The All-Or-Nothing Anti-Theft Policy—Theft Protection for Pervasive Computing

Jakob Illeborg Pagter*
The Alexandra Institute Ltd.
jakob.i.pagter@alexandra.dk

Michael Østergaard Pedersen†
University of Aarhus
michael@daimi.au.dk

Abstract

In many application scenarios for pervasive computing, theft is a serious security threat. In this paper we present the All-Or-Nothing anti-theft policy aimed at providing theft protection for pervasive computing.

The overall idea behind the All-Or-Nothing anti-theft policy is to chain devices together in friendly networks so that any device will only work when it can see all of its friends. Thus a thief will have to keep the network of friendly devices together even if he only desires to steal one of the devices. Otherwise the device will not work.

We provide a detailed security policy, present the required cryptographic protocols, provide three different applications, and finally we document that the policy is suitable for implementation on typical pervasive computing devices.

1 Introduction

Recent studies indicate that one of the major security issues for personal pervasive computing devices is that they get stolen or users forget them in cabs (see e.g. [1]), meeting rooms etc. Another security issue in pervasive computing is related to high-end entertainments systems (TVs, hi-fi equipment, etc.) which are often the target of systematic burglary.

In this paper we present a security policy called the All-Or-Nothing security policy, which may help remedy this situation. We describe how this policy can be realised, provide sample applications and documentation that our solution is realisable on commercial pervasive computing platforms such as cell phones. A benefit of this policy is that it also provides distribution of cryptographic key material, based on which a security architecture for providing confidentiality or authenticity can be leveraged.

We would like to point out that this paper focuses on key distribution and anti-theft in a network of devices belonging to a single user. Clearly there are many other issues related to this area such as usability, thorough performance evaluation, configuration of devices, management of the network, etc. These issues are not addressed in this paper.

1.1 Related work

Anti-theft policies and solutions are not widespread in the literature, maybe because designers are afraid to disclose the workings of their mechanisms. One recent contribution in the literature is by Droz et al. [7] who describe a system relying on credits and blacklists, with each protected device periodically requesting new credit in order to continue operating. Our system is different in that we protect a system of devices, not a single device, and the solution of [7] rely on a central server whereas our solution is highly de-centralised. Another source on anti-theft is Ross Anderson's book on Security Engineering [2] which contain the description of various solutions (not only for anti-theft) and their pros and cons. Wired Magazine also recently carried a more popular, but highly enjoyable and relevant, read on car theft [9].

Our security policy is described as an extension of the Resurrecting Duckling security policy model by Stajano and Anderson [8], which has become a standard template for handling secure device discovery in ad-hoc networks.

Finally, the realisation of the All-Or-Nothing security policy is partially based upon classical symmetric protocols [4], which we simply employ in this "new" setting as they provide a secure, simple, and elegant basis for the realisation of the policy.

*Supported in part by ISIS Katrinebjerg Software (http://www.isis.alexandra.dk/software).

†Supported by the eu-DOMAIN IST EU project, contract no. IST-004420.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

2 The All-Or-Nothing Anti-Theft Policy

2.1 Basic Principle

Historically the primary security threat against many types of devices (digital or not) has been theft. With the digitalisation of such systems a number of new threats such as data disclosure, identity theft, etc. also emerges as serious problems, but in this paper we focus on the old-fashioned theft of the physical devices in the realm of pervasive computing.

The starting point for our work is the vision of pervasive computing by Mark Weiser [11], where people will be surrounded by devices with embedded computers. Pervasive computing devices owned by a particular person will often have (at least) three interesting properties related to our work in this paper: 1) they are the target of theft, 2) they have embedded computers, and 3) there are many devices owned by the same person.

Our overall idea is to use the embedded computers to cryptographically chain these many devices owned by the same person together in a manner which forces a thief to steal *all* of the devices belonging to a particular person, if the thief is to get the desired functionality from these devices.

