OPENNESS AND SECRECY IN SECURITY SYSTEMS: POLYCIPHER [SM] DOWNLOADABLE CONDITIONAL ACCESS

Tom Lookabaugh, PolyCipher
James Fahrny, Comcast Cable

Abstract

The PolyCipher Downloadable Conditional Access System provides a new approach to security in cable: a small hardware footprint is used to enable both high security and the flexibility of software downloadable security clients for set-top boxes and other cable-ready devices. An important and often asked question is how much of this system specification should be disclosed publicly? Here we explain the tradeoffs involved in disclosure and motivate the ultimate choice: a combination of public cryptographic primitives embedded in a private defense-in-depth system.

INTRODUCTION

PolyCipher is developing with Cable Television Laboratories a proposed foundation for downloadable conditional access (DCAS) for the U.S. cable industry. This system represents a major departure in cable security system design, opening up the possibilities of lower cost and increased flexibility in managing access to cable content, while maintaining backwards compatibility with the industry's installed base of security equipment.

In designing a major security system, there are important questions on what kinds of information are made public and what is kept secret. The choices made affect both the basic security of the system and the industry and the range and simplicity of implementations.

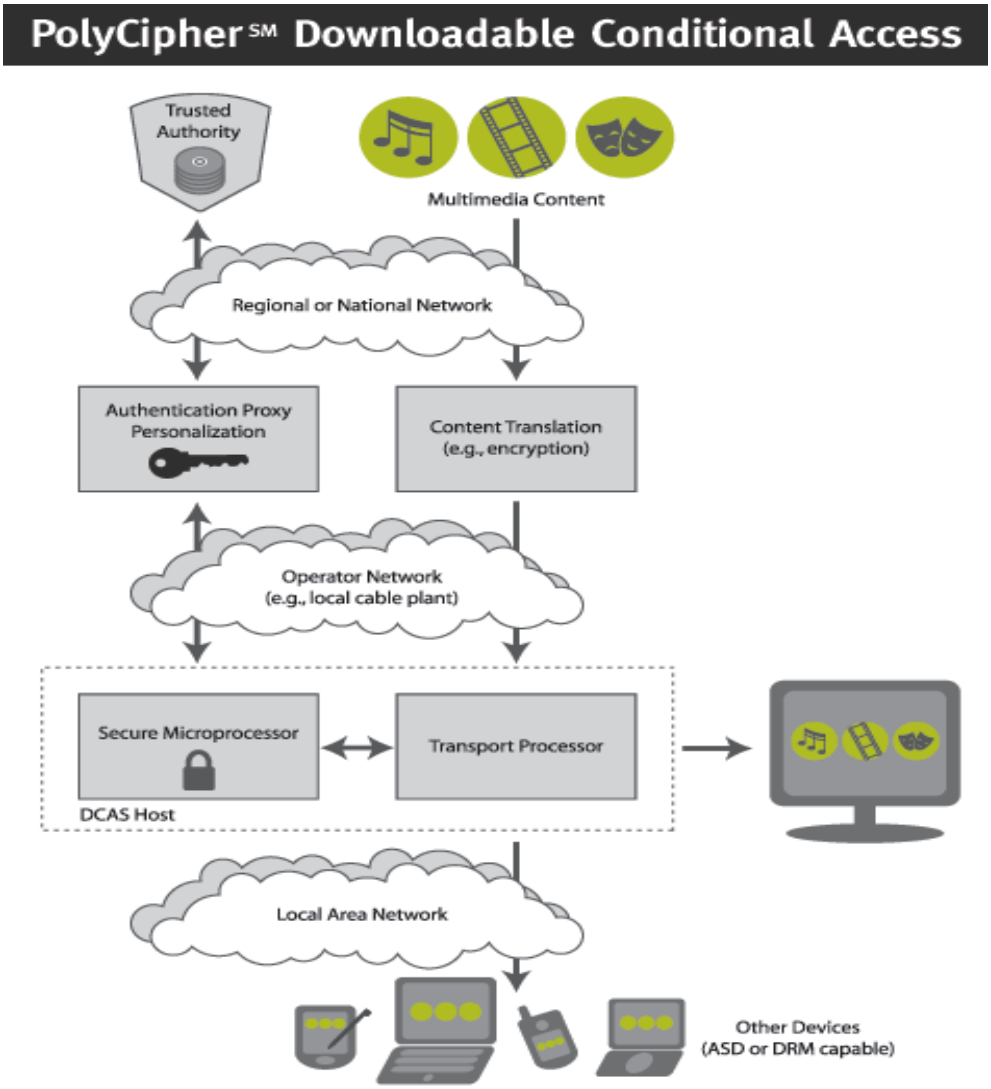
The principle security paradigm employed in DCAS is "defense-in-depth." In the paper we explain how this contrasts and interacts with other important concepts in security design, including "security by design," "security by obscurity," Kerckhoffs' principle, the economics of attack and defense, effective approaches to security system review and qualification, and the role of open source, cryptographic primitives, and modes of standardization for security systems.

