# The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks

Frank Stajano[1 2] and Ross Anderson[1]

[1] University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
*name.surname*@cl.cam.ac.uk
[2] AT&T Laboratories Cambridge,
24a Trumpington Street, Cambridge CB2 1QA, UK
fstajano@uk.research.att.com

**Abstract.** In the near future, many personal electronic devices will be able to communicate with each other over a short range wireless channel. We investigate the principal security issues for such an environment. Our discussion is based on the concrete example of a thermometer that makes its readings available to other nodes over the air. Some lessons learned from this example appear to be quite general to ad-hoc networks, and rather different from what we have come to expect in more conventional systems: denial of service, the goals of authentication, and the problems of naming all need re-examination. We present the *resurrecting duckling* security policy model, which describes secure transient association of a device with multiple serialised owners.

## 1 Introduction

The established trend in consumer electronics is to embed a microprocessor in everything—cellphones, car stereos, televisions, VCRs, watches, GPS (Global Positioning System) receivers, digital cameras—to the point that most users have already lost track of the number of items they own that contain one. In some specific environments such as avionics, electronic devices are already becoming networked; in others, work is underway. Medical device manufacturers want instruments such as thermometers, heart monitors and blood oxygen meters to report to a nursing station; consumer electronics makers are promoting the Firewire standard [8] for PCs, stereos, TVs and DVD players to talk to each other; and kitchen appliance vendors envisage a future in which the oven will talk to the fridge, which will reorder food over the net.

We envisage that, in the near future, this networking will become much more general. The next step is to embed a short range wireless transceiver into everything; then many gadgets can become more useful and effective by communicating and cooperating with each other. A camera, for example, might obtain the geographical position and exact time from a nearby GPS unit every time a picture is taken, and record that information with the image. At present, if the

photographer wants to record a voice note with the picture, the camera must incorporate digital audio hardware; in the future, the camera might let him speak into his digital audio recorder or cellphone. Each device, by becoming a network node, may take advantage of the services offered by other nearby devices instead of having to duplicate their functionality.

### 1.1 Ad-hoc Wireless Networks

This vision of embeddable wireless connectivity has been in development for several years at AT&T Laboratories Cambridge in the context of the Piconet [4] project and is also being pursued, although with emphasis on different aspects, by several other groups including HomeRF [6,12], IrDA [3] (which uses infrared instead of radio) and Bluetooth [13,7].

Everyone—including potential users—knows that wireless networking is more prone to passive eavesdropping attacks. But it would be highly misleading to take this as the only, or even the main, security concern.

In this paper we investigate the security issues of an environment characterised by the presence of many principals acting as network peers in intermittent contact with each other. To base the discussion on a concrete example we shall consider a wireless temperature sensor. Nearby nodes may be authorised to request the current temperature, or to register a "watch" that will cause the thermometer to send out a reading when the temperature enters a specific range. We wish to make our thermometer useful in the widest range of environments including environmental monitoring, industrial process control and medicine.

We will therefore consider how we can enable our thermometer to support all the security properties that might be required, including *confidentiality*, *integrity* (and its close relative *authenticity*) and *availability*. Contrary to academic tradition, however, we shall examine them in the opposite order, as this often (and certainly in our case) reflects their actual importance. First, however, we have to mention some of the resource constraints under which such networks operate.

### 1.2 System constraints

The three main constraints on Piconet, and on similar systems which support ad-hoc networks of battery operated personal devices, are as follows:

**Peanut CPU:** the computing power of the processor in the node is typically small, so large computations are slow.

**Battery power:** the total energy available to the node is a scarce resource. The node likes to go to sleep whenever possible. It is not desirable to use idle time to perform large computations in the background.

**High latency:** to conserve power, nodes are off most of the time and only turn on their receiver periodically. Communicating with such nodes involves waiting until they next wake up.

