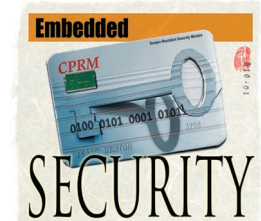


Making Home Automation Communications Secure



A network security system can leverage limited resources to provide confidentiality, authentication, authorization, and integrity for remote monitoring and control of home automation devices.

Peter Bergstrom
Kevin Driscoll
John Kimball
 Honeywell
 Laboratories

Long the futuristic domain of hobbyists, home automation is now moving from the bleeding edge to the mainstream. Utility service providers offer specialized and valuable applications such as energy load management to retain customers in the face of competition. More new homes are wired for intelligent control, with security, comfort, and convenience systems becoming network-aware. Homeowners can orchestrate and monitor appliances from multiple locations within the house or even remotely via telephone and the Internet, and delegate limited controls to utility service providers.

New communications technology offers numerous benefits, but it also opens home automation to many security threats. To protect against these threats within the limited resources of a typical home automation system, we have developed a family of products based on Honeywell's Global Home Server, a remote Web site that provides secure Internet access and other services to client installations.

APPLICATION DOMAIN

Home automation involves three significant technological developments.

- *Focused subsystems.* These features use local information to automate desired performance, such as programmable thermostats that change home temperature based on a time schedule. The average homeowner has limited ability to integrate different home functions, but hobbyists have often gone to great lengths to do so.
- *Integrated whole-home behavior.* Safety, comfort, health, information, and entertainment can be

integrated into one system that, for example, can change comfort settings based on variable home occupant activities rather than a fixed time schedule. These site-specific systems are mostly custom-designed at significant cost, and updating them as the homeowner's needs evolve generally takes a fair amount of effort, often involving a trip by a service technician.

- *Distributed home automation.* Enabled by widespread adoption of the Internet, a new class of applications is under development that "run" substantially outside the home, thereby eliminating the need for a costly, dedicated PC. Using controllable devices and network bridges within the home, upgrading these applications is easy and they can be tailored for specific individuals and markets, such as different regions of the world.

Rather than evolving sequentially, all these systems exist in various markets, complicating the application domain.

Until recently, most home automation products functioned within focused subsystems or integrated whole-home systems, with cost usually the key discriminating factor. Distributed automation combined with improvements in controllable products—especially those incorporating the Internet's communications potential—now make the long-anticipated smart home affordable.

Expanded communication, however, increases the risk of security breaches. In developing a family of Web-enabled home automation products for residential markets worldwide, we have focused on safe and secure operation. Understanding the severely limiting

communication and computation resource constraints for successful products led us to a comprehensive communications security approach scalable from very small up to much more powerful devices.

GLOBAL HOME SERVER

The Global Home Server is the Internet-based communications connection that lets common Internet applications, such as Web browsers, monitor and control homes with a home automation system (HAS). The targeted capacity for each GHS network is support for up to 100,000 homes by a small cluster of modest server machines, and support for larger communities with additional configuration and network management. We are implementing multiple GHS networks, some managed by Honeywell and others by various service providers. A utility service provider operating a GHS can include services that customers access online. The service provider retains complete control of implementing and deploying these services, particularly the choice of Web servers and security measures.

As Figure 1 shows, the GHS design permits a wide variety of in-home devices to use their own preferred communications protocols and project the capabilities in standard ways for GHS application developers. Protocol handlers provide a clean way to integrate several existing and forthcoming product families. A guiding principle is to encourage application-programming-interface-level uniformity while respecting these markets' tremendous diversity—no single company can provide all of the products that consumers will demand, but it is reasonable for a homeowner to expect control to be integrated.

The GHS application programmer interface is based upon Universal Plug and Play (<http://www.upnp.org>), which combines the hypertext transfer protocol, Extensible Markup Language, and simple object access protocol (SOAP) to facilitate discovery and interoperation of diverse devices and services within a home. We chose UPnP protocols for their platform independence, flexibility, and rich application modeling capability. This modeling capability will play an important role as service providers develop more complex applications. We also have prototyped complete UPnP HAS products—including an application, a Web server, and a full TCP/IP stack in little more than 100 Kbytes of code on an 8-bit processor—demonstrating that UPnP can span a wide range of platforms.