The resulting choices are intended to produce a robust security system approach that can achieve both lower costs for the industry and appropriate security to maintain the industry's unparalleled access to high quality content over the coming decades.

DOWNLOADABLE CONDITIONAL ACCESS

The PolyCipher Downloadable Conditional Access System (DCAS) is an emerging architecture designed to bring increased power and flexibility to the cable industry's effort to combat the piracy of its video, audio and other content.

Specifically, the PolyCipher DCAS architecture is focused on delivering security-related software clients to



A DCAS Host (a set-top box, TV set, or other compliant device) establishes its bona fides by contacting a trusted authority (a database of all authorized devices) via an authentication proxy.

Once authenticated, the DCAS host is personalized with the appropriate security client that will allow it to decrypt and display content transmitted over the cable network (typically, video & audio). Content can also be translated for use by ASD- or DRM-capable devices elsewhere on a local network.

compliant cable-ready hosts, including:

- Set-top boxes and devices
- Cable-ready televisions
- Home entertainment systems
- Cable-ready mobile and portable devices
- Other emerging products

Content security for these devices has traditionally been handled via hardware, through some combination of set-top devices

or the installation of CableCARD™s. Unfortunately, this hardware module-driven approach requires significant manual effort to upgrade or change security systems at the cable customer level. Furthermore, hardware modules must be shipped, inventoried and repaired, all of which drives up operating expenses and limits the flexibility of the cable operator.

The PolyCipher DCAS architecture eliminates all this hardware module shuffling because it allows security systems to be automatically downloaded to compliant devices, using the existing cable infrastructure. Furthermore, the PolyCipher DCAS specification provides control over a broad range of security-related functions, including traditional conditional access systems (CAS: control at the host device of access to content), authorized service domain (ASD: control of other local devices all secured by the cable operator's security system), and digital rights management (DRM: bridging to other security systems on other local devices).

Hardware Architecture

The PolyCipher DCAS hardware architecture includes a Secure Micro (SM) and a Transport Processor (TP). The SM is a hardened and limited-capability microprocessor that primarily enables the decryption of multiple video streams, under direction of the installed CAS client. It does this by providing the necessary key management services for the TP and enabling a secure bootstrap of the software system [1].

The download of clients (CAS, ASD or DRM) to the SM is securely managed in the network operator's headend via the interaction of the SM software and a DCAS authentication proxy. The TP is primarily used for encrypting and decrypting the video and media protected by the SM clients.

Software Architecture

The PolyCipher DCAS specification defines many key elements:

- Messages between the SM and the DCAS servers in the headend environment
- Requirements for SM and DCAS Hosts to support DCAS
- Requirements of the headend server
- A new key management infrastructure

The new key management infrastructure supports the DCAS architecture by providing custom protocols, performance and security requirements, and by defining the necessary levels of interoperability, accountability and security.

OPENNESS AND SECRECY IN DCAS

In simple terms, a security system can be thought of as an algorithm and a key. The algorithm explains what happens to accomplish the functions desired if the key is known. The key is a piece of data that is kept secret except to those who need to use it to operate the system.

While there is no debate about the importance of keeping the key secret, there is an ongoing debate on when to keep the algorithm secret.

