Imagine the future: hundreds of embedded computers per person, all cooperating via ad hoc wireless networks. What will the security implications be?

The Resurrecting Duckling:
Security Issues for Ubiquitous Computing

Frank Stajano and Ross Anderson, University of Cambridge

A common view of the Internet divides its history into three waves: originally, mainframes and terminals; until yesterday, PCs, browsers, and a GUI; starting tomorrow, wirelessly networked processors embedded in everyday objects. By 2003, there could be more mobile phones connected to the Internet than computers. Within a few years, we will see many of the world's fridges, heart monitors, bus ticket dispensers, burglar alarms, and electricity meters sending messages to each other. Networked processors will be extremely cheap commodities embedded in everything from furniture to clothes. On the nanotechnology front, swarms of microscopic robots will cooperate in decentralized federations of autonomous agents that will give the terms "distributed system" and "peer-to-peer" entirely new meanings.

The ubiquitous computing vision—of spontaneous interaction between the digital devices that surround and serve us—could bring a great deal of convenience but also a great deal of risk. If it takes off as anticipated, ubiquitous computing will have an impact on society similar to that of the Web. So its vulnerabilities will have major repercussions, and it is prudent for scientists and engineers to study the protection issues before a critical mass of applications gets built and deployed.[1]

The traditional taxonomy of security threats identifies three main classes, depending on whether the system property being threatened is confidentiality, integrity or availability. *Confidentiality* is violated when unauthorized principals learn protected information, such as your medical records. *Integrity* is violated when unauthorized principals modify information, as when someone changes the amount or the beneficiary on a check. *Availability* is violated when the system is prevented from performing its intended function, as when someone brings down the Web site of an online store.

These protection properties all rely on a distinction between authorized and unauthorized principals. Discriminating between the two usually involves a three-step process: identification (the user says who she is), authentication (the system verifies the validity of this claim), and authorization (she is granted specific access rights). A failure of authentication can easily lead to violations of confidentiality, integrity, and availability. For example, protecting your secrets with encryption does little good if the true identity of your recipient is not what you anticipated. So it is natural, given the task of protecting a new computing environment, to look at authentication first.

Authentication

Peer-to-peer and ubiquitous computing systems involve many principals, but their network connectivity is intermittent and not guaranteed. Traditional approaches to authentication, from Kerberos to public-key certificates, are therefore unworkable, because they rely on online connectivity to an authentication or revocation server. We need new solutions.

Secure transient association

The main application of authentication to intermittently connected networks is itself new. We call it *secure transient association*, and we have identified many instances of this paradigm of interaction in applications as diverse as mobile computing, consumer electronics, car security systems, medical equipment, weapons systems, vehicle tachographs, and automatic teller machines.

To visualize secure transient association, imagine the following scenario: In the ubiquitous computing world, you no longer want to litter your coffee table with an array of remote controls for your TV, stereo, DVD, VCR, curtains, central heating, and air conditioning. Instead, you want all of these systems to obey a universal remote control, which for the sake of argument will be some kind of PDA. Because you no longer buy the remote control with the appliance, you need to be able to establish an *association* between the two after purchasing the appliance. Because you don't want your neighbor to be able to activate your appliances (whether by accident or malice), you want this association to be *secure*. And, because you want to be able to resell your old stereo while keeping your PDA, and you want to be able to replace a broken PDA without losing control of all your appliances, you also want this association to be *transient*, or revocable.

We have been working on this issue since 1998.[2] Our solu-

tion is a security policy model describing the properties that a system should possess to implement a satisfactory secure transient association: we call it the *Resurrecting Duckling* policy. The name was inspired by the work of Konrad Lorenz, the Nobel-winning investigator of animal behavior, who described how a goose hatchling assumes that the first moving object it sees must be its mother.

The Resurrecting Duckling security policy model

Our idea is to have a slave device (such as your DVD player) imprint itself to a master (such as your PDA) through the transfer of an imprinting key, or "soul." Once the slave device, the "duckling," is imprinted, it remains faithful to the master, its "mother duck," for as long as that soul persists. When the duckling dies, the soul dissolves, and the duckling's body is ready for imprinting to a new and possibly different mother duck. In other words, a duckling that dies may be resurrected later with a different soul. (Adherents of many religions believe in metempsychosis—that, on death, the body dissolves and the soul migrates into a new body. Our proposal might be described as "reverse metempsychosis:" the soul dissolves, and the body is inhabited by a new soul.)