Common Object Request Broker Architecture, the Distributed Component Object Model, Java's remote method invocation, Jini (Sun Microsystems' device-to-device interoperability protocol), and the Open Services Gateway Initiative (OSGi) also support rich execution models, but their platform specificity and cost complexity pose problems. Like SOAP, Corba, DCOM, and RMI are candidates for shaping the

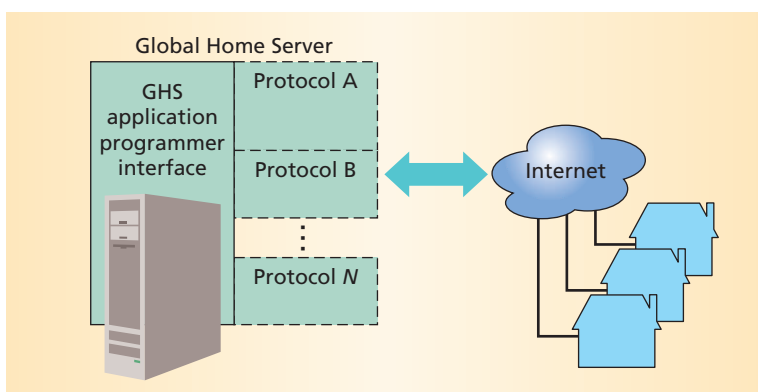


Figure 1. The Global Home Server lets a wide variety of Web-enabled home automation applications use their own protocols.

server-side API. However, UPnP provides a basic yet flexible structure for organizing devices and describing capabilities.

Platform independence is another UPnP key strength. Choice of server back-end platform is no issue, but the customer's application environment often ranges from Java to C++ to server-side scripted Web pages. Because SOAP uses UPnP in an expressly platform-neutral way, a very efficient implementation is possible.

We considered Jini and OSGi for embedded products, but even the most stripped-down Java virtual machine requires 100 Kbytes or more of code space without considering the applications. In addition, while the Jini specification was apparently written to enable more efficient non-Java implementations, a JVM-based implementation is more practical. As a result, we selected one UPnP-based interface as a standard for a wide range of devices and applications as possible.