Making the algorithm public allows for a broad community with diverse tools and perspectives to carefully evaluate it. Many an assistant professor or graduate student can win glory (and maybe tenure) by finding a critical flaw in a proposed algorithm. The notion that a security system should only rely

on the secrecy of the key for its security goes by the name Kerckhoffs' principle [2, 3]. A frequent complement to Kerckhoffs' principle is the notion that reliance on the principle can only be reasonably assured by submitting a security system to broad public scrutiny.

Many important security primitives (algorithms focused on a narrow security function) are publicly vetted in exactly this way, examples include AES, RSA, and a variety of others that are used commonly in security systems of any scale. Some, like DES, were originally created in secret but later subjected to intense public scrutiny.

More generally, software systems such as operating systems and browsers have been the subject of a debate on the relative merits for security of open and closed source. These systems can be quite complex but are also of intrinsic interest to a large community of developers. There does not seem to be a definitive answer as to whether such systems (complex but of broad interest) are more secure in open or closed source form, although a number of authors give the edge to recommending open source for security [4, 5, 6, 7].

However, many large security systems do not fully subscribe to public scrutiny but choose, instead, to keep substantial parts of the security system algorithms private. A relevant alternative paradigm, especially common in pay-TV systems, is called "defense in depth" [8]. This approach views security as an economic competition between the security operator and his opponents (pirates and hackers); the goal is not perfect security – which is deemed to be unachievable – but rather a situation in which the economic gain to the pirates and economic losses of the security operator are both sufficiently low to maintain the viability of the operator's business model. Defense in depth envisions a series of counter measures, all kept secret, which are sequentially deployed when and if previous deployed counter measures are breached. The idea is that with each deployment, there is an extension of the period of low gain for pirates and low loss for the operator.

So, what should reasonably decide which kind or which parts of a security system algorithm should be public versus secret?

A useful answer is: when the system is simple enough that the cost to evaluators to test it is more than offset by their expected gains (in money or, more typically, fame and reputation), the public scrutiny that is frequently equated with Kerckhoffs' principle will be valuable. Conversely, when the system is complex enough and of limited enough interest that public scrutiny will only result in incomplete vetting, public disclosure may backfire, since an attacker needs find only one flaw in a public algorithm to breach it (while the evaluators have the much harder task of attempting to test and eliminate all possible flaws). As Schneier points out, there are limits to the amount of gratis work one can expect of the security community: "Security researchers are fickle and busy people. They do not have the time, nor the inclination, to examine every piece of source code that is published." [9, pp. 343-346]. Or, in Anderson's words: "Arguments against open source center on the fact that once software becomes large and complex, there may be few or no capable motivated people studying it, hence major vulnerabilities may take years to be discovered….the important questions are how much effort was expended by capable people in checking and testing the code – and whether they tell you everything they find" [8, pp. 296-207].

The Data Encryption Standard (DES) encryption algorithm, for example, can be described in about 100 lines of source code. It is straightforward for academics, hobbyists, and professionals to understand

and analyze it in detail from many different perspectives. The same is true of many cryptographic primitives.

The protocols, procedures, and algorithms involved in a full scale conditional access system sit near the other end of a continuum. Describing these fully could easily run to thousands of pages of documentation. It is so expensive to fully comprehend these that the amount of evaluation that can be expected (other from those explicitly paid to do so) is quite limited. And the intrinsic motivation for the security community to protect a particular instance of a conditional access system could reasonably be expected to be much less than that for a widely used application like an operating system or browser. Moreover, systems this complex are simply never bug free; the combinatorics of analysis and testing make this infeasible. The result is that such systems are rarely if ever made public. This does not mean there is an intention that the secrecy of the algorithm is its sole defense; indeed Kerckhoffs' principle is as much an objective here as it is in a cryptographic primitive. But both the reality of large scale system creation and system test and the particular economic incentives of system creators, pirates, and potential reviewers mean that striving for the goal of Kerckhoffs' principle is supplemented by the use of defense in depth to manage the economics that are the fundamental driver in protecting a commercial conditional access.

