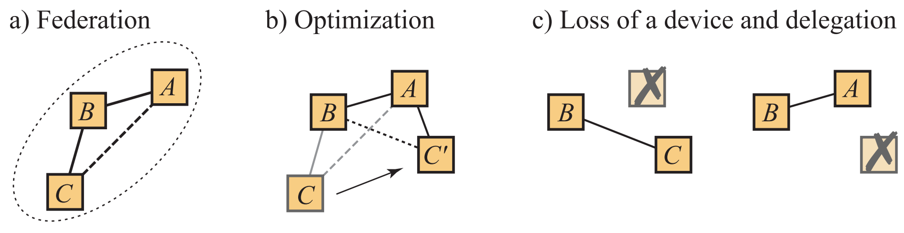

Fig. 2. The ownership model simplifies adding new devices and supports situations where devices get lost or delegated to another user. (a) New devices can be added to a federation though pairing with an arbitrary device of the federation. (b) The resulting structure of a federation can automatically be optimized. (c) Even if a device gets lost, the remaining devices will be able to recognize their relation.
Legend: *solid line* - security relation represented by a certificate. *dashed line* - security relation implied by a certificate chain. *dotted line* - same certificate root. This is the weakest form of security relation.

2.2 Computational Complexity

The advantages of federating devices come on the expense of computational complexity. The expensive operations are key generation and pairing, often based on Diffie-Hellman key exchange. The ownership model makes also use of certificates. All these operations are executed at the initialization when a new device is added to a federation of devices. Rough estimates showed that these operations can be executed within a few seconds even on a smart-card - a time that might be tolerated during initialization because it has to be done once. Use of elliptic curve cryptography and pre-computation of keys can reduce the delay. However, the integration of weak devices require an individual adequate solution based on shared secrets, and therefore depending on other devices acting as proxy for them.

2.3 Exception Handling

With devices being mobile, it is likely that some devices get lost, stolen, or simply borrowed by other users. It is therefore important to avoid others to take over ownership, to alter security policies, or to integrate their devices into someone else's set of owned devices. Our approach to these problems is to require proximity of the user: the ability to assign ownership and defining policy updates is made dependent on the presence of dedicated devices such as, for example, the user's wrist watch. If no dedicated device is available, a pre-defined password for pairing is required. Recovery mechanisms make use of the redundant security relations among devices owned by the same user. The policy is used to propagate recovery lists within a federation and to re-assign certificates whenever a loss is noticed.

3 Security Policy

Every device has its own security policy to make access control decisions. The proposed security policy is expressed in the Security Policy Definition Language (SPDL, [8]). It does not only describe access rights, but also describes policies that allow remote configuration and authentic key exchange. This will support devices with a limited user interface and introduces security relations to other devices. Initially, policy updates are accepted only from devices involved in the ownership certificate chain. Policy updates can be used to define other devices that are allowed to send policy updates.

3.1 Security Policy Definition Language

SPDL evolved from an instance of the IETF policy core information model [9]. It describes principals (defined by credentials, typically the public keys serving as identity), services (defined by their service description), and constraints such as time, frequency, and network interface. These three elements are combined by logical expressions. The resulting logical expression is evaluated when an other device tries to access a service.

Logical expressions build a decision tree that have the advantage that decisions are definite (i.e., no precedence cases can appear), and that consecutive updates can be added anywhere in the decision tree.

Languages are often not easy to use because one has to remember syntax and semantic. One of the design goals of SPDL was to use it as a meta language that mediates between intuitive metaphors on the user interface (e.g., drag and drop a picture of a principal to a service logo) and configuration files of specific security protocols. As suitable user interfaces are not available on all devices, remote configuration of security policies is supported:

3.2 Letter of Authority

Device security policies expressed in SPDL are a sequence of policy updates applied to the default policy. The letter of authority is a signed policy update that allows authentic remote configuration of the device policy. It has two useful properties. First, the letter of authority allows to transport a key from one device to another in an authentic way, and second, allows to pass-on a policy update to the target device via other devices in case there is no direct connectivity.

3.3 Authentic Key Exchange

SPDL describes devices with credentials such as keys (i.e., the device identifier) or certificates for expressing roles that a device assumes. Instead of configuring these credentials in the policy, SPDL allows wildcards specifying the conditions to accept the credential of the next device that gets paired with the target device. The involved devices thus exchange their credentials themselves without requiring the user to cope with cryptographic material.