What this basically means is that a user will enrol all of his devices into a network of friendly devices, and any device on this network will only operate with full functionality if it can "see" all (or most) of its friends.

This leads us to the governing principle of our security policy:

- **All-Or-Nothing It must be "sufficiently" hard to use stolen equipment in any other network configuration than the one it was in when stolen.**

Of course, any thief succeeding in stealing all of the devices in a network will have stolen a network of fully functional devices. But, according to the All-Or-Nothing principle the thief cannot (well, it is hard) de-compose the network or revoke devices from it. I.e., if the thief is to be successful he must steal all devices and keep them together, even when he fences them off to some "customer" who in turn must also keep all the devices together to keep them operational.

2.2 The All-Or-Nothing Security Policy

Realising the All-Or-Nothing policy means that a device should be unable to function properly when it is removed from its friends. Of course this might not always be desirable in a strict sense, since every device would have to be available all the time for this to work. So, we are satisfied with just having some subset of size t of all devices in the system available.

Now we describe our security policy as an extension of the resurrecting duckling model [8]. At any given point of time we call the number of devices n.

State. Each device, say D_0, is in one of three states:

Imprintable. In this state it is promiscuous and will associate to any network of devices to which it is presented.

Imprinted. In this state the device is chained to a network of friendly devices: $D_1, ..., D_{n-1}$.
An imprinted device will occasionally, at least when imprinting another device and when performing an operation like power on, verify the presence of the other members of the network. If it fails to verify this presence it will move to the Emergency state.

Emergency. In this state the device will perform some set of application specific actions to recover. Following these actions, the device will either return to the imprinted or imprintable state. The latter, imprintable, state only(!) following user authentication towards the device, as this amounts to performing death (see below).

Imprinting. The user authenticates to some device, D_i, which then associates with the new device, D_0. Afterwards, D_0 will automatically associate with the remaining devices in the network.

Death. The user authenticates to the device being killed, and the device reverts to the imprintable state.

Assassination. It must be "sufficiently" hard to subvert a device from the imprinted to the imprintable state without authenticating towards the device.

From the above we see that security mechanisms realising this policy must address the following issues:

- Protocol for device association in the imprinting phase.
- Protocol for presence verification.
- Choose a threshold t.
- User authentication towards devices.
- Frequency of presence verification.
- Emergency rules.

We describe general application-independent protocols for device association and presence verification, but leave details on thresholds, user authentication, frequency of presence verification and emergency rules to the specific applications. With respect to user authentication, this must technically result in the device being present with a PIN. Whether this PIN is directly entered, read from a biometric scanner, entered through another device, etc., must be decided for each specific application.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

Note also that a full real-life security architecture must probably also address other attacks than theft. Standard security goals such as confidentiality, etc. may also be of relevance. We do not address these issues in this paper, but note that our solution presented is based on equipping all devices with cryptographic keys, which can of course have many uses besides the ones described in this paper.

2.3 Protocol for device association

The system consists of devices $D_1, ..., D_{n-1}$ and the new device to be imprinted will be called D_0.

Imprinting takes place in three phases: First the user transfers a master PIN to D_0 and one other device on the network. We assume without loss of generality that this device is D_1. Second, D_0 and D_1 establish a shared key using a password-based key exchange protocol with PIN as the shared secret. Third, D_0 runs a protocol with the remaining devices to obtain shared keys with them.

Let H be a cryptographic hash function and let D_i denote the identity of a device, such as the hardware address. We assume that the system is in a state where devices have shared keys with every other device, so K_{ij} denotes the key shared by D_i and D_j which of course is similar to K_{ji}. Furthermore, device D_i stores $H(\textit{PIN}\|D_i)$, where $\|$ denotes concatenation.

We make use of two primitives: *authenticated ping* and *presence verification*. Authenticated ping is a symmetric two-way authentication protocol to allow two devices to prove to each other that they know a shared secret key. Presence verification is simply authenticated ping from one device to t other devices, to verify that the device has not been removed from the system. Both are described in more details in section 2.4.