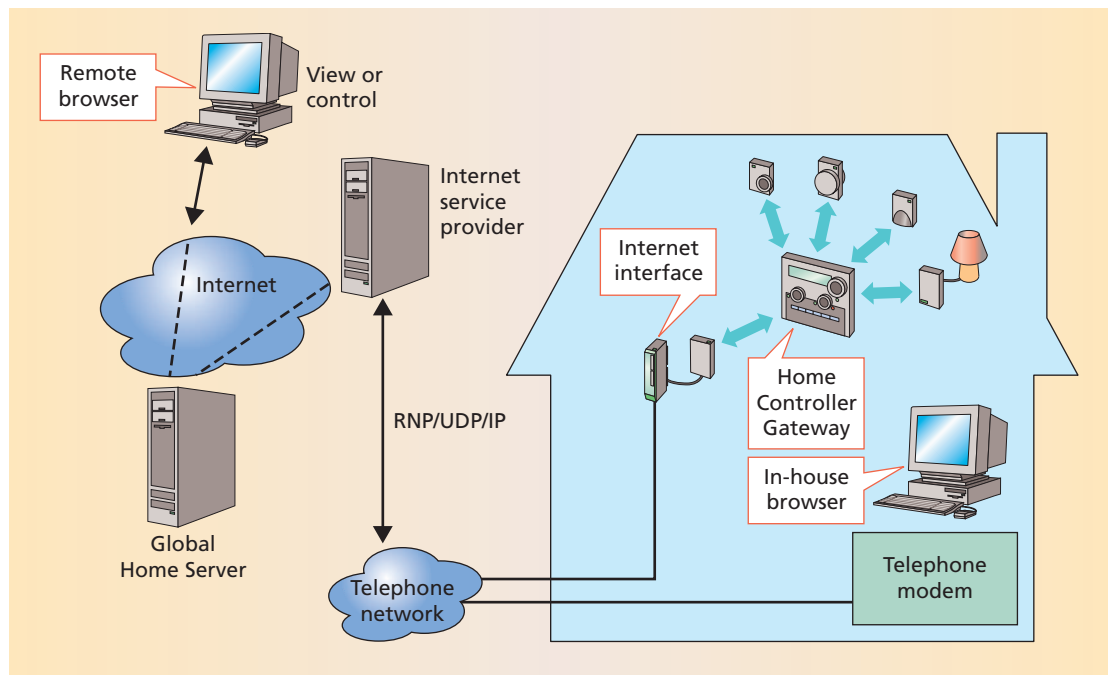
HOME PRODUCTS AND CONNECTIVITY

Physical devices such as communicating thermostats, sensors, security systems, light switches, and cameras form the basis for residential control. Many of these products have logic capability built in. They communicate via a variety of media and protocols. For example, the residential network protocol (RNP), a general-purpose application layer protocol designed for resource-limited devices, is used over wired and wireless networks to express home-control-related semantics. Web-enabled control of these devices is supervisory rather than via continuous real-time intervention.

In the North American and Asia-Pacific markets, the Honeywell Home Controller Gateway (HCG) provides

- heating, ventilation, and air-conditioning (HVAC) control;
- physical intrusion and fire security detection; and
- lighting and appliance control.

Figure 2. Remote access to a home automation system via GHS. Homeowners can monitor and control a variety of devices through a remote Web browser. Devices within the home interact over a home network.



The HCG is a broadband-enabled device that retains a continuous connection with the GHS to provide a rich user-interface experience. In Europe, the Honeywell Hometronic system provides HVAC control, lighting, shutter and appliance controls, and light, water, wind speed, and other sensors. The Hometronic Internet Module (HIM) uses a dial-up telephone connection via an Internet service provider to gain access to the GHS. Because online GHS connections are established upon user demand and designed to minimize bandwidth, the HIM can enter markets with limited or no always-on broadband availability.

As Figure 2 shows, a Web-enabled application lets homeowners use the Internet to monitor and control home devices from remote locations. After secure login, the GHS system presents the homeowner with a capable interface tailored to the service provider, which may be an energy utility company or residential community management firm. The GHS system establishes a communications session with the appropriate home system and permits the person to view and adjust the home controls. The sensitivity of this information—for example, knowledge that the home is presently in “away” mode—makes communications privacy critical.

SECURITY CONCERNS

An increasing number of embedded devices have network connections. Inside the home, networks transmit status and control information among sensors, appliances, and the HAS. Linking the home network with Internet-based Web servers can make home automation much more useful. In addition to remotely accessing, monitoring, and controlling the HAS, homeowners can benefit from new services that alert the HAS to weather, energy rate, or other changes. These services can also act on information the HAS provides, alerting users that, for example, the base-

ment is flooding or an appliance needs maintenance.

However, as embedded devices become accessible via the Internet or telephone, they incur significant security risks. Many systems advertise the capability for using cell phones or paging systems for remote control of a HAS, ignoring the security issues and providing little or no protection. Most US cell phones lack encryption, while those that do have it use weak encryption algorithms. Worse, the ease with which PINs and passwords can be compromised makes these measures virtually useless.

Thus, implementing security with the limited resources available to an embedded device presents a serious challenge that any system must address directly. Applying postdeployment security patches to embedded devices without an existing secure infrastructure offers little protection given the obstacles to correctly distributing these updates to a widely scattered user base.

Targets

A malicious hacker, disgruntled insider, industrial spy, or cyberterrorist could attack the company providing the system, a particular house, or a large number of random houses in many ways. If their attack on a distributed home control system is successful, hackers could eavesdrop on home sensors or manipulate home actuators—with predictably dire consequences.

Confidentiality. An attacker could use a home automation system’s audio and video sensors to intercept information about one or many houses. Services providing information about the home status—for example, lights on/off, temperature, home/away mode—enable indirect surveillance of home activity. An enterprising burglar could use a script to collect a list of unoccupied houses with disabled alarms. Lack of security in a home automation system also makes it possible to remotely stalk an individual.

Inadequate remote-access security exposes home automation systems to dire consequences.

Authenticity. An attacker who has learned how to masquerade as a legitimate HAS operator can obtain security-critical home-state and sensor readings or issue commands that range from annoying to dangerous. In the simplest form of masquerading, an attacker records the control communications until observing the desired action—for example, the burglar alarm is off—and later replays the recording into the communication channel. In more sophisticated man-in-the-middle attacks, an attacker pretends to be either a legitimate operator or the HAS at the other end.