The consequence of those constraints is that, while strong symmetric cryptography is feasible, modular arithmetic is difficult and so is strong asymmetric cryptography. Where a peanut node (e.g. the one embedded in a camera) interacts with a more powerful one (e.g. the one embedded in a mobile phone or laptop), one may use techniques such as low exponent RSA, with the protocols designed so that the peanut node sticks to the cheap operations of encryption and verification while avoiding the expensive ones of decryption and signature.

More generally, where there is a trade-off between security and (say) battery life, we may want to let the user control this. For example, if our thermometer is used to drive a wall display in someone's living room that shows the outside temperature, then the owner is unlikely to opt for validated and encrypted communication if this means that he must change the battery every month instead of once a year.

One challenge is to integrate this flexibility in the system without introducing major architectural holes of the sort that would allow the attacker, too, to turn off security at will.

## 2 Availability

*Availability* means ensuring that the service offered by the node will be available to its users when expected. In most non-military scenarios, this is the security property of greatest relevance for the user. All else counts little if the device cannot do what it should.

### 2.1 Radio jamming

In the traditional threat model—derived from the military—an attacker can deny service to the nodes in a given area by jamming the radio frequencies they use. Traditional defences include spread spectrum and frequency hopping, both of which force the attacker to jam a wider frequency band and thus use more power. We will revisit them briefly in section 5 below. However such concerns are of less relevance to the commercial world, where such attacks are dealt with by complaining to the authorities and having the operator of the jamming station arrested.

The novel and interesting service denial threat is different, and concerns battery exhaustion.

### 2.2 Battery exhaustion

A malicious user may interact with a node in an otherwise legitimate way, but for no other purpose than to consume its battery energy. Battery life is the critical parameter for many portable devices, and many techniques are used to maximise it; in Piconet, for example, nodes try to spend most of the time in a sleep mode in which they only listen for radio signals once in a while (the period can be set from a few seconds to several minutes). In this environment, power

exhaustion attacks are a real threat, and are much more powerful than better known denial of service threats such as CPU exhaustion; once the battery runs out the attacker can stop and walk away, leaving the victim disabled. We call this technique the **sleep deprivation torture** attack.

For any public access server, there is necessarily a tension between the contrasting goals of being useful to unknown users and not succumbing to vandals. Whereas some applications can restrict access to known principals, in others (such as web servers and name servers) this is infeasible since the very usefulness of the service comes from its being universally available.

If a server has a primary function (such as sending the outside temperature to the meteorological office every hour) and a distinct auxiliary function (such as sending the current temperature to anyone who requests it) then these functions can be prioritised; a reservation mechanism can ensure that the higher priority use receives a guaranteed share of the resource regardless of the number of requests generated by the lower priority uses. (The highest priority use of all may be battery management: if one can estimate fairly accurately the amount of usable energy remaining, then the service can be monitored and managed provided that the process does not itself consume too much of the resource it is intended to conserve.)

## 3 Authenticity

### 3.1 To whom can a principal talk?

In some applications our thermometer will broadcast its temperature readings, but in general it will only send them to recipients who have been authorised in some way. For example, in a hospital, it might be authorised to send temperature readings to any doctor's palmtop computer or any nursing station. But it might also be required to restrict transmission (e.g. of the temperature of a celebrity) to a particular station or device.

The usual authorisation mechanisms (which turn out to be the least interesting in this case) involve a centralised system administrator. This may be implemented as access control lists (the administrator tells the thermometer who is authorised) or capabilities (the administrator gives some principals a signed certificate which they present to the thermometer when they want a reading). However, the ad-hoc network environment poses a fundamental new problem: the *absence of an online server*.

Interactions with the administrator after the thermometer has been manufactured (or personalised by the institution that owns it) may be expensive or time-consuming, as they may entail establishing a network connection to a central server (perhaps using gossiping via intermediate nodes), or bringing a management device physically close to each affected node. In the particular case of a thermometer, the device might be calibrated every six months, at which time new security state can be loaded; however, rapid changes may be too expensive for a central administrator to make regularly.