Swire has developed a thoughtful analysis of the economic, legal, and regulatory implications of tradeoffs in security system disclosure in a pair of papers [10, 11]. There he provides a useful comparison with military cryptography (remember, Kerckhoffs was in fact addressing military uses): why is it that militaries consistently find it valuable to keep cryptographic algorithms secret, even while adhering to Kerckhoffs' idea that they shouldn't design with a dependence on algorithms' secrecy for their success? Ultimately, the analysis there is economically motivated – as in this paper: if defenders profit more from exposure than attackers, then *disclosure* is valuable; if not, then not. Note again, though that Kerchoffs' principle is always valuable in *design*.

The implications for open standardization of security systems follow. Security primitives certainly benefit from the evaluation possible in an open standards setting (although this is not the only way to obtain public scrutiny – for example, an alternative is to publish a patented algorithm and provide a prize to those who breach it). Large scale security systems benefit if they proportionally scale in their interest to the security community – so experts find it worth their while to provide free scrutiny. Less interesting systems, such as a particular conditional access system implementation, do not benefit from open standardization – in the sense that security is weakened rather than improved in the particular and deciding context of system economics. This leaves open the exact boundary between a security primitive and a large scale system, but the principle is clear.

PolyCipher Approach

The PolyCipher approach attempts to find the best combination of open, public algorithms and closed, defense-in-depth system design. Cryptographic primitives used in the system design are drawn from those that are widely used and well established, such as the DES and AES encryption algorithms.

The overall system is not public, though, and is divided into tiers of system design and documentation that are increasingly access restricted. Although the overall system is too complex to expect that any substantial gratis vetting would be applied if the system were made public, *paid* vetting by experts is

extensively applied. Additionally, stakeholders who will depend on the security of the system and who are themselves expert in this type of security are invited to audit the system design. Motivations for stakeholders vary and in many cases they can be expected to be quite critical of the system. The resulting review environment is contentious but thorough.

CONCLUSION

The PolyCipher DCAS system offers a new strategy for providing hardware rooted downloadable security for the cable industry. The question of how much of the system to publicly disclose is an important one and requires careful thought. But after consideration, the complexity of the system and narrowness of application suggests a hybrid approach: the use of well vetted publicly known cryptographic primitives embedded in a "defense in depth' system subject to extensive but private review.

REFERENCES

[1] William A. Arbaugh, David J. Farber, Jonathan M. Smith, "A Secure and Reliable Bootstrap Architecutre," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 4-7, 1997.

[2] August Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaries*, vol. IX, pp. 5-38, Jan. 1883, pp. 161-181, Fev. 1883.

[3] Bruce Schneier, "Secrecy, Security, and Obscurity," *Crypto-Gram Newsletter*, May 15, 2002, available at http://www.schneier.com/crypto-gram.html.

[4] Brian Whitten, Carl Landwehr, and Michael Caloyannides, "Does Open Source Improve System Security," *IEEE Software*, September/October 2001, pp. 57-61.

[5] Crispin Cowan, "Software Security for Open-Source Systems," *IEEE Security and Privacy*, January/February 2003, pp. 39-46.

[6] Rebecca T. Mercuri, "Trusting in Transparency," *Communications of the ACM*, May 2005, pp. 15-19.

[7] Jaap-Henk Hoepman and Bart Jacobs, "Increased Security Through Open Source," *Communications of the ACM*, January 2007, pp. 79-83.

[8] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, New York: John Wiley & Sons, Inc. 2001.

[9] Bruce Schneier, *Secrets and Lies: Digital Security in a Networked World*. New York: John Wiley and Sons, Inc. 2000.

[10] Peter Swire, "A Model for When Disclosure Helps Security: What is Different about Computer and Network Security," *Journal of Telecommunications and High Technology Law*, vol.2, 2004.

[11] Peter Swire, "A Theory of Disclosure for Security and Competitive Reasons: Open Source, Proprietary Software, and Government Agencies," *Houston Law Review*, vol. 42, No. 5, January 2006.

AUTHORS

Tom Lookabaugh is with PolyCipher, 999 18th St., Su. 1925, Denver, CO 80202, and by email at tom.lookabaugh@polycipher.com.

James Fahrny is with Comcast Cable, 1500 Market St., Philadelphia, PA 19102 and with PolyCipher, 999 18th St., Su. 1925, Denver, CO 80202. He can be reached by email at Jim_Fahrny@cable.comcast.com.