More formally, four principles define the Resurrecting Duckling security policy model: *two states*, *imprinting*, *death*, and *assassination*. (For definitions of these principles, see the sidebar.)

Interestingly, several real-world security artifacts, such as a laptop with a password-protected BIOS, already behave almost like ducklings; but they do not comply with the assassination principle, and therefore are too easy to subvert. Sometimes this is a deliberate tradeoff in favor of availability—a means to let the legitimate owner regain control even if she can no longer prove that she is the mother duck—but more often it is simply a matter of cost and convenience.[3]

We can extend our policy model to encompass a great variety of relationships between devices.[4] The mother-to-duckling relationship is a master-slave one, but sometimes we might wish the duckling to interact with peers, or to have other masters besides its mother. When we want our camera to ask our cellphone to use its microphone and A/D conversion facility to annotate photographs vocally, it makes little sense for the camera to become the cellphone's mother duck; still, we want one device to be able to give instructions to the other.

The way we address this problem is to introduce a "device policy," in the style of Policymaker,[5] as a level of indirection. The mother duck is the entity that can edit the duckling's device policy, while the device policy itself says which credentials a principal must present for the duckling to perform a given action for it. This means that we now have two levels of being master: the long-term master is the mother duck, who can delegate a subset of her powers to another principal or even to a group of principals along the peer-to-peer model.

It is prudent to set some limits, however. If the mother duck allows anyone else to order the duckling to commit suicide, then this delegate might commandeer the duckling without the original mother duck being able to stop him. Allowing your local flower-arranging society to meet in your home every Saturday is one thing; but if you gave every one of its members a copy of your house key, you might come home to find that one of them had changed the cylinder in your absence and locked you out. So the ultimate control over most devices will probably be closely held, as a precaution against service-denial attacks by malicious or subverted principals.

The Four Principles of the Resurrecting Duckling

The Resurrecting Duckling security policy model describes a way of establishing a secure transient association between two devices—a master and a slave. Four principles define the Resurrecting Duckling:

- *Two states*. The duckling can be in one of two states: imprintable or imprinted. In the imprintable state, anyone can take it over. In the imprinted state, it obeys only its mother duck.
- *Imprinting*. The transition from imprintable to imprinted happens when the mother duck sends an imprinting key to the duckling. This must be done using a channel whose confidentiality and integrity are adequately protected.
- *Death*. The transition back from imprinted to imprintable is known as death, and can only be initiated by an order from the mother duck.
- *Assassination*. The duckling must be constructed in such a way that it will be uneconomical for an attacker to assassinate it—that is, to cause the duckling's death artificially in circumstances other than the one prescribed by the death principle.

Confidentiality

When people think about security issues for ubiquitous computing, they first think of eavesdropping as a consequence of wireless networking. But this concern is vastly exaggerated: once we have solved the hard problem of authenticating the principals and sharing key material, we have mature and robust symmetric ciphers for protecting a communications channel's confidentiality. The actual problems are elsewhere.

Bits per second or bits per joule?

The size and shape of the typical ubiquitous computing device impose new constraints. Untethered devices are battery powered, so they can't use the fastest and most powerful processors available lest they require extremely frequent recharges (as laptop users know all too well). Many ubiquitous computing devices therefore have "peanut" processors that are too slow for computationally intensive tasks such as public-key cryptography.

This is well known, and one traditional way of dealing with peanut processors is to do most of the work as background tasks or as precomputations. But the batteries of miniature portable devices hold only a small, finite amount of energy; this places a bound on the total amount of computation the devices can perform, rather than on the rate at which they can perform it. This problem is new and more interesting: to evaluate a cipher (or any other algorithm) on a peanut device (as opposed to a peanut processor), the most relevant performance figure is no longer bits per second, but bits per joule. (Power constraints could favor the introduction of asynchronous

processors, which run without a clock and halt when no computation is being performed.)