It follows that the length of any validity period (whether for certificates, access control lists or capabilities) will be a trade-off between timeliness and convenience. But relying on expiration dates imposes on the nodes the extra cost of running a secure clock—otherwise the holder of an expired certificate might reset a node's clock to a time within the validity period. As many Piconet nodes would not normally have an onboard clock, the classical approach to authentication is suspect. Thankfully, there is a better way.

### 3.2 Secure transient association

The novel and interesting authentication problem in ad-hoc networks of wireless devices is that of **secure transient association**. If a householder owns a device, say a universal remote control, that lets her control various other devices in her home (such as hi-fi and television components, the heating system, lights, curtains and even the locks and burglar alarm) then she will need to ensure that a new device she buys from the shop will obey her commands, and not her neighbour's. She will want to be assured that a burglar cannot take over the heat sensing floodlight in the garden, or unlock the back door, just by sending it a command from a remote control bought in the same shop.

As well as being *secure* (whatever that means), the association between the controller and the peripheral must also be *transient*. When a householder resells or gives away her television set or hi-fi or fridge, it will have to obey another controller; when her controller breaks down (or she decides to replace it or upgrade its operating system), she must be able to regain control of all the gadgets she already owns.

A central authentication service is possible for expensive consumer durables; most governments run such services for houses and cars. But there is no prospect that this will be extended to all durable consumer goods; the UK government abandoned dog licensing some years ago as uneconomic. In any case, there would be very grave civil liberties objections to the government maintaining lists of all PCs, hifis and DVD players in the country; the outcry over the Pentium III processor ID indicates the likely level of political resistance. Even the existing central services stop short of managing keys; the replacement of car keys is left to the motor trade, while house locks are completely uncontrolled. So it is desirable that key management be performed locally: the last thing we want is to impose an expensive and unpopular central solution. Yet it would be nice if we could still provide some means of making a stolen DVD player harder to resell.

Another insight comes from scenarios where we have a pool of identical devices, such as a bowl of disinfectant containing ten thermometers. The doctor does not really care which thermometer she gets when she picks one up, but she does care that the one her palmtop talks to is the same one she is holding and not any other one in the bowl or nearby in the ward.

Many more potential applications of wireless devices require establishing a secure transient association between two principals (typically, but not necessarily, a user and a peripheral). For example, there has been significant interest in the possibility of a police pistol that can only fire when held by the officer to

whom it was issued, who for this purpose might be wearing a very short range radio ring: at present, in the USA, a large number of the firearm injuries sustained by policemen come from stolen police guns. Similar considerations might apply to more substantial weapon systems, such as artillery, that might fall into enemy hands.

### 3.3 The "resurrecting duckling" security policy

A metaphor inspired by biology will help us describe the behaviour of a device that properly implements secure transient association.

As Konrad Lorenz beautifully narrates [10], a duckling emerging from its egg will recognise as its mother the first moving object it sees that makes a sound, regardless of what it looks like: this phenomenon is called **imprinting**. Similarly, our device (whose egg is the shrink-wrapped box that encloses it as it comes out of the factory) will recognise as its owner the first entity that sends it a secret key. As soon as this 'ignition key' is received, the device is no longer a newborn and will stay faithful to its owner for the rest of its life. If several entities are present at the device's birth, then the first one that sends it a key becomes the owner: to use another biological metaphor, only the first sperm gets to fertilise the egg.

We can view the hardware of the device as the body, and the software (particularly the state) as the soul. As long as the soul stays in the body, the duckling remains alive and bound to the same mother to which it was imprinted. But this bond is broken by death: thereupon, the soul dissolves and the body returns in its pre-birth state, with the resurrecting duckling ready for another imprinting that will start a new life with another soul. Death is the only event that returns a live device to the pre-birth state in which it will accept an imprinting. We call this process **reverse metempsychosis**. Metempsychosis refers to the transmigration of souls as proposed in a number of religions; our policy is the reverse of this as, rather than a single soul inhabiting a succession of bodies, we have a single body inhabited by a succession of souls[1].