Authorization. An attacker could use false or pirated authorizations to perform prohibited actions. For example, a homeowner may want to delegate the authority to execute functions such as allowing a utility to read the meter for billing or controlling loads for off-peak programs or emergency load shedding. However, the homeowner may not want the utility to monitor or control the burglar alarm. Several sets of authorizations can exist—for each service provider, for household adults versus children, for visitors, and so on. Although authentication often controls authorization, the two are not synonymous.

Integrity. An attacker can undermine the system's integrity by modifying data or commands. Tampering with messages in transit or replaying, reordering, deleting, or delaying messages can cause the system to issue commands not intended by a legitimate HAS user. If the data in transit is a software download, tampering can have wide-ranging effects that do not manifest immediately. An attacker could, for example, write a “poltergeist virus” that turns home automation against the homeowner by manipulating remotely controllable appliances and systems such as HVAC.

Consequences

Inadequate remote-access security exposes home automation systems to dire consequences. In the case of home healthcare devices, a compromised system can cause a life-threatening situation. Simply turning off all the electricity at the wrong moment can cause a fatality.

The proliferation of insecure remote access to home automation can even threaten a nation's electrical infrastructure. An attack can set up numerous dupe proxies, each connected to a group of homes, causing the homes to simultaneously turn their electrical loads off and on. Orchestrated correctly, the wide load swings can direct current surges that cause specific circuit breakers to trip, creating a domino effect that disables an entire regional or national electrical grid.

A LAYERED DEFENSE

System administrators face the challenge of defending against these threats while providing access to mul-

iple services for different user classes, all within an embedded device's limitations. Fortunately, networking allows layered architectures that do not rely solely on the embedded device. Potentially untrustworthy principals never connect directly to the HAS. Instead, the GHS, which has more computational resources and is more accessible for monitoring and maintenance, serves as the point of contact for homeowners, contractors, and other parties connecting via the Internet. The HAS extends and effectively hides behind the GHS, which performs all user authentication and authorization functions.

For layering to work, the HAS only communicates with the GHS via a secure, two-way Internet link. Protecting message traffic to and from the HAS against third-party eavesdropping and manipulation requires satisfying five goals:

- *Confidentiality.* Third parties cannot read messages in transit.
- *Authentication.* Upon receipt of a message purporting to be from the embedded device or its central server, the system must be able to identify the sender.
- *Authorization.* Actions can occur only with permission.
- *Integrity.* Upon receipt of a message, the system must be able to determine whether someone tampered with that message in transit.
- *Key management.* Establishing and updating the shared secrets needed to implement these capabilities must be reliable and secure.

Security layer

To meet these goals, communication between the GHS and each HAS includes a security layer that

- prevents snooping of confidential homeowner data;
- authenticates and protects the integrity of messages critical to home safety and security by preventing impersonation, message tampering, or message replay, reorder, deletion, or delay from affecting home safety and security;
- prevents snooping of security data such as secret keys that could be used to compromise the in-home device or central server;
- authenticates and protects the integrity of messages that modify security data or software;
- protects device availability and reliability by not accepting commands from outsiders; and
- prevents leapfrogging by preventing attacks against the central server—and, transitively, other homes—launched from a subverted in-home device.

Cryptography Law

Complex and continually in flux, cryptography law (see <http://cwis.kub.nl/~frw/people/koops/lawsurvey.htm> for a survey) varies widely from country to country. A firm in one country creating cryptographic software often needs an export license or license exception to send the software to a company located in another country that is building the hardware and integrating the software. The manufacturer must also comply with any import laws. All the countries involved in the product's distribution likewise face import and export regulations, and the company installing the product must comply with its nation's import and domestic use laws as well. In addition, some countries demand access to all encrypted communications within their borders.

Designed by a joint US-Dutch team, Honeywell's Hometronic Internet Module illustrates the gauntlet that any high-volume embedded communication security product faces in today's global economy. Because of the HIM's 224-bit key size, US team members had to get a license exception for unlimited-key-size algorithms from the US Department of Commerce's Bureau of Export Administration (<http://www.bxa.doc.gov/Encryption/guidance.htm>) just to communicate with its Dutch teammate.