Biometrics, coercion, traffic analysis, and more

While it is straightforward to protect the confidentiality of wireless traffic, it is much harder to protect the confidentiality of the information held in the devices themselves. At present, few people worry about this. Most PDAs, for example, which are relatively likely to be lost or stolen, are not even password protected. Even those that are password protected use unencrypted storage, and are therefore at the mercy of the moderately resourceful attacker.[6] This is hardly surprising, as the value of the information held in the typical PDA is small, both for the owner and for the potential thief.

In the future, however, the ubiquity of computing devices will multiply the opportunities for storage of information about our activities. Our digital butlers will have as their explicit mission the job of discovering and remembering as much as possible about our habits and idiosyncrasies. Finally, it would not be the first time that a technology infrastructure introduced for one purpose was misused for another to the detriment of personal privacy—think of credit cards being used to gather purchasing patterns.

The ubiquity of processing and communicating power will bring convenience, but also a great deal of risk.

It is thus important to protect the confidentiality of the data held in at least the mother-duck devices. There are several components to this problem. The first is in finding methods that let users authenticate themselves to the device, whether by a password or a biometric (such as a manuscript signature done with the pen on the PDA)—and this is harder than it looks.[7] The second is protecting within the device any long-term keys used to encrypt private data, such as the user's profile and the imprinting keys of controlled devices. The third is security renewability—the problem of recovering from a PDA theft that occurs while the device is "live" or when the thief has observed or can guess the owner's password. (The Duckling model could be part of the solution here.) The fourth is the issue of resistance to coercion.

Finally, we must consider metadata protection. Anonymity, traceability, and traffic analysis are aspects of confidentiality that have so far been underestimated, but they will take on much greater importance in the ubiquitous computing context. Encryption makes it easy to protect the *what* of a conversation; but the *when*, the *from*, and the *to*—not to mention the very fact that a conversation is taking place—remain observable. Defending against traffic analysis is a difficult problem, and an active research area. From the user's point of view, a conscious effort to support location privacy and to make a user's transactions difficult to link to each other is necessary at the design stage; otherwise the ubiquitous computing infrastructure will become a tool for ubiquitous surveillance.

Integrity

The basic integrity problem is to ensure that messages from one principal to another are not corrupted by a malicious third principal. This is similar to confidentiality in that, once we know how to do authentication and key distribution, the problem is trivial to fix using well understood cryptographic mechanisms, such as message authentication codes. Authenticating broadcast data is somewhat trickier if we wish to avoid the power cost of computing a series of digital signatures, but researchers have devised several chaining protocols to tackle this problem.[8] The most serious integrity problem for ubiquitous computing, therefore, is once again not with the messages in transit but with the device itself.

Tamper resistance and tamper evidence

How can I establish whether a device I am using for communication has been subtly modified or even replaced with a fake? It is easy to recognize this as an authentication problem, and there is a close relationship between authentication and integrity. The Duckling solution may come in handy once again. There is, however, another aspect to the solution—physical tamper protection.

The usual assumption underlying authentication is that the network is insecure and under the control of the attacker, but that the principals involved are capable of keeping their secrets. Ubiquitous computing takes this assumption and, as it does with so many others, turns it on its head: network attacks will often be easy to deal with, but attackers are likely to subvert many of the principals.

Providing high-grade tamper resistance, which makes it impossible for an attacker to access or modify the secrets held inside a device, is expensive and difficult.[9] It is often better to rely instead on *tamper evidence*, which ensures that tampering attacks leave a visible trace. The main objection to this strategy is that it breaks open the loop of machine-based verification. A physical seal's integrity cannot be verified as part of the authentication protocol; instead, it requires human inspection.

Some might see this as a security hole, but it could actually be an advantage. It means that the responsibility for protection rests with the person relying on that protection, rather than with some third party who might have different motives. In addition, managing the protection is also a matter of common sense. Two very common causes of security failure are that the principal responsible for the security is not the principal relying on it,[7] and that technical mechanisms such as public-key certification are too hard for normal mortals to understand and manage.[10] It is somewhat hubristic for engineers to assume that they must solve all problems using mechanisms within their realm of professional expertise, when other, simpler, mechanisms might be more robust.