Whenever we say that something is verified, we assume that the device will abort the protocol, and forget all data received during the protocol if verification fails.

The first phase is the user phase:

1. The user transfers *PIN* to D_1
2. D_1 verifies *PIN* using $H(\textit{PIN}\|D_1)$
3. D_1 performs *presence verification*
4. The user transfers *PIN* to D_0

The second phase is the key exchange phase. Here D_0 and D_1 exchange a shared key:

1. D_0 and D_1 perform a password-based key exchange with *PIN* as the secret, resulting in K_{01}.
2. D_0 runs *authenticated ping* with D_1 to verify K_{01}
3. D_1 accepts D_0 as a new device in the network
4. D_1 sends $E_{K_{01}}(H(K_{1i}\|D_0))$ to D_0 for all $2 \leq i < n$
5. D_1 forgets *PIN*
6. D_0 computes $H(\textit{PIN}\|D_i)$ for all $2 \leq i < n$ and forgets *PIN*

The third phase is the key distribution phase, and is executed between D_0 and every other device $D_2, ..., D_{n-1}$. Here we show the execution between D_0 and D_2

1. D_0 tells D_2 that D_1 has granted it access
2. D_2 perform *presence verification*
3. D_0 and D_2 performs a password-based key exchange with $H(K_{12}\|D_0)$ as the secret, resulting in K_{02}.
4. D_0 runs *authenticated ping* with D_2 to verify K_{02}
5. D_0 sends $E_{K_{02}}(H(\textit{PIN}\|D_2))$ to D_2
6. D_2 verifies the value above and if correct, accepts D_0 as a new device in the network
7. D_0 forgets $H(K_{12}\|D_0)$

We note that in case some devices are offline during the key distribution phase, D_0 can store the value $H(K_{1i}\|D_0)$ until D_i becomes available. Finally in this case E_K denotes both encryption and authentication under key K, for example by performing encryption in GCM [10] mode.

2.4 Protocol for presence verification

In order to perform an important operation, a device will require permission from t other devices to make sure that these devices are still around. We call this presence verification with threshold t. If t devices are present, this is a very good indication that the device has not been removed from the system and the operation is allowed.

Call the device that wants to perform the operation D_0 and another device in the system D_1. The following protocol, authenticated ping, now takes place between D_0 and D_1

1. D_0 says hello to D_1
2. D_1 sends a nonce N_1 to D_0
3. D_0 replies with $E(\{D_1 \oplus N_1\} \oplus E(N_0 \oplus E(N_1))), N_0$
4. D_1 verifies the value from the previous step
5. D_1 replies with $E(N_0 \oplus E(N_1))$
6. D_0 verifies the value from the previous step

The protocol is a secure two-way authentication protocol from [4]. E is a symmetric encryption function in ECB mode using the key shared between D_0 and D_1 and $\oplus$ denotes XOR. After completing this protocol with t devices, D_0 will assume that it has not been removed from the system, and will continue to operate normally.

2.5 Security Analysis

The goal of the adversary is to make stolen devices function as if they were still part of the system, or to transform

a stolen imprinted device into the imprintable state, which would allow him to add the device to a new system. In order to do this he might choose to compromise some devices and use the secrets stored on them to mount the attack.

It is trivial to observe that if an adversary can compromise a device, that device can be removed from the system and still maintain full functionality. Another trivial observation is that if the adversary gets access to the PIN he has full control over the system. What the system does ensure is that short obtaining the PIN from the user, or using brute-force to guess it using data from a compromised device, the only feasible attack seems to involve compromising either many devices or the device the adversary wants to steal. The difficulty in compromising a device depends on how many resources are spent securing the device, but even many cheap devices can help protect a few expensive devices in the system from theft.