The product is also subject to a dizzying array of usage restrictions, import and export laws, one-time reviews, and other obstacles. Truly international in nature, the HIM underwent beta-testing in the US and Europe, with a Global Home Server controlling HIMs in multiple countries, and the system will soon go into service in Asia, Africa, North America, and Europe all manufacture HIM system components.

Insider attacks also pose a threat to the system. Manufacturing, distribution, installation, and operation of products using encrypted software can involve numerous companies in different countries. Distrust among the parties can make it difficult to ensure the dependability of all employees involved in low-cost, high-volume products. As the "Cryptography Law" sidebar indicates, for these and other reasons manufacturers and distributors face complex and ever-changing legal hurdles.

The problem in manufacturing and distribution is not the subversion of one product instance—determining where that product is installed is infeasible—but the systematic subversion of many, if not all, product instances. Targeting specific sites can occur during installation and operation, but this is of less concern because potential perpetrators and victims have a closer relationship. To eliminate threats from the manufacturing and distribution chain, the installation process can insert into a product additional keying data known only to the installer and the GHS.

The system is interactive: A user requesting an action must receive an immediate response that indicates the action's completion. To meet this requirement, the application constructs the request in the user's browser, encrypts it, and sends it to the GHS. The GHS in turn decrypts it, determines how to route it to the correct home controller, re-encrypts it, and sends it to the home controller. The home controller decrypts the request, performs the action, constructs a response, encrypts the response, and sends it to the GHS. Next, the GHS decrypts the response, determines how to route it to the correct user, re-encrypts

it, and sends it back to the user. Finally, the user's browser decrypts and displays the response.

Given the time that the steps in this sequence consume and that each request requires four encryptions and four decryptions, cryptography must add only an imperceptible delay. Because residential network protocol (RNP) message payloads average only 8 bytes, even a small per-message overhead for security can have a significant bandwidth impact.

Reliable transport layer

Guarding against random failures in a global network, not against intentional security attacks, is the first stage of defense for an embedded Internet device.¹ The security layer therefore relies on an underlying reliable transport layer (RTL) to provide ordered and error-free communications data delivery.

The transmission-control protocol provides the normal solution for reliable transport on the Internet. For embedded devices, however, TCP is often too bulky and overly general, providing services the device will never use. As Figure 3 shows, our application's in-home device uses a user datagram/Internet/point-to-point protocol stack. Atop the UDP, a stripped-down reliable transport protocol ensures ordered data delivery for packet-oriented connections across a wide-area network while keeping a very small code footprint.

The RTL guarantees that if packets the application sent do not arrive on the other end, it will notify the application of the failure. Based on a half-duplex, one-bit sliding window protocol,² which lets one party send only one message before waiting for a response or acknowledgment from the other party, the RTL uses an algorithm originally designed for the datalink layer. The level of service that the UDP/IP/PPP stack provides is analogous to the datalink layer.

The RTL invokes the security layer to encrypt, decrypt, authenticate, and provide integrity for all RNP payloads. Layering lets the security facility assume it is receiving packets in order and without transport-induced errors, thus making autokey stream ciphering—in which encryption depends on all previous text—possible.

Resource constraints

Various cost and power constraints, listed in Table 1, preclude using dedicated encryption hardware or a HAS coprocessor to achieve network security. The protocol stack's security layer can consume only a fraction of the limited RAM, ROM, and processor cycles available. For example, the HIM uses an inexpensive 8-bit CPU supporting a heavily multitasked embedded control system.

Because the HIM communicates with the GHS via a slow, 2,400-baud modem, minimizing bandwidth and execution time is also important. In many coun-

tries, the telecom industry charges customers for local telephone time when they remotely access their homes.

Encryption algorithms

We tried several encryption algorithms to see if they would fit the resource constraints, starting with memory. We ruled out using a hybrid cryptosystem, the most common scheme, because it uses public-key encryption for authentication and key exchange and a secret-key algorithm for confidentiality. Having separate public- and secret-key algorithms would use much more memory than we had available. Eliminating the secret-key algorithm and using public-key encryption for confidentiality would reduce the memory required, but not by enough, and public-key encryption of every message would exceed our time and bandwidth constraints by a wide margin.