Availability

The classical attack on a wireless system's availability is to jam the communication channel. This problem has been exten-

sively studied for its military implications,[7] and we can recycle much of the know-how developed in that field for civilian use.

Ubiquitous systems that depend on short-range RF communication will of course fail completely in the presence of jamming, but the methods for dealing with it lie outside system design: once the jammer moves out of range (or once the police take him away), the network can resume normal activity. The more interesting and novel denial-of-service attack emerges from the relationship between security and power conservation that we mentioned earlier. If a device has limited battery energy and tries to sleep as often as possible to conserve it, keeping it awake until this energy runs out can be an effective and selective attack. Once the battery is flat, the attacker can walk away, leaving the victim disabled. We call this cruel treatment *sleep deprivation torture*.

You might think that authentication could prevent such attacks, but this is not always the case. Authentication lets you distinguish friends from unknowns; but in some applications you cannot refuse to serve unknowns—for example, if you are a Web server. The dilemma for the server is whether to answer queries from unknowns: they might be staging a denial-of-service attack, but they might be genuinely interested in the answer. Identifying repeat offenders is futile, both because source information can easily be faked, and because a villain might subvert multiple "innocent" principals into cooperating in the attack—the so-called DDOS (distributed denial-of-service) attack.

When the server has several functions of different importance, we can prioritize them and use a resource allocation strategy to hard-limit the amount of resources that the less-important uses can consume. This guarantees a certain level of service to the more important uses. Of course, this still fails to protect against some types of attacks, such as those from authorized insiders.

One approach to this problem is what we call *plutocratic access control*: you receive service if you've got the money to pay. By charging for access, the server limits the extent to which clients can indiscriminately ask for resources. In fact, if the charge is such that the server makes a profit in serving a user, the denial-of-service problem may no longer be a concern—exhaustion of the available capacity simply means that the server has made as much money as it possibly could!

If charging actual money is not practical, the server can still use the same limiting strategy by forcing users to undergo some expensive sacrificial ritual in exchange for service. Several writers have suggested that servers make clients solve cryptographic puzzles or answer a question that would be easy for a human but hard for a machine. The latter might be more suited to peer-to-peer applications, while the former might be better in ubiquitous computing environments.

Ubiquitous computing is widely believed to be the Internet's next evolutionary stage, and it is already under way. But having hundreds or thousands of computers per human being, instead of just a few, will change the game in a fundamental way.

According to the ubicomp vision, computers will evolve from versatile, general-purpose, but complicated and unreliable machines to dedicated, specialized, inflexible, but simple and reliable information appliances. Researchers are still divided about whether this will bring great usability benefits[11] or simply frustrate more people at a higher level.[12] Whichever of these visions proves the more prophetic, the ubiquity of processing and communicating power will bring a great deal of risk. There will be more ways, and more complex ways, in which things can go wrong, and many of these will be exploited by malicious people to gain some advantage. It is important to study the risks now, before we deploy an infrastructure that might otherwise be insecure, unreliable, and intrusive.

The principal security issues for ubiquitous computing differ in a number of interesting ways from the protection issues in conventional distributed systems, but they are often similar to the issues in peer-to-peer communications. Authentication of anonymous principals is important; attacks on nodes are more probable than attacks on communications; and service-denial attacks are one of the principal problems we have to manage. To tackle the new problem of secure transient association, we've offered an original solution, the Resurrecting Duckling policy model.

We hope that this work, by promoting awareness of the security issues that we face, will contribute to the deployment of a ubiquitous computing infrastructure designed to minimize the corresponding economic and social risks.