Another type of attack against such authentication systems is the relay attack, also known as the mafia fraud attack. In this attack an adversary would steal a device and install a proxy device in the network. This device would simply relay communication between the network and the stolen device over a long distance, meaning that the stolen device could always perform a valid authenticated ping. While the relay attack is devastating in some scenarios, for theft protection it does not seem feasible. We will discuss the implications of this attack in the scenarios in section 3.

Due to space constraints a more thorough security analysis has been omitted, and we refer to the full version of this paper.

3 Three Applications

3.1 Entertainment Systems

Theft is a huge problem, especially for high value products such as modern entertainment systems. There probably exist sophisticated burglar rings gathering intelligence on what equipment can be stolen where and use this knowledge to obtain, on demand, whatever items their "customers" have on their shopping list. Such intelligence could come from insiders at stores that sell such equipment, burglars can drive around residential areas at night with product specific remote controls to find out where certain (types of) devices can be found, etc.

Of course it is impossible to completely prevent theft, hence the strategy often employed is to minimise the value of stolen equipment by maximising the efforts needed to get stolen equipment to function as if it was acquired legitimately. In other words, our security goal is to make using stolen devices as troublesome as possible, which in the context of the All-Or-Nothing policy translates to forcing a thief to steal and fence entire systems rather than single devices, or to spend more resources trying to compromise a device, than what the device is worth.

Some special features of devices in such systems are that they are not mobile, but they may not all have specific input devices (e.g. loudspeakers). We shall assume that the devices have plenty of computing power (comparable to, say, a cheap PC), are networked (i.e., communicating via a digital bus), and that all devices are equipped with an USB-port.

As we have a relatively static set of non-mobile devices we shall set the threshold to be the total number of devices minus one, $t = n - 1$. I.e., any device must be able to see all other devices[1].

User authentication will be based on the user having a special USB PIN-pad. This will be connected to the relevant devices when PINs are to be entered. We imagine that the user receives this low cost unit when he purchases his first device.

The frequency of presence verification is defined so that in addition to when doing imprinting, we perform presence verification whenever a device has been without electricity. The reason is that any thief must unplug a device to steal it.

Finally, we define the following emergency rules. 1) If presence verification fails the device will request user authentication to perform death. If this fails, the system will shutdown for a period of time (this period of time should increase for each failure), after which the device will again request user authentication to perform death. 2) If the remaining devices reappear the device will also require user authentication with the same increasing shutdown period, before the device can return to the imprinted state and normal operation can be resumed.

In this manner, all devices must be stolen simultaneously without any interruption in the power supply if the thief is to have any functionality.

In this scenario, the relay attack does not seem very likely as the owner of the devices will know immediately that devices have been stolen. If some devices have been stolen, the remaining devices should not work anymore. If they do it is a sure sign that something is wrong. We note that even if one device is stolen, the user can still perform death on the remaining devices and add them to a new system.

3.2 Personal Devices

Recently Sony Ericsson released a Bluetooth watch which can control cell phones and show information such as caller ID. Further, you may configure the watch to issue an alarm if it looses connection to the cell phone.

Devices such as cell phones or Bluetooth enabled watches cannot be assumed to have plenty of computing

[1] To allow the owner to take single devices to service etc., one could perhaps work with a threshold of $t = n - 2$ or $t = n - 3$.

power, but as our implementation experiments show (see section 4), they have sufficient power for our needs. In contrast to entertainment systems they are highly mobile, albeit always close to a particular person and thus close to each other. Also, we will assume that devices either have input devices themselves or can be configured over Bluetooth from another device.

In this case we focus on a user with just a few devices (such as a watch, a cell phone, and a PDA, $n = 3$) and set the threshold to $t = 1$. This means that for a device to successfully perform presence verification, at least one other device must be present.