With some devices, death can be designed to follow an identifiable transaction: our medical thermometer can be designed to die (and lose its memory of the previous key and patient) when returned to the bowl of disinfectant. With others, we can arrange a simple timeout, so that the duckling dies of old age. With other devices (and particularly those liable to be stolen) we will arrange that the duckling will only die when so instructed by its mother: thus only the currently authorised user may transfer control of the device. In order to enforce this, some level of tamper resistance will be required: assassinating the duckling without damaging its body should be made suitably difficult and expensive.

In some applications we may need to be able to recover from circumstances in which the legitimate user loses the shared secret (e.g. the password is forgotten

[1] Prior art on this technique includes Larry Niven's science fiction novel *A World Out of Time* (1977) in which convicted criminals have their personalities "wiped" and their bodies recycled.

or the remote control is broken beyond repair). To be able to regain control of the duckling, one should allow for **escrowed seppuku**: someone other than the mother, such as the manufacturer, holds the role of Shōgun with a master password that can command the device to commit suicide.

In other applications, only part of the duckling's soul should perish. In fact, our thermometer will typically be calibrated every six months by the hospital's (or manufacturer's) technician, and the calibration information must not be erased along with the patient data and user key when the device is disinfected, but only when it is plugged into a calibration station. So we may consider the device to be endowed with two souls—the calibration state and the user state—and a rule that the latter may not influence the former. So our resurrecting duckling security policy may be combined with multilevel security concepts (in fact, "multilevel secure souls" are a neat application of the Biba integrity policy model [5]).

### 3.4 Imprinting

During the imprinting phase, as we said, a shared secret is established between the duckling and the mother. Again, we might think that this is easy to do. If at least one of the two principals involved can perform the expensive public key operations (decrypt and sign), the other device then simply generates a random secret and encrypts it under the public key of the powerful device from which it gets back a signed confirmation.

But many of our nodes lack the ability to do public key, and even if they did it would still not help much. Suppose that a doctor picks up a thermometer and tries to get his palmtop to do a Diffie-Hellman key exchange with it over the air. How can he be sure that the key has been established with the right thermometer? If both devices have screens, then a hash of the key might be displayed and verified manually; but this is bad engineering as it is both tedious and error-prone, and in an environment where we want neither. We are not likely to want to give a screen to every device; after all, sharing peripherals is one of the goals of ad-hoc networking.

In many applications, there will only be one satisfactory solution, and we advocate its use generally as it is effective, cheap and simple: physical contact. When the device is in the pre-birth state, simply touching it with an electrical contact that transfers the bits of a shared secret constitutes the imprinting. No cryptography is involved, since the secret is transmitted in plaintext, and there is no ambiguity about which two entities are involved in the binding.

Note that an imprinted duckling may still *interact* with principals other than its mother—it just cannot be *controlled* by them. In our medical application, we would usually want the thermometer to report the patient's temperature to any device in the ward which asked for it. Only in exceptional circumstances (such as a celebrity patient, or a patient with a socially stigmatised condition) would the patient require encrypted communications to a single doctor's PDA. So should we also have an option of imprinting the device with a cleartext access control list (and perhaps the patient's name), rather than an ignition key?

This brings us back to the issue raised at the end of section 1.2, namely how we might enable a single device to support security mechanisms of differing strength. The solution that we favour is to always bootstrap by establishing a shared secret and to use strong cryptography to download more specific policies into the node. The mother can always send the duckling an access control list or whatever in a message protected by the shared secret. Having a key in place means that the mother can change its mind later; so if the patient is diagnosed HIV positive and requests secure handling of his data from then on, the doctor does not have to kill and reinitialise all the equipment at his bedside. In general, it appears sound policy to delegate from a position of strength.

## 4 Integrity

So far we have seen that denial of service, the goals of authentication, and the mechanisms for identifying other principals are surprisingly different in an ad-hoc network. Is there any role for the more conventional computer security mechanisms? The answer appears to be a qualified yes when we look at integrity.