We thus had no choice but use a secret-key algorithm for both authentication and confidentiality. Because all communication passes through the GHS, and the GHS does not have storage constraints, we can easily use paired secret keys to implement secret-key authentication. Authentication and integrity can be added to a message via a secret-key-based message authentication code, which typically uses a secure hash function seeded with a secret. However, embedded-system security constraints usually cannot accommodate a separate MAC algorithm in addition to a confidentiality algorithm.

A confidentiality algorithm can also perform authentication and integrity if it propagates message changes forward to the end of the message and appends a known value. If decrypting the message returns the known value, the message is authentic and intact. Block ciphers can be used in a mode with this propagation property.³ Some stream ciphers use an autokey mechanism, which feeds either ciphertext or plaintext back into the algorithm's state to affect subsequent encryption. Plain-fed autokey, or plaintext feedback, propagates any change in the message through to the message's end; cipher-fed autokey, or ciphertext feedback, propagates errors forward for only a limited length of ciphertext.

Our first secret-key algorithm candidate was Rijndael, selected as the National Institute of Standards and Technology's Advanced Encryption Standard.⁴ However, a trial implementation of Rijndael on the HIM processor exceeded the memory limits just trying to implement confidentiality, making it pointless to add integrity and key management. These results were consistent with those achieved using a Motorola 6805,⁵ which has a similar processor.

We next tried algorithms with reputedly small memory requirements. The extended tiny encryption algorithm (XTEA)⁶ fit the memory constraints. However,

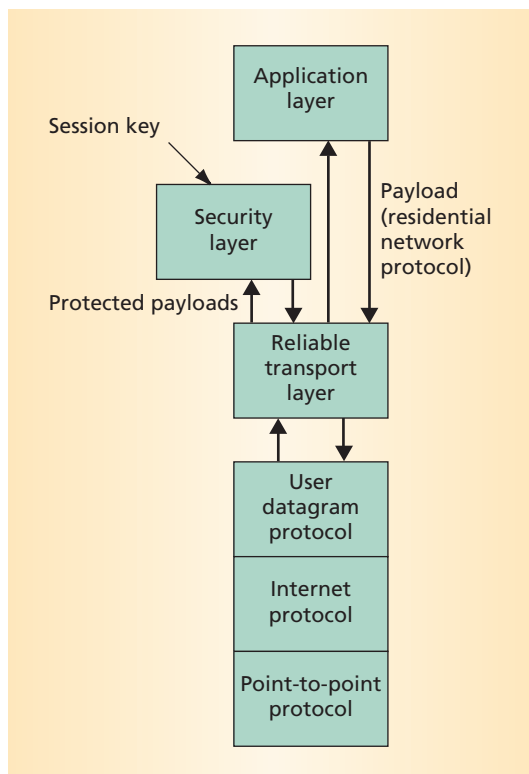


Figure 3. Between each application and the user datagram/Internet/point-to-point protocol stack are security and reliability protocol layers designed specifically for limited-resource processors.

adding a simple integrity mechanism using cipher block chaining mode with a running sum of the intermediate values—plaintext exclusive- or ciphertext—for the integrity check value exceeded the ROM limits. A quick look at the Skipjack algorithm⁷ indicated that it would be better than XTEA for RAM but not ROM.

We then tried an existing proprietary algorithm designed for higher-end embedded CPUs and digital signal processors—which tend to be 32-bit machines—and optimized for real-time encryption. Surprisingly, a solution based on this algorithm fit into 28 bytes of RAM, 28 bytes of electronically erasable programmable ROM, and 1,628 bytes of ROM—954 of it for the encryption routine—even though we had to synthesize the algorithm's 32-bit operations out of 8- and 16-bit instructions. This algorithm thus met the memory constraints, whereas the other candidates

Table 1. Communications security layer resource constraints.

Algorithm need	Constraint	Resource
Variables	50 bytes	Random-access memory
Semiconstants	40 bytes	Electronically erasable, programmable read-only memory
Code and constants	1,638 bytes	Flash EPROM
Bandwidth	2,400 baud	Built-in modem
Message overhead	Minimize	Cost per connect minute
Power consumption	As low as possible	Battery backup
Speed	Keep up with 2,400 baud in multitasking environment	8-bit CPU

exceeded the memory limits without even including key management.