User authentication is done by directly entering a PIN on each device. This is standard for devices such as cell phones and PDAs. For watches and similar, user authentication could be done by having the user enter the PIN through, say, the cell phone, over Bluetooth, which can then be displayed on the watch and acknowledged through the push of a button. At first this approach may seem insecure, but done in the privacy of your home there is little practical risk involved: if we can make sure that the right PIN is entered, it seems unlikely that a pickpocket or another will be monitoring Bluetooth traffic in your home.

In this case we choose a high frequency for presence verification, as you want to know immediately if you have forgotten your PDA in a taxi, or someone has taken your cell phone from your pocket.

We define the following emergency rules: 1) If a device in the emergency state is able to perform presence verification it returns to the imprinted state and resumes normal operation. 2) Otherwise death must be performed by entering the PIN. We have two alternative rules if the user is not able to supply the correct PIN. Either, the devices will shutdown for increased periods of time as suggested for entertainment systems, or after a fixed number of failures the devices will erase all data and become imprintable or locked. The latter rule should preferably be combined with a sound back-up policy. It might also be made more strict for devices containing sensitive information, e.g. classified business data, in which case data could be erased immediately when entering the emergency state.

One benefit of employing the All-Or-Nothing policy in this scenario is that a user might not need to protect his cell phone or PDA with a PIN as such, i.e. having to enter the PIN each time he desires to use the device. Of course, this leaves the user vulnerable to relay attack performed by a sophisticated pickpocket who simultaneously removes your cell phone while leaving a relay device in your pocket (this of course is not a problem if the threat model is primarily for forgotten devices). Still, as users often do not use the PIN features in cell phone (see e.g. Dourish et al. [6]), the All-Or-Nothing might still provide better realised security than the PIN solution.

3.3 Cars

Our final example is theft protection for cars. As described in [9] there may be multiple back doors and attack opportunities for the social engineer. Further, realising the assassination principle, i.e., founding security in a tamper resistant computing base is notoriously hard [3]. The All-Or-Nothing policy does not specify how to deal with these issues, so we assume that key material etc. is handled diligently and is sufficiently well protected.

Car theft protection is often referred to as immobiliser as they make the car immobile. A widespread current technology is the Digital Signature Transponder (DST) proved cryptographically insecure due to weak proprietary algorithms and too short a key length by Bono et al. [5]. The solution we propose will behave very similarly to the DST solution, but with our algorithms available for public scrutiny.

An obvious solution would be to simply enrol the car in the network of personal devices described above. If there is more than one user of the car, we can extend the policy to allow for multiple users of each device.

Alternatively, and perhaps more suited to the use of most drivers, we could root the policy in the car and the car keys. The threshold would be $t = 1$, i.e., both the car and the car key (assuming that there is only one key) must be present. The user authentication would be done by the manufacturer, perhaps with a possibility for the car owner to choose a new PIN (and new keys) using a special purpose device as the one used for entertainment systems. Other that empowering the user to provide his devices with new keys, this might come in handy if the car key is stolen and you want to make sure that the thief cannot later steal the car as well, in which case you perform death on the car and imprint with a new (car) key.

With respect to frequency and emergency rules, this instantiation of the policy is a bit special. The reason is that the car key is more or less passive. Fortunately it is not the key we wish to protect. The car will verify presence whenever someone is attempting to turn the car on. The emergency rules are simple, the car requires user authentication to perform death, but will automatically revert to normal operation whenever presence is reestablished.

In this way the car will actually be in the emergency state much of the time (whenever the key is not in the car).

The relay attack is not unrealisable in this scenario. A sketch of an attack could be to drop a relay device in the pocket of the car owner. However, when the car owner finds out the car is stolen he could perform death on the key which will immobilise the car. So, the All-Or-Nothing policy does not prevent car theft as such, but may prevent a car thief from actually using the car once the theft is detected.

4 Implementation

The purpose of our implementation efforts was to see whether or not this idea is realisable in practise, not to implement the full scheme. We chose to implement authenticated ping since it is a building block used in every other part of the scheme, and performance-wise none of the other protocols seem much harder to execute, so we believe it is a good indication of the performance of the complete scheme.