*Integrity* means ensuring that the node has not been maliciously altered. The recipient wants to be sure that the measurements come from the genuine thermometer and not from a node that has been modified to send out incorrect temperature values (maybe so as to disrupt the operation of the recipient's nuclear power plant).

### 4.1 If you can't afford signatures...

Prudence dictates that a patient's temperature should only be measured by a "known good" thermometer, such as one that passed a calibration inspection within the last six months. So it is natural for calibrators to issue signed dated certificates (though some care must be taken if some of the thermometer's prospective clients do not possess a clock). But the certificate could have been replayed by a middleman. What sort of mechanisms should be implemented to prevent this?

If the thermometer can perform digital signatures and the palmtop can check them, the solution is straightforward: the thermometer's calibration certificate can include the node's public key. Where the thermometer cannot perform public key cryptography, the palmtop will establish a common secret with the thermometer using the techniques of section 3.4 and, having verified its certificate, will be able to accept messages protected by a MAC keyed with the shared secret.

At this point, we depart once more from the conventional wisdom of the computer security community. The obvious objection is that, since neither certificates nor IDs are secret, a false device might be constructed which clones a genuine one; and that the only proper way to use a certificate is to certify a public key whose private key is known only to the device. However, this is tied up closely with the issues of tamper proofness and tamper evidentness. If our devices are not tamper-proof, then the private key can be read out and installed

in a bogus device; but if they meet the much weaker requirement of tamper-evidentness (say with sealed enclosures), a forger will not be able to produce an intact seal on the bogus device. So we will have confidence in a certificate which we receive protected under an ignition key that we shared successfully with a device whose seal was intact. (This is the first example we know of a purely "bearer" certificate: it need not contain a name or even a pseudonym.) We will now discuss this in more detail.

### 4.2 Tamper resistance

The doctor's reliance on the "genuine thermometer" certificate assumed that, after the thermometer was last inspected, calibrated and certified, it stayed that way. This assumption may be questioned in many applications, especially as in Piconet not all nodes are well guarded, highly personal accessories such as Java rings [11] over which the owner is expected to keep close and continuous control. On the contrary, many nodes (such as broadcasting sensors) may be commodities that are scattered around and left to their fate. With such a model an attacker may, and sooner or later will, modify or forge a deployed node, possibly redeploying the corrupted node in an unsuspecting environment.

This can in theory be avoided by making the node tamper-proof, but it is much easier to talk about this property than to implement it in practice [1], especially within the cost and form factor constraints of personal consumer electronics devices. Under the circumstances, it is not clear how much extra assurance is given by furnishing our thermometer with the ability to do public key cryptography; such a device can have its private key read out just as a device with a certificate but without a private/public keypair can be forged.

In such environments it may often be more suitable to use physical tamper-evidence mechanisms (such as seals) rather than electronic mechanisms (such as tamper sensing switches that zeroise memory). In this case, one must still design the device so that non-intrusive attacks (such as those based on protocol failure, power analysis and glitch attacks [2]) are not practical; it is also necessary to take into account the time that might pass before a broken seal is noticed, and the likelihood of successful attacks on the sealing mechanism [9].

It must also be realised that the tampering may not be limited to the on-board code and keys: a very effective attack on the unattended thermometer is to simply replace its analogue sensing element with a bad one. This attack highlights that even enclosing the entire processor, memory and backup battery in a high-grade tamper resistant enclosure, with only a ribbon connector to interface with the outside world, would still leave us vulnerable to direct attacks on its "peripherals". Bringing the sensor itself within the tamper resistant enclosure may make manufacturing too expensive (the computing and communication core willl no longer be a modular building block) and may even interfere with the proper working of the sensor. So the transducer may be an Achilles' heel, and it may not be worth spending large sums on tamper-proofing the core if the sensor cannot economically be protected.