References

1. F. Stajano, *Security for Ubiquitous Computing,* John Wiley & Sons, Chichester, UK, 2002.
2. F. Stajano and R.J. Anderson, "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks," *Proc. Seventh Security Protocols Workshop, Lecture Notes in Computer Science 1796,* Springer-Verlag, Berlin, 2000, pp. 172–182.
3. R.J. Anderson and M.G. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Proc. Fifth Security Protocols Workshop, Lecture Notes in Computer Science 1361,* Springer-Verlag, Berlin, 1998, pp. 125–136.
4. F. Stajano, "The Resurrecting Duckling—What Next?" *Proc. Eighth Security Protocols Workshop, Lecture Notes in Computer Science 2133,* Springer-Verlag, Berlin, 2001, pp. 204–214.
5. M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *Proc. 17th IEEE Symp. Security and Privacy,* IEEE CS Press, Los Alamitos, Calif., 1996, pp. 164–173.
6. A.J. Sammes and B. Jenkinson, *Forensic Computing—A Practitioner's Guide,* Springer-Verlag, London, 2000.
7. R.J. Anderson, *Security Engineering—A Guide to Building Dependable Distributed Systems,* John Wiley & Sons, New York, 2001.
8. A. Perrig et al., "Efficient and Secure Source Authentication for Multicast," Proc. Network and Distributed System Security Symp., Internet Society, Reston, Va., 2001; www.isoc.org/isoc/conferences/ndss/01/2001/INDEX.HTM.
9. R.J. Anderson and M.G. Kuhn, "Tamper Resistance—A Cautionary Note," *Proc. Second Usenix Workshop on Electronic Commerce,* Usenix Assn., Berkeley, Calif., 1996, pp. 1–11.

10. D. Davis, "Compliance Defects in Public-Key Cryptography," *Proc. Sixth Usenix Security Symp.*, Usenix Assn., Berkeley, Calif., 1996, pp. 171–178.

11. D.A. Norman, *The Invisible Computer*, MIT Press, Cambridge, Mass., 1998.

12. A. Odlyzko, "The Visible Problems of the Invisible Computer: A Skeptical Look at Information Appliances," *First Monday*, vol. 4, no. 9, Sept. 1999; www.firstmonday.dk/issues/issue4_9/odlyzko/index.html.

Frank Stajano is a faculty member at the Laboratory for Communications Engineering of the University of Cambridge, where he holds the ARM Lectureship in Ubiquitous Computing. His book, *Security for Ubiquitous Computing* (John Wiley & Sons, Chichester), develops in detail the topics touched upon in this article. Contact him at http://www-lce.eng.cam.ac.uk/~fms27/.

Ross Anderson leads the security group at the Computer Laboratory of the University of Cambridge, where he is Reader in Security Engineering. He is the author of *Security Engineering—A Guide to Building Dependable Distributed Systems* (John Wiley & Sons, Chichester). Contact him at www.cl.cam.ac.uk/~rja14/.

News Briefs

IT Spending Outlook: Security Still a Growth Market

Many companies have scaled back new technology purchases, but business is booming for companies that provide security software and managed security services. Despite tight IT spending budgets, Gartner Dataquest, a market analysis firm, predicts the worldwide security software market will reach $4.3 billion in 2002, an 18 percent increase over $3.6 billion in 2001.

In 2001, the events of 11 Sept., several well publicized hacks, virus outbreaks, and distributed denial-of-service attacks all heightened public awareness of the growing need for information security, ultimately boosting the market. "Enterprises are looking particularly at defensive security technologies such as antivirus software, intrusion detection systems, and firewalls," said Dataquest analyst Colleen Graham.

Meanwhile, the US market for managed security services, which amounted to about $720 million in 2000, will grow to $2.2 billion by 2005, according to IDC, a market analysis firm. Firms providing such services include network or systems integrators, service providers or xSPs, technology owners, and pure-play security service firms.

"The managed security services market is being driven primarily by resource constraints to capital and security expertise, as well as the growing complexity of networks and rogue access points," said IDC analyst Allan Carey. He added that demand for managed security service providers will likely be strongest among small and midsized businesses, which need strong security but have limited information security skills and resources on hand.

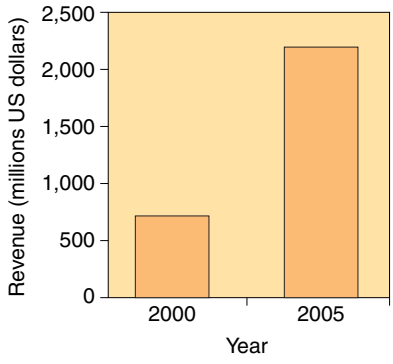

Figure 1. Managed security service revenues are expected to grow nearly 25 percent annually through 2005. (Source: IDC)