We have chosen to use motes for our implementation as they represent one of the more resource constrained platforms which might form the basis for some of our proposed applications. So, we argue that if our solution is viable on this platform, it also is on more powerful ones. We use TMote Sky motes from Moteiv which are commercially available.

The hardware was four TMote Sky, equipped with a 16-bit 8MHz TI MSP430 CPU, 10kb of RAM, 48kb of flash memory and a 250kbps 2.4GHz IEEE 802.15.4 wireless transceiver. They have three LEDs and two buttons, where one is the reset button. Development was done on TinyOS 1.x which takes care of mesh networking and communication. The programming language was NesC.

We implemented authenticated ping on the motes with a threshold of $t = 2$, meaning that if one of the motes was down, or communication failed for some reason, the ping would still succeed. Since we did not implement any key exchange, the encryption keys were hard-coded at development time. For symmetric encryption we implemented the XTEA algorithm with 32 rounds. XTEA uses 128-bit keys and 64-bit blocks, which had to be taken into account, since the maximum number of bytes TinyOS supports in a message is 29.

When the user button was pressed, the mote would perform presence verification and the green LED would light up if it succeeded, otherwise the red LED would light up. Since the motes cannot generate random data we used a PRNG seeded with the mote's address, which does not provide cryptographic random data, but was fine enough for our proof of concept. Another problem was related to interrupt handling on the motes which caused messages to be dropped if two messages arrived while the mote was performing some computation. We circumvented this problem by inserting a small fixed delay between each attempted authenticated ping giving the previous one time to finish before starting the next.

The binary uploaded to the mote is around 10kb, including everything (TinyOS, our application, libraries, etc) and uses around 500 bytes of RAM when running. Since the priority was just to get it running, there is room for improvement. Especially the code size can be made smaller, but also the RAM usage. A rough estimate is that the authenticated ping takes around 10ms to perform, so for all practical purposes presence verification succeeds immediately when the button is pressed.

5 Conclusions and Future Work

In this paper we have presented the All-Or-Nothing anti-theft policy, including a detailed security policy, the required cryptographic protocols, three different applications, and finally documentation that the policy is suitable for implementation on typical pervasive computing devices.

In the future it would be nice to have a full implementation of the protocols in a sample application to study properties of performance and usability, as we reckon that especially usability will be the deciding factor as to whether or not this idea is realisable.

References

[1] Taxis Hailed as Black Hole for Lost Cell Phones and PDAs, as Confidential Data Gets Taken for a Ride, 06. http://www.pointsec.com/news/newsreleases/release.cfm?PressId=386.

[2] R. Anderson. *Security Engineering—A Guide to Building Dependable Distributed Systems*. Wiley, 2001. `http://www.cl.cam.ac.uk/~rja14/book.html`.

[3] R. Anderson and M. Kuhn. Tamper Resistance—a Cautionary Note. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, November 1996.

[4] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. *Lecture Notes in Computer Science*, 576, 1991.

[5] S. C. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. In *Proceedings of the 14th Usenix Security Symposium*, 2005.

[6] P. Dourish, R. E. Grinter, J. D. de la Flor, and M. Joseph. Security in the wild: user´strategies for managing security as an everyday, practical problem. *Personal and Ubiquitous Computing*, 8, 2004.

[7] P. Droz, C. Gülcü, and R. Haas. Wanted: A theft-deterrent solution for the pervasive computing world.

[8] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of 7th International Workshop on Security Protocols*, 1999.

[9] B. Stone. Pinch My Ride. `http://www.wired.com/wired/archive/14.08/carkey.html`, August 2006. Wired Magazine.

[10] J. Viega and D. McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106 (Proposed Standard), June 2005.

[11] M. Weiser. Ubiquitous computing. `http://www.ubiq.com/hypertext/weiser/UbiHome.html`.

0-7695-2847-3/07 $20.00