When making decisions about what level of tamper-proofness or tamper-evidentness a system needs, it is as well to bear in mind that corrupt nodes can be used in a number of ways. Attacks might be immediate and direct, or alternatively the attacker might field a number of nodes which would accept software upgrades from him as well as from the authorised source.

### 4.3 Software upload

For nodes to be useful, there has to be a way to upload software into them, if nothing else during manufacture; in many applications we will also want to do this after deployment. So we will want to prevent opponents from exploiting the upload mechanism, whatever it is, to infiltrate malicious code, and we will want to be able to detect whether a given node is running genuine software or not.

Neither of these goals can be met without assuming that at least some core bootstrap portion of the node escapes tampering. The validity of such an assumption will depend on the circumstances; the expected motivation and ability of the attackers, and the effort spent not just on protecting the node with tamper-resistance mechanisms and seals, but in inspection, audit and other system controls.

## 5 Confidentiality

We find that we have little to say about confidentiality other than remarking that it is pointless to attempt to protect the secrecy of a communication without first ensuring that one is talking to the right principal. Authenticity is where the real issues are and, once these are solved, protecting confidentiality is simply a matter of encrypting the session using whatever key material is available.

In the event that covert or jam-resistant communications are required, then the key material can be used to initialise spread-spectrum or frequency-hopping communication. Note that, in the absence of shared key material and an accurate time source, such techniques are problematic during the important initial resource discovery phase in which devices try to determine which other nodes are nearby.

## 6 Conclusions

We examined the main security issues that arise in an ad-hoc wireless network of mobile devices. The design space of this environment is constrained by tight bounds on power budget and CPU cycles, and by the intermittent nature of communication. This combination makes much of the conventional wisdom about authentication, naming and service denial irrelevant; even tamper resistance is not completely straightforward.

There are interesting new attacks, such as the sleep deprivation torture, and limitations on the acceptable primitives for cryptographic protocols. However,

there are also new opportunities opened up by the model of secure transient association, which we believe may become increasingly important in real networking applications.

The contribution of this paper was to spell out the new problems and opportunities, and to offer a new way of thinking about the solution space—the resurrecting duckling security policy model.

## 7 Acknowledgements

We thank Alan Jones for suggesting the wireless thermometer, a prototype of which had just been built in the context of Piconet, as a minimal but still meaningful practical example.

## References

1. Ross Anderson and Markus Kuhn. Tamper resistance—a cautionary note. In *Proc. 2nd USENIX Workshop on Electronic Commerce*, 1996.
2. Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In Mark Lomas et al., editor, *Security Protocols, 5th International Workshop Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 1997.
3. Infrared Data Association. http://www.irda.org/.
4. Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications*, 4(5):8–15, October 1997.
5. Kenneth J. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, MITRE Corporation, April 1975.
6. HomeRF Working Group. http://www.homerf.org/.
7. Jaap Haartsen, Mahmoud Naghshineh, Jon Inouye, Olaf J. Joeressen, and Warren Allen. Bluetooth: Visions, goals, and architecture. *ACM Mobile Computing and Communications Review*, 2(4):38–45, October 1998.
8. IEEE. IEEE standard for a high performance serial bus. IEEE Standard 1394, 1995.
9. Roger G. Johnston and Anthony R.E. Garcia. Vulnerability assessment of security seals. *Journal of Security Administration*, 20(1):15–27, June 1997.
10. Konrad Lorenz. *Er redete mit dem Vieh, den Vögeln und den Fischen* (King Solomon's ring). Borotha-Schoeler, Wien, 1949.
11. Sun Microsystems. http://java.sun.com/features/1998/03/rings.html.
12. Kevin J. Negus, John Waters, Jean Tourrilhes, Chris Romans, Jim Lansford, and Stephen Hui. HomeRF and SWAP: Wireless networking for the connected home. *ACM Mobile Computing and Communications Review*, 2(4):28–37, October 1998.
13. Bluetooth SIG. http://www.bluetooth.com/.