4 Delegation

During its lifetime, a device might change its owner, or it will be temporarily used by another user. Instead of resetting the device, we propose an extension to SPDL that allows the original owner of the device to delegate only specific functionality of the device. At the same time, the new user can make sure that the original user cannot continue to use just whatever functionality he wants. In some situations it might, however, still be useful to make use of selected security relations previously defined. A device manufacturer might, for example, define relations to authorized technicians that the owner of a device might activate upon need.

We achieve the flexibility of partial delegation by constraining parts of the old and new main policy with logical expressions (see Figure 3). The old policy is by default constrained by `and`-ing it with `false`, which can be relaxed to a specific service that is allowed. The new policy is constrained by `and`-ing it with either a list of services that are allowed, or simply `true` if all functionality is delegated. A tamperproof implementation is required for this form of delegation to guarantee that a user can only change and read policy elements he is in charge of.

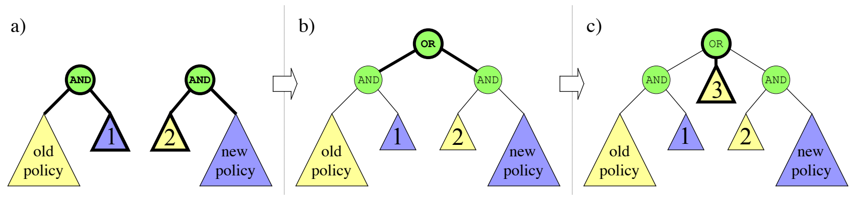

Fig. 3. (a) Logical expressions '1' and '2' block or specifically allow parts of the old and new main policy. (b) The old and new main policies are parts of one device policy. (c) A special policy element '3' defines a condition to terminate delegation.
Legend: *light color* - policy elements defined by the original owner of the device. *dark color* - policy elements defined by the new owner of the device.

5 Concluding Remarks

Most of the devices that we intend to connect will be personal and private devices of our daily life. A crucial aspect is therefore ease-of-use because the user cannot be expected to configure complex protocols. We consider the security bootstrapping as a first step towards the ambitious goal of *self-configuration* in dynamic networks. Having defined federations of trusted devices, many other configuration aspects might be easier to realise.

We have implemented the proposed mechanisms and were successful in hiding the complexity of key handling and security protocol configurations from the user [8]. Ongoing work extends the security bootstrapping with two additional important aspects, which are support for devices with very limited resources and practical implementations of pairing mechanisms: Limited devices such as sensors might not have the resources to provide advanced security mechanisms. We investigate techniques to off-load operations to more capable devices, for example devices acting as security proxy transparently providing access control and confidentiality towards other devices, and using adequate security mechanisms 'on the last hop'.

In this paper we presented a practical way of implementing key generation, authentication and distribution in dynamic environments without relying on central infrastructure: A key-pair is generated or pre-installed on the device and used as identifier of the device. These keys are authenticated during pairing, and distributed via signed policy update.

References

1. S.M. Bellovin and M. Merritt. Encrypted Key Exchange: Password based protocols secure against dictionary attacks. In *Proceedings 1992 IEEE Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society, 1992.
2. N. Asokan and P. Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23, 2000.
3. M. Cagalj and J-P. Hubaux. *Key agreement over a radioi link*. EPFL-IC Technical Report No. IC/2004/16, 2004.
4. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In B. Pfitzmann, editor, *Eurocrypt 2001*, number 2045 in Lecture Notes in Computer Science, pages 475–494. Springer-Verlag, 2001.
5. L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Proceedings Ubicomp 2001*, number 2201 in Lecture Notes in Computer Science, pages 116–122. Springer-Verlag, 2001.
6. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks. In M. Roe B. Christianson, B. Crispo, editor, *Security Protocols, 7th International Workshop Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 1999.
7. F. Stajano. The Resurrecting Duckling - what next? In M. Roe B. Christianson, B. Crispo, editor, *Security Protocols, 8th International Workshop Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 2000.
8. Ch. Rohner. *Security in Ad-hoc Distributed Systems.* PhD thesis, ETH Zürich, 2003.
9. B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. *Policy Core Information Model – Version 1 Specification.* IETF RFC 3060, February 2001.