The next resource constraint was minimizing message size expansion. Most encryption algorithms need some expansion. Basic block ciphers round message sizes up to be an exact multiple of the block size. Stream ciphers and block ciphers that use streamlike modes need an initialization vector. Our algorithm is a stream cipher that does not need block padding. Given the RTL layer's guarantees of error-free and in-order delivery, we can carry over the encryption algorithm's state so that it handles a session's multiple messages like packets of one large message. This eliminates the need for initialization vectors on the messages within a session and also prevents message replay attacks. The security layer's key management routines deal with session replays.

Because we specifically designed our algorithm to meet the low latency requirements of real-time systems, its execution speed could easily keep up with a 2,400-baud modem.

Key management

During installation, each HIM receives a key-message key known only to the HIM and the GHS. The method for securely delivering key-message keys from the GHS to each HIM depends on the particular GHS network operator. At the beginning of each session, the GHS generates a truly random session key—via electrical noise rather than a pseudorandom algorithm—and sends it to the HIM encrypted with its key-message key.

The initialization vector for this message includes a session sequence number to prevent session replay attacks. The GHS and the HIM then exchange messages with integrity checking. If the integrity checks determine that the correct sequence number was used—allowing for off-by-1 sequence errors—mutual authentication occurs, enabling continuation of the session and updating of the sequence number. If the checks fail, the GHS negative acknowledges the session-key message and tries to restart the session with a new session key. The key-message and session keys are 224 bits each—larger than necessary to prevent brute-force attacks (trying every key). The excess key size helps make the algorithm small and fast.

The GHS system is operational with initial product deployments in both the United States and Europe. As we continue to expand communication limits while maintaining security, we are developing products that use different processors and media such as RF pagers and broadband, can function in aircraft as well as in homes, and perform various new applications. Accompanying the emergence of pervasive computing, which will offer far more

people improved automatic control of their living and working environments, will be a need for secure communications using very limited resources. ✨

Acknowledgments

We thank all members of the development team, especially Denis Foo Kune, who designed and implemented the RTL, and Charles Obranovich, who integrated the security layer and RTL.

References

1. B. Schneier, "Why Computers Are Insecure," *Crypto-Gram*, Nov. 1999; <http://www.counterpane.com/crypto-gram-9911.html>.
2. A.S. Tanenbaum, *Computer Networks*, Prentice Hall, Englewood Cliffs, N.J., 1989.
3. B. Schneier, *Applied Cryptography*, John Wiley & Sons, New York, 1996.
4. J. Daemen and V. Rijmen, "AES Proposal: Rijndael," June 1998; <http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf>.
5. G. Keating, "Performance Analysis of AES Candidates on the 6805 CPU Core," revision of paper presented at the 2nd AES Candidate Conf., Apr. 1999; <http://members.ozemail.com.au/~geoffk/aes-6805/paper.pdf>.
6. R. Needham and D. Wheeler, "Tea Extensions," draft tech. report, Computer Laboratory, University of Cambridge, UK, Oct. 1997; <http://www.ftp.cl.cam.ac.uk/ftp/users/djw3/xtea.ps>.
7. United States National Security Agency, "Skipjack and KEA Algorithm Specifications," v2.0, May 1998; <http://csrc.nist.gov/encryption/skipjack/skipjack.pdf>.

Peter Bergstrom is a principal research scientist at Honeywell Laboratories. His research interests include secure, Internet-based services and equipment health management. Bergstrom received a BA in computer science from Macalester College. He is a member of the IEEE. Contact him at peter.a.bergstrom@honeywell.com.

Kevin Driscoll is a senior staff scientist at Honeywell Laboratories. His research interests include real-time, embedded, safety-critical, and secure systems' architecture. Driscoll received a BS in computer science from the University of Minnesota. Contact him at Driscoll@htc.honeywell.com.

John Kimball is a principal research scientist at Honeywell Laboratories. His research interests include security for Internet devices and tooling for testing safety-critical software. Kimball received an MS in computer science from the University of Illinois at Urbana-Champaign. Contact him at john.kimball@honeywell.com.